

Radical Blue Gaming

Incredibly Innovative Gaming Solutions

Release Notes – RadBlue G2S Scope (RGS)

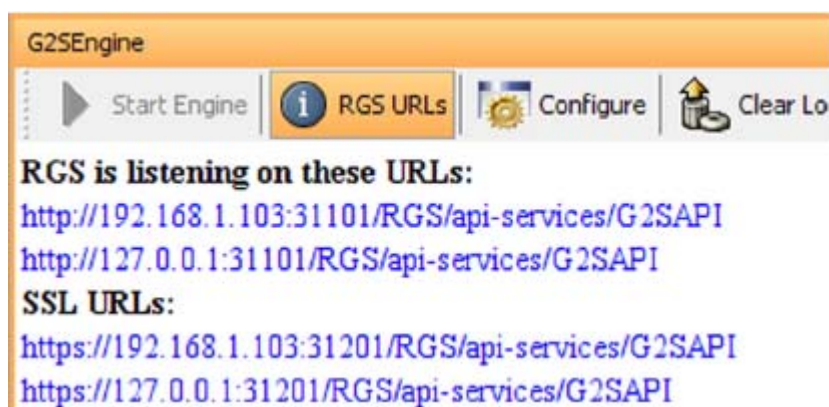
Version 1.8 [released: December 8, 2008]

High-Level Summary

In version 1.8, we made finding the RGS URL easier and improved the use of the download class on the Send Command layout.

Improvements

- A new **RGS URLs** button on the G2S Engine layout lets you easily find the URL that RGS is listening on. This feature is useful for setting up RGS and RST on two different computers, or when you're connecting RGS to an EGM.



- The *packageId* and *scriptId* from the latest `packageStatus` and `scriptStatus` responses are now used as default values for subsequent download commands as you move through the process of downloading content.
- RGS now accepts GZIP messages, which are compressed into HTTP_STACK, to the same path as non-GZIP messages.
- The `getTransportOptions` command has been modified to return NO_GZIP and GZIP_IN_HTTP_STACK.

Corrections

- When an EGM tries to redeem an invalid voucher, the RGS now correctly sends a voucher authorization response with *hostAction*=G2S_reject and *hostException*=4 (Voucher Not Found).

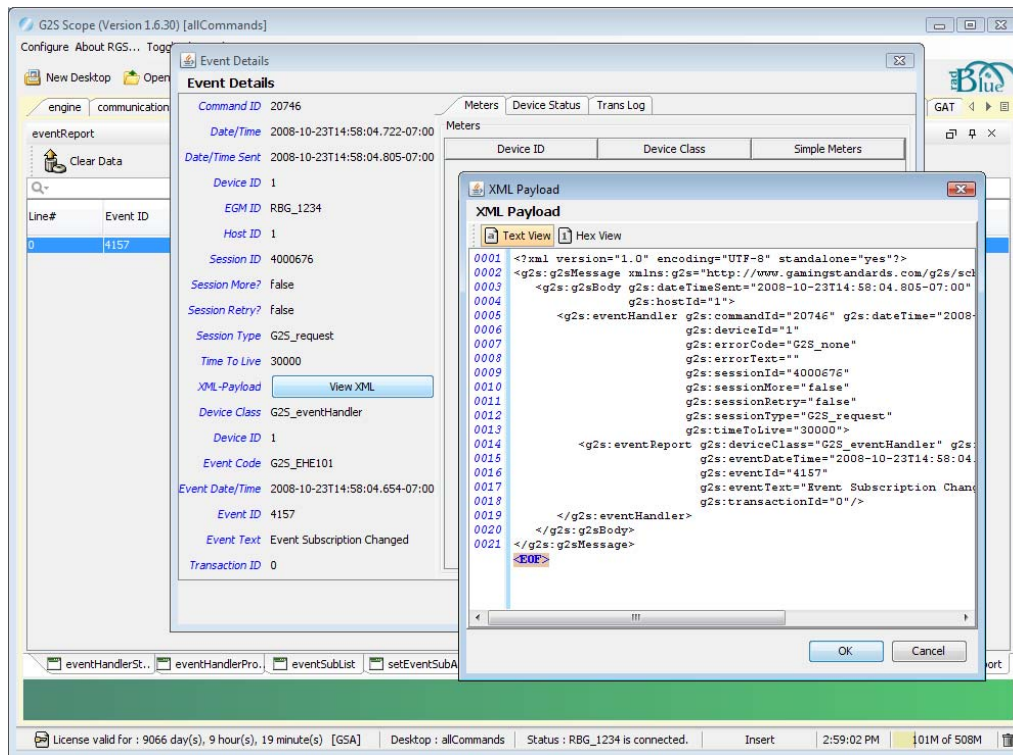
Version 1.7 [released: November 3, 2008]

High-Level Summary

In version 1.7, we corrected a few minor issues.

Corrections

- RGS now honors the retry flag in the `eventReport` message - no `eventId` errors are reported for retried `eventReport` messages.
- When the `eventId` attribute in an `eventReport` message is received out-of-order, RGS now logs a warning rather than an error.
- On the **About RGS** screen, the build date is now updated whenever a build occurs.
- Previously, the **View XML** window remembered its resized value only when the height *and* width were both changed. Now, if you change either the height *or* width, the new size is retained.

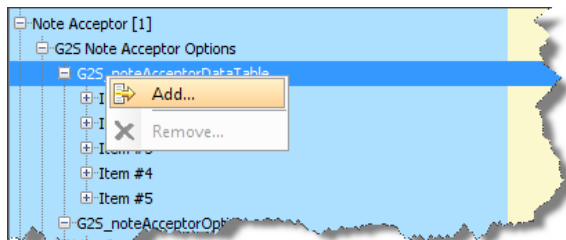


Version 1.6 [released: September 29, 2008]**High-Level Summary**

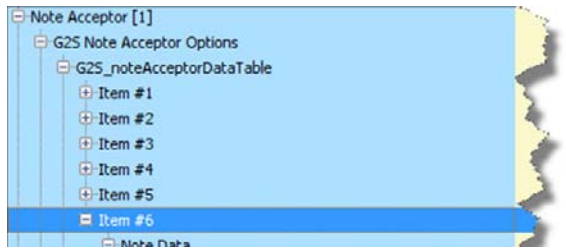
In version 1.6, we made a few minor corrections.

