radblue

RGS REST Interface Programming Guide

04 JUN 2013

Radical Blue Gaming, Inc.
85 Keystone Avenue Suite F
Reno, Nevada 89503

call us: +1.775.329.0990
visit us: www.radblue.com
drop us an email: sales@radblue.com

Need help?

At the RadBlue forum you can find the latest release information, report issues, get your questions answered, and submit suggestions for improving our products. Simply log on to: http://radblue.mywowbb.com

Find out more about the GSA protocols

If you want to find out more about the Gaming Standards Association and the work being done in the area of protocol standardization for the gaming industry, we encourage you to visit their website at www.gamingstandards.com.

Contents

RGS REST Interface Programming Guide **1**

About RadBlue **3**

Contents **5**

Chapter 1: Introducing RGS REST **9**

About the RGS REST Interface 9

A Note on Negative Testing 10

Method Overview 11

Notes on Using this Document 13

Additional Resources 13

Chapter 6: Getting Started **15**

Get Going with RGS REST 15

Sample Use Cases 15

Use Case 1 15

Use Case 2 16

RGS Scratch Pad 16

Connect to RGS REST 19

Chapter 2: Interacting with the EGM **21**

Method: connectedEgms 21

Request: connectedEgms 21

Response: egmList 21

Element 21

Elements 21

Example 22

Method: runScripts 23

Request: runScriptRequest 23
Elements 23
Example 23
Response: runScriptResponse 23
Elements 23
Example 24
Method: runScriptStatus 25
Request: runScriptStatusRequest 25
Element 25
Example 25
Response: RunScriptStatusResponse 25
Elements 25
Example 26
Method: scheduleMsx003 27
Request: EgmIdRequest 27
Element 27
Example 27
Response: Status 27
Elements 27
Example 28
Method: setActiveResponseManager 29
Request: FileNameRequest 29
Element 29
Example 29
Response: Status 29
Elements 29

Example 30
Method: setActiveStartupAlgorithm 31
Request: FileNameRequest 31
Element 31
Example 31
Response: Status 31
Elements 31
Example 32
Chapter 3: Accessing the Transcript 33
Method: fetchStatus 33
Request: TicketStatusRequest 33
Example 33
Response: CallResult 34
NameValueMap 34
Example 34
Method: fetchTranscript 35
Request: G2sTranscript2SnippetRequest 35
Example 36
Response: G2sTranscript2SnippetResponse 37
G2sTranscript2MessageType 38
Example 39
Method: insertTranscriptMarker 45
Request: TranscriptMarkerRequest 45
Example 45
Response: SimpleCallResult 45
Example 45

Chapter 4: Enumerations **47**

CommsState Enumeration 47

responseType 47

Status Enumeration 48

StatusErrorCode Enumeration 48

TicketCode 48

XML Data Types 49

Index **51**

About the RGS REST Interface

The REST (Representational State Transfer) interface allows you to control the RGS Tester Toolkit module remotely (The Send Command and other RGS standard functionality cannot be accessed through RGS REST). This interface, coupled with the ability of RGS to save testing artifacts as XML files in a known location, provides most automated test consoles with the information they need for stimulation of a G2S EGM and analysis of the G2S messages that are produced by the EGM.

You would use RGS REST if you wanted to remotely drive the testing of a G2S EGM or automate the host side as much as possible, using you own favorite programming language. You can confirm your test results by examining the G2S traffic between the EGM under test and RGS through transcript records.

> **Note:** When testing with RGS REST, you should refrain from using the RGS user interface in conjunction with the REST interface. The RGS user interface works on the same internal data structures as RGS Remote Control. It would, therefore, be possible to use the RGS user interface to modify RGS behavior in such a way as to compromise the transcript data and introduce false positive/negative results.

The REST interface can:

1. **Request a list of connected EGMs.**
   The response is a list of the EGMs that are currently connected to the RGS.
2. **Force an MSX003 for a connected EGM**.
   This option causes the selected EGM to restart its G2S communications, thereby using a newly selected start-up algorithm.
3. **Specify the current start-up algorithm**.
   The start-up algorithm is used by the RGS and selected from a list of start-up algorithms in the Tester Toolkit.
4. **Select a custom Response Manager script**.
   This option is used by the RGS to create non-standard responses to selected messages. The active Response Manager script is selected from any of the Response Manager scripts in the Tester Toolkit.
5. **Run a custom script**.
   Selected custom scripts can contain any supported G2S commands or script verbs, as long as the script does not require run-time device reconciliation. To run a script, specify the EGM to interact with and the script to run. A script ID is returned in the response.

6. **Check on the status of a running script**.
   Using the script ID that is returned when a custom script is launched, you can check the status of the script to determine when it will complete.
7. **Query the transcript**.
   Through the REST interface, you can fetch a set of transcript records, using the following filtering parameters:
   - Specify Maximum Records to return
   - Specify a specific EGM ID to match
   - Specify a specific Host ID to match
   - Specify a Date Range (start date/time, end date/time)
   - Specify a Message ID Range (start/end)
8. **Specify a starting and ending transcript marker**.
   You can set transcript markers through Custom Scripting. This filter can be useful if you create a transcript marker at the start of a test, and a second at the end of a test, and then request all transcript records between those two markers for the EGM under test.

Here is a the basic algorithm for using the RGS Remote Control:

1. Have the EGM under test connect to the RGS.
2. Connect to the RGS through the REST interface, and fetch the list of connected EGMs.
3. For the EGM of interest, retain the EGM Data Model information. You'll need the Device IDs from the descriptor list.
4. Perform the following loop as many times as needed:
   a. Invoke a Custom Script through the `runScript` method.
   b. Using `runScriptStatus`, poll the RGS waiting for the script to complete.
   c. Using `fetchTranscript`, retrieve the block of transcript records that pertains to this run.
   d. Perform analysis on the received transcript records. You'll need the Device IDs from step #3 to make sure you are looking at the right commands.

A Note on Negative Testing

If you are going to perform any negative testing, you will need to make sure that you account for this when analyzing the transcript. You can send bad XML using a Custom Script. That bad XML will arrive in the records sent back from `fetchTranscript`. Your parsing/analysis phase will need to take this into consideration.

Method Overview

The following table summarizes the function of each RGS REST method and its usage.

Method	Description	When to Use
connectedEgms	The `connectedEgms` method lists each currently connected EGM, its connection status and the G2S schema version it is running.	To do anything with the API you first need to know the following information: * The EGM IDs of the connected EGMs * The descriptors of the devices reported by the EGMs This API returns all of this information. From here you can target individual EGMs, both for scripts and for examining the transcript.
fetchStatus	The `fetchStatus` method checks the status of the previously submitted asynchronous request.	Use this method to find out the ultimate status of a previous asynchronous request.
fetchTranscript	The `fetchTranscript` method allows you to fetch some or all of the current transcript.	This is the key method for building your test system. You will need to fetch transcript records, parse them apart and then dig into the resulting data. There are a lot of different ways of doing this; which option you choose will be dependent on your environment and language of choice. The key choice to make when using this method is how best to search/select your transcript records. You can select by date, by EGM ID, by Message ID and quite a few additional criteria. Which criteria you use will be dependent on when you plan on pulling the transcript and how you will analyze the records.
insertTranscriptMarker	The `insertTranscriptMarker` method is used to insert a transcript marker into the RST transcript.	Use this method to insert a transcript marker. You can use transcript markers when fetching transcript records, to make it easier to find the records you want.
runScript	The `runScript` method lets you remotely run a custom script against one of the connected EGMs. Note that the custom script must exist within RGS (created using the Custom Scripting feature in the Tester Toolkit module).	This method allows you to execute any script currently registered in the RGS. Once you have determined which EGM you wish to test you can then invoke your script (or scripts). Here are some points to remember:

Method	Description	When to Use
		• The script runs asynchronously relative to your software. So you have to code in a periodic check (see runScriptStatus below) to check if the script is done. • Since there is no display there are some Custom Script verbs that don't make sense. • RGS Remote Control doesn't return any artifacts when the script finishes. It is assumed that you are using the interface because you want to generate Transcript traffic and then perform you own analysis.
runScriptStatus	The `runScriptStatus` method lets you receive information about the status of a custom script run against an EGM.	This method you to check for the status of a previously submitted custom script. You should invoke this method to find out when a submitted script has terminated. You can't run two scripts at the same time; they are executed sequentially. So this method allows you to figure out when one script ends so that you can analyze the results and then run a new script.
scheduleMsx003	The `scheduleMsx003` method schedules the sending of a `G2S_MSX003` (Communications Not Online) error that puts the communications channel for the specified EGM into a “lost” state.	This method is useful if you are testing Chapter 1 behavior related to the MSX003 error code. The specification defines very specific behavior required when the EGM receives the code.
setActiveResponseManager	The `setActiveResponseManager` method sets the selected response manager file as active in RGS. Note that the response manager file must exist within RGS (created using the Response Manager feature in the Tester Toolkit module).	Some classes of tests are best handled by setting a Response Manager config file and causing the RGS to return responses that contain errors. Using a registered Response Manager configuration you can perform negative testing of certain G2S requests that originate from the EGM under test.
setActiveStartupAlgorithm	The `setActiveStartupAlgorithm` method sets the selected start-up algorithm file as active in RGS. Note that the start-up algorithm must exist within RGS	Some classes of tests are best handled with a different Start-up script. Use this method to programmatically change the start-up algorithm, then use the

Method	Description	When to Use
	(created using the Start-up Algorithms feature in the Tester Toolkit module).	scheduleMsx003 method to force the EGM to restart its communications channel. A good use of a custom start-up script is to get the EGM into a particular state prior to the start of the test.

Notes on Using this Document

- Methods in this document are grouped by function:
  - **Interacting with the EGM** - Includes methods that let you send custom scripts, set custom start-up algorithms and set custom responses.
  - **Accessing the Transcript** - Includes methods that let you access Message Transcript data..
- All elements and attributes are *required* for each method.
- All amounts are in *millicents* (for example, $20.00 equals 2000000 millicents).
- The message flow of all *asynchronous* commands is:

RGS REST		EGM
Call method sent to EGM.	→	
	←	EGM returns a ticket number.
`fetchStatus` sent to EGM.	→	
	←	Result of call method sent to RGS REST.

Note that the call result for each asynchronous method appears at the end of the method description and *not* under the `fetchStatus` method.

Additional Resources

- RGS User Guide
- Tester Toolkit User Guide
- G2S Message Protocol
- W3C XML Schema Definition Language (XSD)1.1 Part 2: Datatypes

Get Going with RGS REST

Before you can begin to interface with RGS, you must write a program to remotely control RGS.

1. Review the sample use cases to understand the general message flow of RGS messages.
2. Use the RGS Scratch Pad, an RGS REST interface demonstration, to better understand what an RGS REST interface might look like.
3. Create a program, using the programming language of your choice, that connects to RGS through the following URL: **http://[localhost or IP address]:31501/remote**
4. Verify that the URL is working by connecting to a browser.
5. Use the method information provided in this document to create your program.

   Note: For a quick reference, the Method Overview provides a description of each available method and its use.

Sample Use Cases

Use Case 1

The following use case illustrates how the method calls can be combined for testing:

1. connectedEgms - Request connected EGMs.
2. setActiveResponseManager - Set the configuration file for custom responses to received commands.
3. insertTranscriptMarker - Insert transcript marker **Marker #1**.
4. runScript - Run a custom script against a specific EGM.
5. runScriptStatus - Get the status of the custom script.
6. insertTranscriptMarker - Insert transcript marker **Marker #2**.
7. fetchTranscript - Fetch all of the G2S transcript entries between **Marker #1** and **Marker #2**.