How to Add a Row to the OptionConfig Data Table

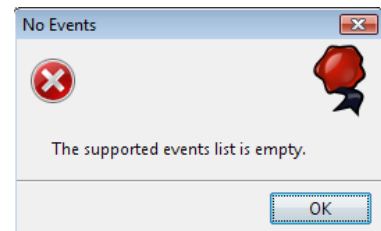
1. Navigate to: **Send Command > G2S_optionConfig > Option Config – Set Option Change.**
2. Under **Configuration Options**, select **Note Acceptor > G2S Note Acceptor Options** for the note acceptor device you want to modify.
3. Right-click **G2S_noteAcceptorDataTable**, and select **Add**.



A new Item is added to the end of the data table.

**Corrections**

- An error message - *The supported events list is empty.* - has been added. This message indicates that a supportedEvents command must be received by RGS before an eventHandler.setEventSubscription command can be sent. (The supportedEvents command is used to populate the Event Handler – Set Event Sub screen.)
- If the start-up algorithm is configured not to send any messages, the RGS now handles that change correctly.



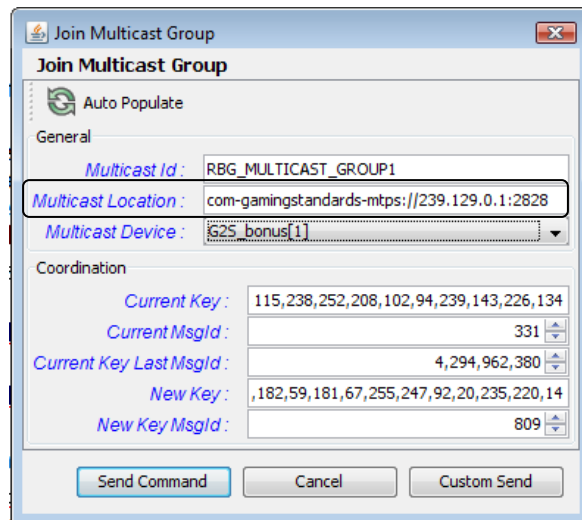
Version 1.5 [released: September 2, 2008]

High-Level Summary

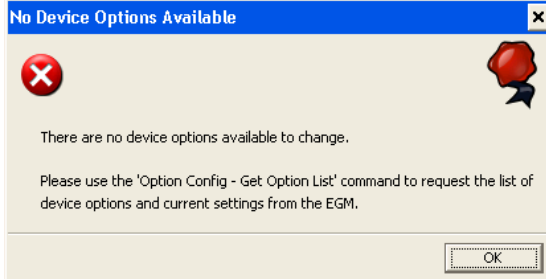
In version 1.5, we have updated the Multicast transport to reflect the changes made in the GSA Multicast Transport Protocol 1.0.7. We have added new configuration options to give you even more control over how RGS handles messages and corrected several issues.

Improvements

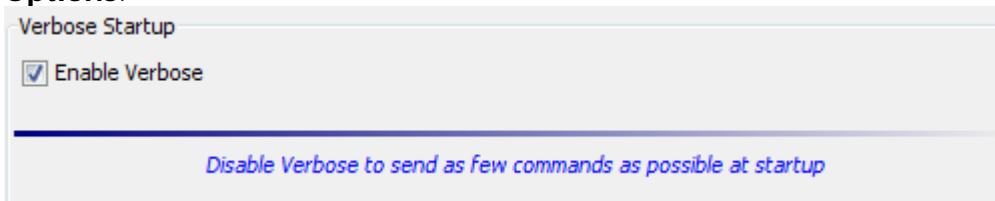
- Multicast has been updated in accordance with the GSA Multicast Transport Protocol v1.0.7.
 - Test vectors have been implemented (as recently discussed in the GSA Transport Committee). Test vectors allow vendors to run test input against mtp schemas and verify that the results are the same. Our Multicast implementation has also been vetted by two major gaming manufacturers.
 - Multicast now uses a separate command ID sequence for Multicast messages.
 - New Multicast URI schemes have been added to RGS.
 - com-gamingstandards-mtp://[ip-address]:[port]** – unencrypted but authenticated, using UMAC.
 - com-gamingstandards-mtps://[ip-address]:[port]** – encrypted and authenticated, using UMAC-AE. (default)
 - The `communications.joinMulticast` command under Send Command defaults to “mtps” when you auto-populate the multicast information.



- The `optionConfig.setOptionChange` command GUI requires that a list of device options first be requested through the `optionConfig.getOptionList` command. If RGS does not have a list of device options when a `optionConfig.setOptionList` command is selected, the following message displays:



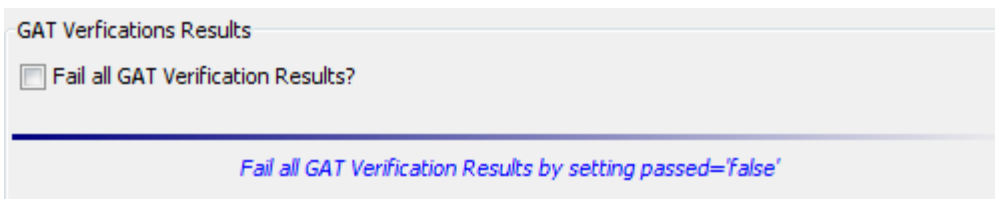
- An **Enable Verbose** option has been added under **Configuration > Engine Options**.



If this option is cleared, when the `commsOnline` message is sent, RGS does not enable devices that the EGM reports as already enabled. If this option is selected, RGS enables all devices regardless of whether they may already be enabled in order to have the EGM execute as wide a variety of commands as possible.

Running RGS with this option deselected most closely simulates host traffic.

- A **Fail All GAT Verification Results?** option has been added under **Configuration > Engine Options**. Select this option to control the `gat.verificationResultAck` response. If this option is selected, the *passed* attribute in all `gat.verificationResultAck` responses will be "false" for each component ID from the `gat.verificationResult` command.



- The *timeToLive* attribute is automatically set to zero (0) on command responses and notifications. The *timeToLive* attribute is the number of milliseconds an endpoint should wait before ignoring a command.

Corrections

- If the *deviceChanged* attribute or the *deviceReset* attribute in the `communications.commOnLine` command is true, subscriptions are always sent (startup algorithm is executed). If both attributes are false, the startup algorithm is not executed.
- RGS now properly responds to an EGM-initiated `optionConfig.optionList` commands with an `optionConfig.optionListAck` Response message.
- The RGS voucher database now supports escape characters in the EGM ID.
- GameDenom, Currency and Wager meters are now updating correctly on the meterInfo screen.
- The `setScript` command for packages (**Send Command > g2s_download > Set Script for Package**) now correctly sends authorization host data.

Version 1.4 [released: June 30, 2008]

High-Level Summary

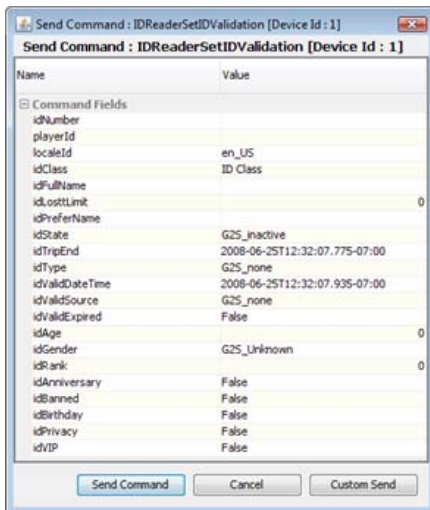
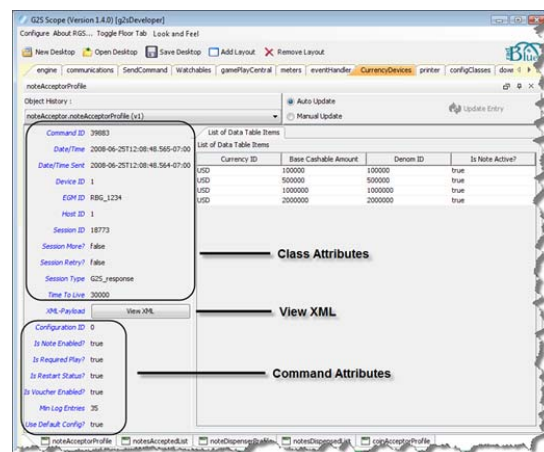
In release 1.4, various usability improvements were made, including the addition of a new **g2sDeveloper** desktop, corrections were made to commsConfig functions.

Improvements

- A new **g2sDeveloper** desktop has been added to RGS. This new desktop represents our revised view on the objects that are needed in the tool. Many of the simple objects that repeat what is shown in the transcript have been removed, providing a much simpler interface to the complex commands.

- The information displayed in the left-hand panel of command object layouts is now sorted in the following order, making it much easier to find a particular attribute:

1. class attributes
2. View XML
3. command attributes



- The `idReader.setIdValidation` command (**Send Command > G2S_idReader > ID Reader – Set ID Validation**) is now completely configurable – all attributes are available on the control.

Corrections

- The `setCommChange` command (**Send Command > G2S_CommConfig > CommConfig – Set Comm Change**) has been updated to allow the *HostId* field to be set to a non-zero number.

Version 1.3 [released: May 27, 2008]**High-Level Summary**

RGS now supports the progressive class. A new Database floor tab has been added that contains a Voucher Database and a Progressive Database. These new screens allow you to view, edit and delete database information. Finally, new security (SSL) options can be set up through the Configure screen.

New Features

- RGS now supports the progressive class. See [Using Progressive](#).
- A new **Progressive** floor tab group has been added, containing the following new progressive command objects:

progressiveProfile	progressiveStatus
progressiveHit	progressiveCommit
getProgressiveHostInfo	progressiveLogStatus
progressiveLogList	progressiveValueAck
Progressive Database	
- A **Database** floor tab has been added. This floor tab contains two new database management objects: Progressive and Voucher. For information on the Progressive database management object, see [Using Progressive](#). For information on the Voucher database management object, see [Using the Voucher Database](#).
- The following progressive commands have been added to the Send Command object:

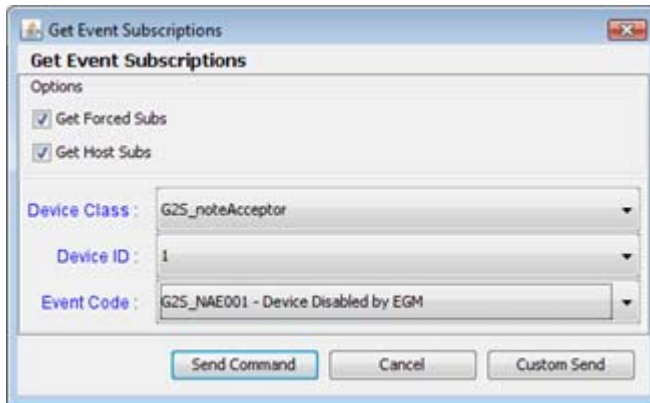
• Get Progressive Log	• Get Progressive Log Status
• Get Progressive Profile	• Get Progressive Status
• Set Progressive Lock Out	• Set Progressive State
• Set Progressive Value	• Set Progressive Value (via Progressive Database)

- Progressives have been added to **Send Command > G2S_optionConfig > Option Config – Set Option Change**.

Configuration Options		
Name	Current Value	New Value
Progressive [1]		
G2S Progressive Options		
G2S_progDataTable		
Item #1		
Progressive Data Table		
Denomination ID	100000	
Game Play ID	1	
Progressive Level	1	
Number of Credits	3	
Paytable ID	RBG_92	
Theme ID	RBG_sweatyTrolls	
Win Level Index	1	
Item #2		
Item #3		
G2S_protocolOptions		
G2S Protocol Parameters		
Configuration Identifier	0	
No Progressive Info	0	
No Response Timer	30000	
Progressive Identifier	10	
Required For Play	false	
Enabled on Restart	true	
Time to Live	0	
Use Default Configuration	true	

- The following options have been added to **Configure > Engine Options**:
 - Filter G2S Set Progressive Values from Transcript** – Select to exclude G2S setProgressiveValue messages from the Transcript and Debug Log. Note that setProgressiveValueAck messages are not filtered with this option. To filter the progressive ACK and the G2S ACK, use the Filter option available in the Transcript and Debug Log.
 - Progressives – Broadcast Interval** – Enter (in millicents) the interval between progressive updates. Zero (0) disables this field. The default is 0.
 - Progressive Updates Via Multicast** – Select to use Multicast to deliver progressive updates.
 - Progressives – Win Command Pay Method** – Select the progressive payment method sent in the setProgressiveWin command. Your choices are: handpay, voucher and any. "Any" indicate that the payment method is chosen by the EGM.

5. A new **Event Handler – Get Event Subscriptions by Wildcards** command under **Send Command > G2S_eventHandler** lets you request event subscriptions using G2S wildcards.

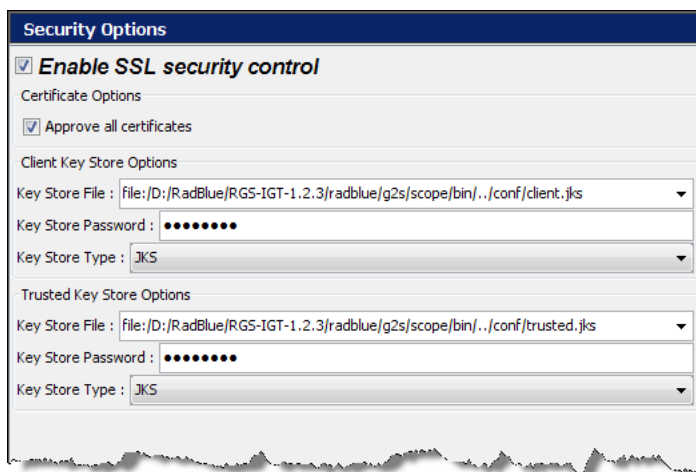


From the **Get Event Subscriptions** dialog box, you can request event subscriptions in a tiered manner:

- **Device Class** – Select **G2S_all** for all classes, or select a specific class. If you select a specific class, the Device ID field provides more options.
- **Device ID** – Select **-1** for all devices in a specific class, or select a specific device ID. If you choose a specific device ID, the Event Codes field provides more options.
- **Event Codes** – Select **G2S_all** to select all event codes, or select a specific event code.

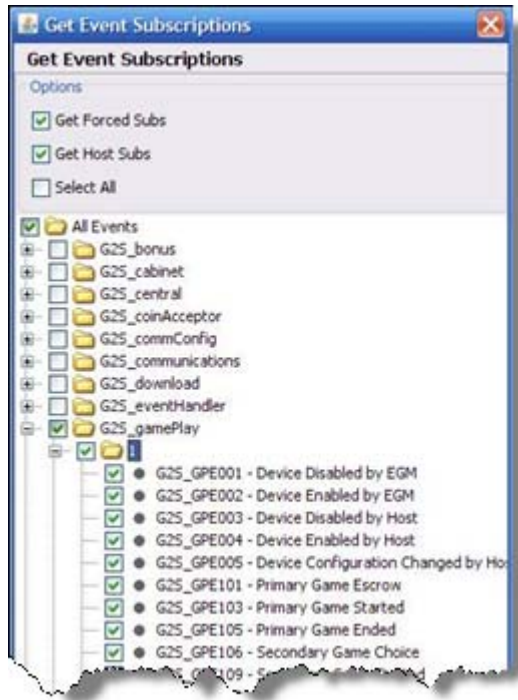
Improvements

- A **Security Options** screen has been added to the **Configure** option on the menu bar. From this screen, you can configure Secure Socket Layer (SSL) encryption information for the application.



- Select **Enable SSL security control** to enable encryption for the application.
- Select **Approve all certificates** if you want to use SSL encryption, but are not concerned with the validity of the certificate authority.
- Enter the **Key Store Options** for the client and the certificate authority.

- The events list in the `getEventSub` command (**Send Command > Current Devices > G2S_eventHandler > Event Handler – Get Event Subscriptions**) now allows you to select all events, a single class, and a single device ID by selecting a single checkbox.



- A new **ID Reader and Player** layout has been added, containing the following command objects:

<code>idReaderStatus</code>	<code>idReaderProfile</code>
<code>playerProfile</code>	<code>playerStatus</code>
<code>playerLogStatus</code>	<code>playerLogList</code>
<code>carryOverAck</code>	<code>hostPointsAck</code>
<code>playerMessageAck</code>	<code>getCountdownOverride</code>
<code>playerSessionStart</code>	<code>playerSessionEnd</code>

- For ease-of-use, the **Event Handler – Get Event Subscriptions** command under **Send Command > G2S_eventHandler** now sorts events alphabetically.

Using Progressive

Progressive Database

The Progressive Database management object displays the content of the RGS progressive database, and allows you to add, modify and delete progressive groups and levels.

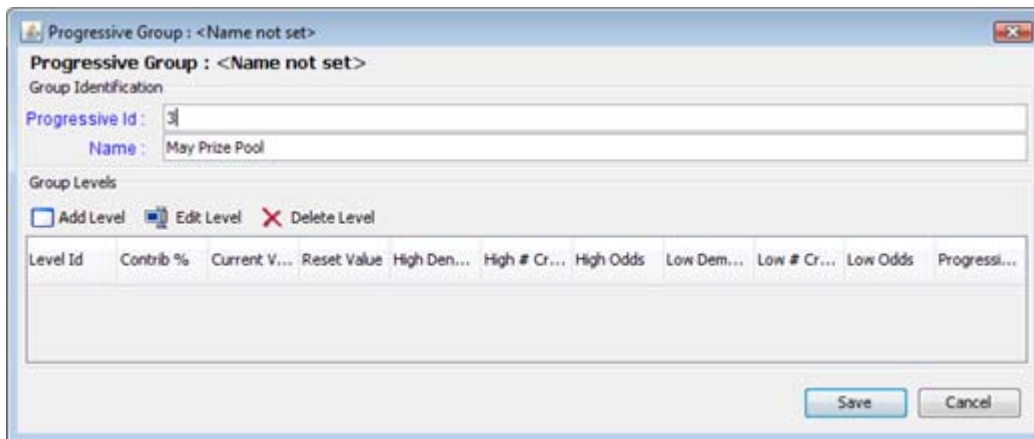
The event that updates the RGS progressive database is G2S_PGE101 (Progressive Money Wagered). If this event is not generated by the EGM (not supported or not subscribed to by the RGS), the RGS progressive values won't update when a game is played. The `setProgressiveValue` command is then used by the RGS to send the updated progressive values back to the EGM.

Add a Progressive Group

1. From the Progressive layout, select the **Progressive Database** object.



2. Click **Add Group**.



3. Enter the **Progressive ID** and the progressive **Name**.
4. Click **Save** to save the new progressive group.

Add or Edit a Progressive Level in a Progressive Group

1. Double-click the progressive group of the level you want to add or edit.
2. To add a new level, click **Add Level**.

or

To edit an existing level, highlight the level you want to edit, and click **Edit Level**.

The screenshot shows a dialog box titled "Level Id : 1". It contains the following fields and values:

Progressive Value Text :	
Level Id :	1
Contribution % :	0.02
Current Value :	0
Reset Value :	100,000
High Denomination Id :	1,000,000
High # of Credits :	4
High Odds :	4
Low Denomination Id :	100,000
Low # of Credits :	1
Low Odds :	1

At the bottom of the dialog box are two buttons: "Save" and "Cancel".

- **Progressive Value Text** – Description of the progressive prize. Note that this text is displayed in the player display of the SmartEGM when a progressive with a non-dollar value prize is hit (Current Value equals zero).
- **Level ID** – Identifier of the progressive group level.
- **Contribution %** - Percentage of each wager that is added to the prize for the specified level.
- **Current Value** – Starting dollar amount of the progressive prize.
- **Reset Value** – Dollar amount of the progressive prize reset.
- **High Denomination ID** – Denomination for the High # of Credits field.
- **High # of Credits** – Number of credits wagered in a high bet.
- **High Odds** – Best bet:odds ratio.
- **Low Denomination ID** – Denomination for the Low # of Credits field.
- **Low # of Credits** – Number of credits wagered in a low bet.
- **Low Odds** – Worst bet:odds ratio.

3. Click **Save** to save your changes.

Delete a Progressive Group

1. From the Progressive Database main screen, select the progressive group you want to delete.



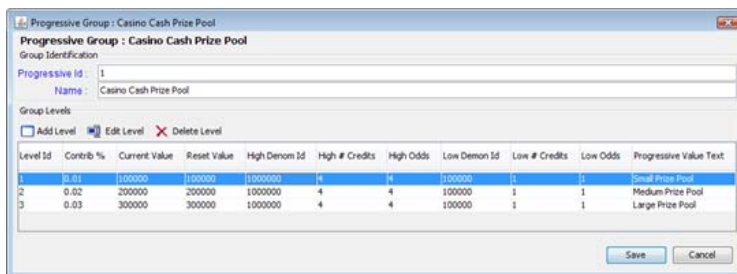
2. Click **Delete Group**.



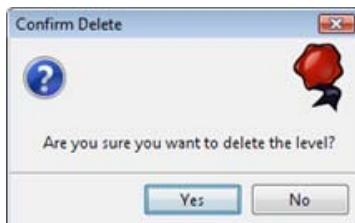
2. Click **Yes** to delete the selected progressive group.

Delete a Progressive Level

1. From the Progressive Database main screen, double-click the progressive group containing the level you want to delete.



2. Highlight the **Level ID**, and click **Delete Level**.



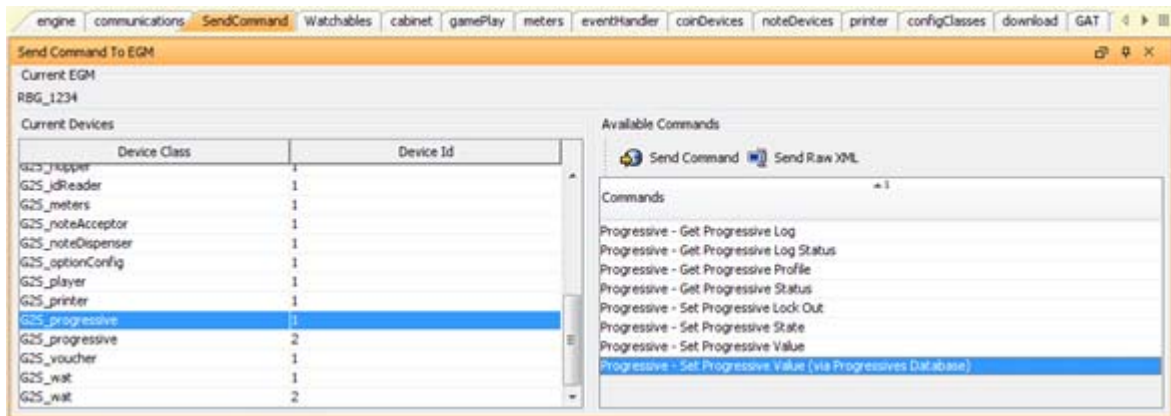
3. Click **Yes** to delete the selected level.

Progressive Send Command

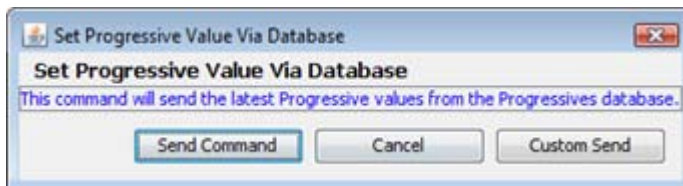
Send Updated Progressive Host Information to the EGM

This procedure allows you to send updated progressive host information to the EGM. The values sent in this command are current values for this progressive ID in the RGS progressive database.

1. From the **Send Command** object, select **G2S_progressive**.



2. Double-click **Set Progressive Value (via Progressive Database)**.

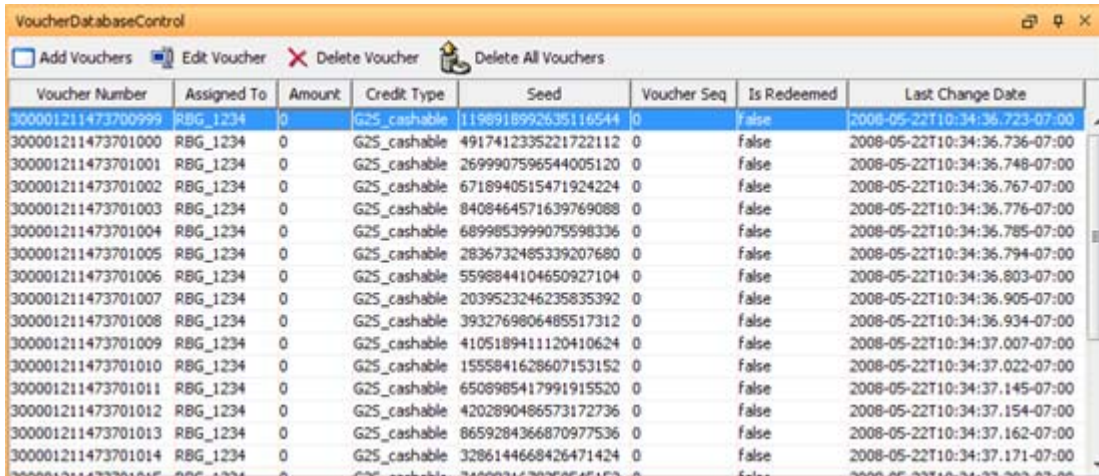


3. Click **Send Command**.
4. Follow steps 1-3 *for each progressive device* you want to update.

Using the Voucher Database

The Voucher Database object displays all records in the RGS voucher database. You can add voucher numbers to specific EGMs, edit voucher records, and delete voucher records. If an EGM sends a `getValidationData` command, RGS creates the voucher block automatically. However, you can manually create vouchers and change each voucher's characteristics as required for testing.

Voucher Database Object



The screenshot shows a window titled "VoucherDatabaseControl" with a toolbar containing "Add Vouchers", "Edit Voucher", "Delete Voucher", and "Delete All Vouchers". Below the toolbar is a table with the following columns: Voucher Number, Assigned To, Amount, Credit Type, Seed, Voucher Seq, Is Redeemed, and Last Change Date. The table contains 15 rows of data, all with an amount of 0 and a credit type of "G25_cashable".

Voucher Number	Assigned To	Amount	Credit Type	Seed	Voucher Seq	Is Redeemed	Last Change Date
300001211473700999	RBG_1234	0	G25_cashable	1198918992635116544	0	false	2008-05-22T10:34:36.723-07:00
300001211473701000	RBG_1234	0	G25_cashable	4917412335221722112	0	false	2008-05-22T10:34:36.736-07:00
300001211473701001	RBG_1234	0	G25_cashable	2699907596544005120	0	false	2008-05-22T10:34:36.748-07:00
300001211473701002	RBG_1234	0	G25_cashable	6718940515471924224	0	false	2008-05-22T10:34:36.767-07:00
300001211473701003	RBG_1234	0	G25_cashable	8408464571639769088	0	false	2008-05-22T10:34:36.776-07:00
300001211473701004	RBG_1234	0	G25_cashable	6899853999075598336	0	false	2008-05-22T10:34:36.785-07:00
300001211473701005	RBG_1234	0	G25_cashable	2836732485339207680	0	false	2008-05-22T10:34:36.794-07:00
300001211473701006	RBG_1234	0	G25_cashable	5598844104650927104	0	false	2008-05-22T10:34:36.803-07:00
300001211473701007	RBG_1234	0	G25_cashable	2039523246235835392	0	false	2008-05-22T10:34:36.905-07:00
300001211473701008	RBG_1234	0	G25_cashable	3932769806485517312	0	false	2008-05-22T10:34:36.934-07:00
300001211473701009	RBG_1234	0	G25_cashable	4105189411120410624	0	false	2008-05-22T10:34:37.007-07:00
300001211473701010	RBG_1234	0	G25_cashable	1555841628607153152	0	false	2008-05-22T10:34:37.022-07:00
300001211473701011	RBG_1234	0	G25_cashable	650885417991915520	0	false	2008-05-22T10:34:37.145-07:00
300001211473701012	RBG_1234	0	G25_cashable	4202890486573172736	0	false	2008-05-22T10:34:37.154-07:00
300001211473701013	RBG_1234	0	G25_cashable	8659284366870977536	0	false	2008-05-22T10:34:37.162-07:00
300001211473701014	RBG_1234	0	G25_cashable	3286144668426471424	0	false	2008-05-22T10:34:37.171-07:00

Each voucher record contains the following fields:

Voucher Number – Unique 18-digit voucher identification number.

Assign To – Identification number of the EGM that has been assigned the specified voucher number. "Undefined" means that the voucher number has not been assigned to an EGM. A voucher remains undefined until you assign an EGM to it or until it is allocated to an EGM is allocated through a `getValidationData` command.

Amount – Dollar amount of voucher, in millicents.

Credit Type – Cashable, non-cashable or promotional.

Seed – Number sent to the EGM for manual voucher validation.

Voucher Seq – Sequence number of the voucher assigned by the EGM at issuance. This number is printed on the voucher.

Is Redeemed – Indicates whether the specified voucher has been redeemed. "True" indicates that the voucher has been redeemed. "False" indicates that the voucher is in a state other than redeemed.

Last Change Date – Date and time the voucher record was changed (for example, from *active* to *pending* or any time the voucher record is edited).

Add Voucher Numbers to the Voucher Database

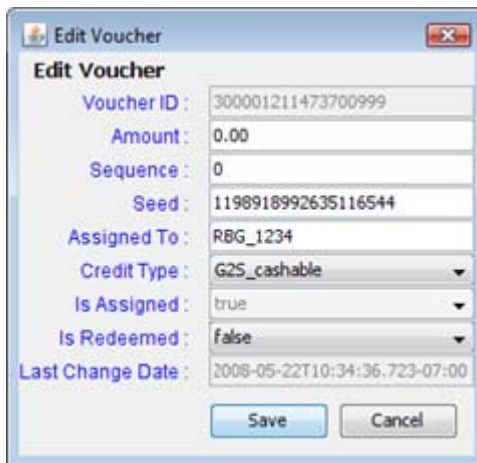
1. From the **Voucher Database** object, click **Add Voucher**.



2. Enter the number of vouchers you want to add.
3. Click **Load**. Note that the **Assigned To** field is "undefined." The new vouchers are appended the end of the voucher list.

Modify a Voucher Record

1. From the **Voucher Database** object, select a voucher record to edit.
2. Click **Edit Voucher**.



3. Modify the voucher record as required. Note that the **Voucher ID**, **Is Assigned**, and **Last Change Date** fields cannot be changed.
4. Click **Save** to save your changes, or click **Cancel** to exit Edit Voucher without saving your changes.

Delete a Voucher

1. From the **Voucher Database** object, highlight the voucher record you want to delete.

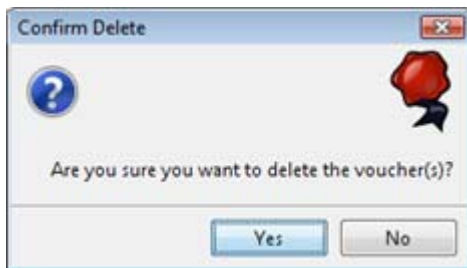
or

To delete a block of voucher records, select the first and the last voucher records to be deleted while holding down the SHIFT key.

or

To delete multiple, non-consecutive voucher records, select the voucher records to be deleted while holding down the CTRL key.

2. Click **Delete Voucher**.



3. Click **Yes** to delete the highlighted voucher record(s).

Delete All Voucher Records

1. From the **Voucher Database Control** object, click **Delete All Vouchers**.



2. Click **Yes** to delete all voucher records in the voucher database.

Version 1.2 [released: April 25, 2008]

High-Level Summary

In version 1.2, we have added the player class and a new EGM Transcript Analysis report, in addition to several improvements and corrections to RGS.

New Features

- Support for the player class has been added to RGS and RST. See [Using Player in RGS](#).
- A new EGM Transcript Analysis report has been added to the Transcript Control object. The EGM Transcript Analysis report provides information about messages that were sent from and received by the application for the period requested. For more information see the *EGM Transcript Analysis Report*.
- A `download.setScriptForPackage` command has been added to the Send Command object. This command allows you to install a package, uninstall a package, and execute (unpack) a package. You can specify the Apply condition, the Disable condition, the start and end date, and reason for the action.

Improvements

- The `gamePlay.setActiveDenoms` command can now be sent without setting any active denominations.
- The SSL installation and configuration has been updated to remove unused code. See *Bulletin 01: Using SSL in RadBlue Tools* for the latest information on SSL installation and use, available at <http://www.radblue.com/documentation.htm>.
- The maximum value of `setOptionConfig` numeric values has been expanded from 2.14 billion to 9.2 quintillion (9,223,372,036,854,775,807 or $2^{63}-1$). If the user enters the value greater than this limit for an Integer Parameter, an error message, "Entered value does not meet constraints," is displayed to the user.
- A new option, **Mark Vouchers as Redeemed**, has been added to **Configure > Engine Options**. Select this option to produce a "double redemption" error. Clear this option to redeem the same voucher multiple times without errors.
- A new option, **Set deleteCurrent attribute**, has been added to **Configure > Engine Option**. The `deleteCurrent` attribute is sent in the `validationData` command. This attribute is sent to delete the current validation and seed information on an EGM. When this option is checked, the `deleteCurrent` attribute is set to "true" in the `validationData` command, so the new validation ID numbers *replace* the current set in the EGM.

Corrections

- When an unknown device class is sent to RGS, the application now gracefully ignores it.
- Multicast messages are now encrypted correctly, if encryption is in use (they were coming through in plain text).
- An XML comment in a G2S message would cause the message to be flagged as invalid and the message would be ignored. XML comments are now gracefully ignored.
- RGS has been modified to correctly set the *scriptId* in the `getscriptStatus` command.
- The `communications.getCommsStatus` command is now sent by RGS as part of the start-up algorithm.
- The hopper class start-up algorithm now sends the `getHopperStatus` command if a hopper device is reported in the EGM descriptor list.
- The *bonusId* field in the `bonus.setBonusAward` command is no longer comma delimited in the GUI.

Using Player in RGS

The player class has been added to RGS. The following player commands have been added to the Send Command object:

- Set Player State
- Get Player Status
- Get Player Profile
- Set Player Override
- Set Countdown Override
- Set Countdown Override Multicast
- Set Carry Over
- Set Point Balance
- Set Host Points
- Set Player Message
- Set Player Message Multicast
- Get Player Log Status
- Get Player Log

A new **id-database.xml** file provides player and employee ID numbers. Once a player ID is inserted into the card reader, player overrides for countdown calculations can be set from the RGS Send Command.

The default ID numbers defined in **id-database.xml** are:

Player Numbers

- 12345678
- 22222222
- 11111111

Employee Numbers

- 99999999
- 88888888

Sample id-database.xml File

```
<id-database>
  <idRecord id="12345678" idType="G2S_player" playerId="P-12345678"
fullName="Elvis Presley" preferName="Elvis" overrideId="1" hostCarryOver="0"
pointBalance="1" playerStart="2008-04-15T10:35:37.299-07:00" playerEnd="2008-04-
15T11:25:37.299-07:00" playerTarget="100" playerIncrement="1"
playerAward="500"/>
  <idRecord id="99999999" idType="G2S_employee" playerId="E-99999999"
fullName="Rick Deckard" preferName="Rick"/>

  ...
</id-database>
```

The **id-database.xml** file is located in the `..\radblue\g2s\scope\conf` directory, and can be modified as needed.

Version 1.1 [released: March 4, 2008]

We've moved to a new numbering system for 2008, and version 1.1 requires a 2008 license (contact Russ@RadBlue.com if you haven't received your new license). For the foreseeable future, you can expect a new release of our tools around the end of each month. The minor number (x.1) will increment each month, and the major number (1.x) will increment on significant events in the life of the tool.

High-Level Summary

In this newest release, we added the G2S handpay class to RGS. Significant improvements have been made to optionConfig and download classes in addition to various program fixes.

New Features

- New versioning convention – RGS has a new versioning convention: [major.minor]. This versioning convention replaces the previous version 1.0.3 build x. The last RGS version using the old convention is 1.0.3 build 12.
- Support has been added for the G2S handpay class:
 - Display objects were created for all EGM generated commands (see the new handpay layout for examples)
 - Using the Send Command object, you can request handpay status, profile, and log information. You can also set the status of the handpay device.
 - You can initiate a `remoteKeyoff` command through the Send Command object, but you will need to know the handpay *transactionId* and the amounts to key off. You can easily discover these in the Handpay Request object (which shows complete details of the latest `handpayRequest`).
- Support for the `bonus.cancelBonusAward` command has been added. It can be accessed from the Send Command to EGM object.
- Because of the problems with implementing SOAP 1.2, we are dropping back to SOAP 1.1 for the moment, but will pay close attention to the GSA transport committee's progress on this issue.

Improvements

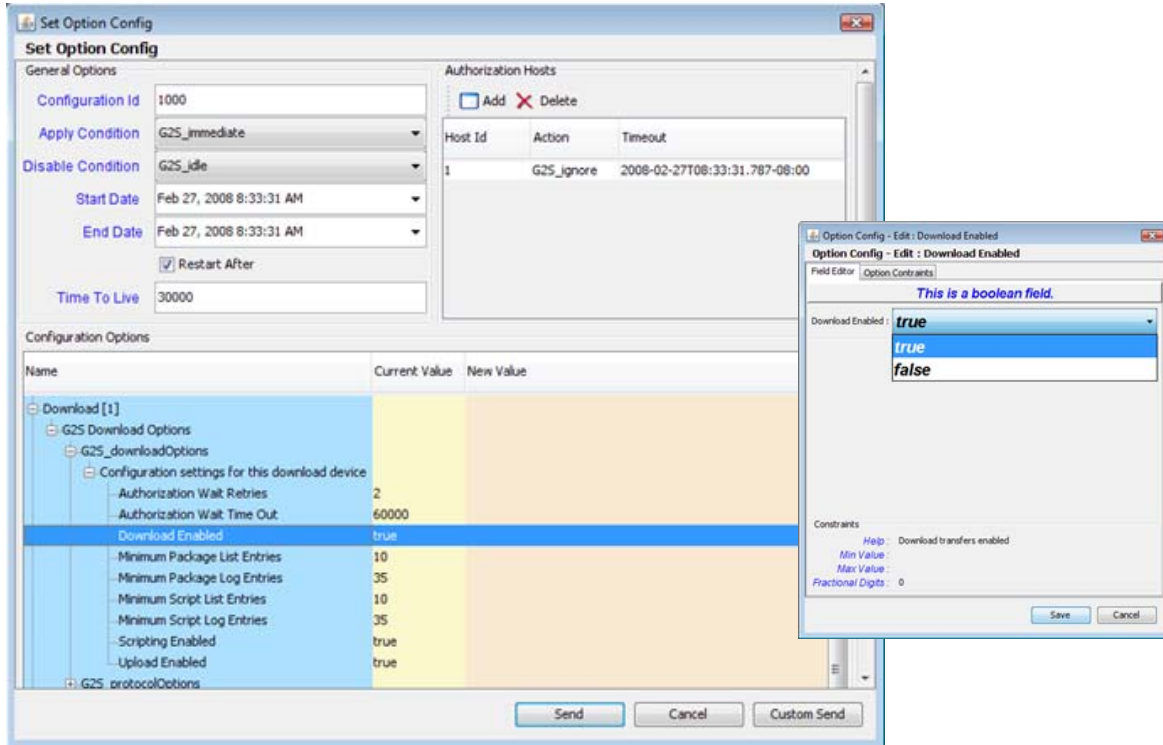
- The optionConfig class has been rewritten to give you a more robust testing environment. See [About Option Config](#) for more information.
- The Send Command to EGM object has been updated so that errors are displayed to the user in a pop-up box. Previously, errors that occurred when sending a command to the EGM were displayed only in the debug log.
- The support for the G2S download class has been overhauled in this version, with a wide variety of additional commands being available. See [About the Download Set Script](#) for more information.
- A voucher history display was created to let you see the validation IDs that have been issued to the EGM, as well as the state of each. Click on the validation Id column to sort the display on that column, and all updates for each validationId will be grouped together.

Corrections

- *includeConfigs* has been added to `communications.getDescriptor` command in the Send Command to EGM object.
- Incorrect default values for several of the attributes in the `gamePlay.gamePlayProfile` command have been corrected.
- RGS now properly handles omitted optional *dateTime* attributes (like *expireDateTime*) in commands in the voucher, bonus, and WAT classes.

About Option Config

Set Option Change allows you to send option configuration information to an EGM. This interface has a tree structure so you can easily drill-down to the options of interest.



The initial view is at the device level. Once you select the option you want to change, double-click the option to view the Option Config dialog box.

1. The Field Editor tab displays the current setting and, if the option is configurable, allows you to enter a new value.
2. The Option Constraints tab shows option details. When you change an option, RGS automatically verifies that it meets the option constraints. If a change falls outside a given set of constraints, an error message is displayed.
3. The Constraints section at the bottom of the dialog box displays a definition of the option as well as any information for setting the option.

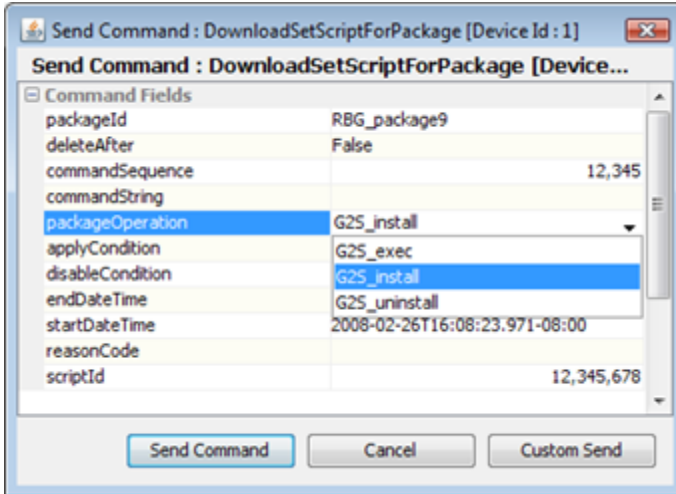
Note:

The `setOptionChange` control is populated by the values returned from the EGM through the `optionList` command. The `getOptionList` command is normally executed during start-up to request a copy of all `optionConfig` parameters for the EGM. If you skip that command, this control will be blank. If you just request the options for a couple of devices, that's all that will populate the control. However, we now store the options in the RGS database, so any changes you make are immediately reflected in the GUI.

About the Download Set Script

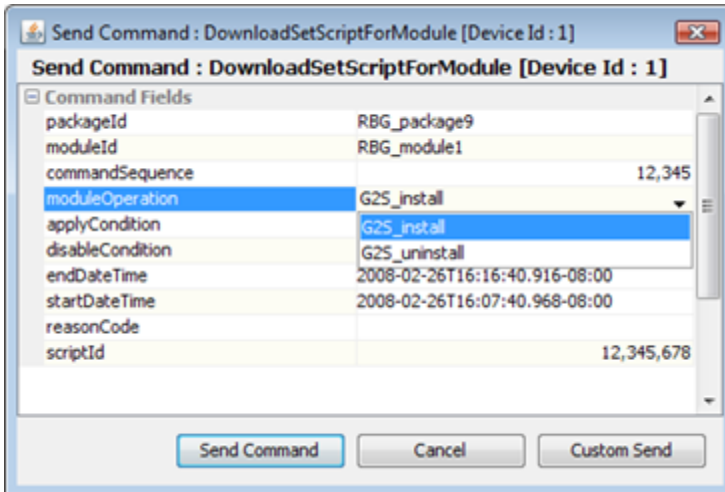
The `download.setScript` command allows you to install and uninstall packages and modules on the EGM. In RGS, two new commands have been added to **Send Command > G2S_download**:

Download – Set Script for Package (No Authorization)



From this screen, you can choose to install a package, uninstall a package and execute a command in the package.

Download – Set Script for Module (No Authorization)



From this screen, you can choose to install a single module in a package or uninstall a module.

Authorization Lists for `setScript` commands will be available in the next release of RGS.