Use Case 2

The following use case illustrates how to test what happens when the EGM receives a `G2S_MSX003` error code.

1. scheduleMsx003 - Send an `G2S_MSX003(Communications Not Online)` error code to simulate lost communications with the EGM.

RGS Scratch Pad

The RGS Remote Control Scratch Pad is a demonstration of what an RGS REST interface might look like. It contains all of the supported methods. Once you've sent a method, the XML for both the request and response displays as well as a summary of the response.

Before you begin, verify that your RGS license supports Scratch Pad:

1. Launch RGS.
2. Go to: **Tools** > **Configure** > **License Manager**.
3. Click **View Features**.
4. Verify that the **remoteControl** field is **true**. If not, contact RadBlue.

Now, to access the RGS Scratch Pad:

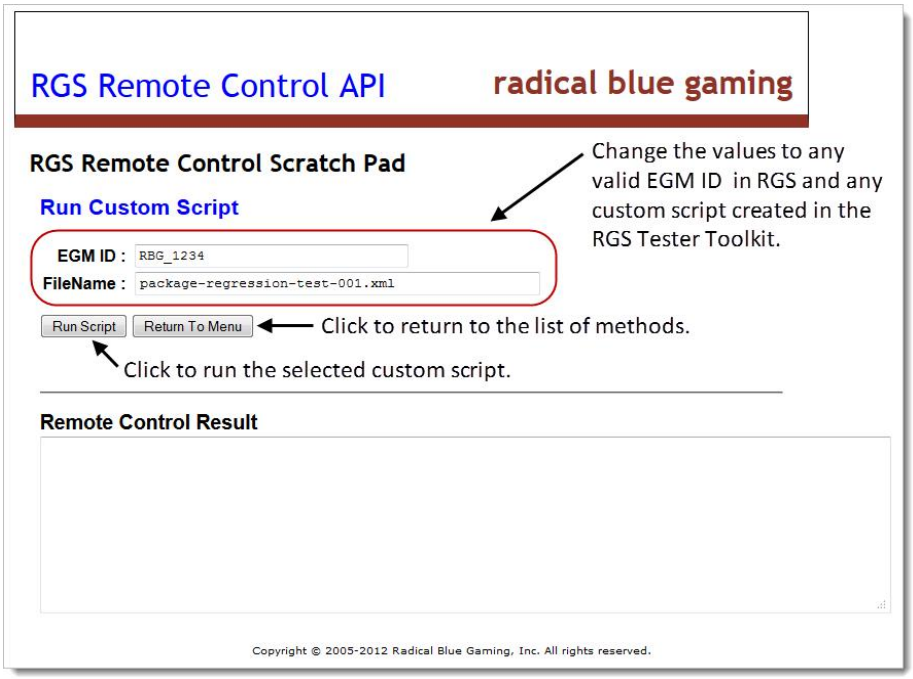
1. RGS starts when it receives a `commsOnline` command from the EGM (or RST, if you are using the RadBlue EGM simulator). Once communications have started, select the **Engine** layout, which contains the Debug Console.
2. Scroll up to the top of the log.
3. Scroll down until you see the following line:
   ```
   2013-05-07T15:48:07.161-07:00 [INFO] {svc-remote-control-clone} *
   ScratchPad URL:
   http://localhost:31501/RGS/remotecontrol/ScratchPad.html
   ```

   Note: To locate the information more quickly, click inside the **Debug Console** screen, and type **scratchpad.html**.
5. Highlight **http://localhost:31501/RGS/remotecontrol/ScratchPad.html**, and copy the selection.
6. Open either a **Firefox** or **Safari** browser.

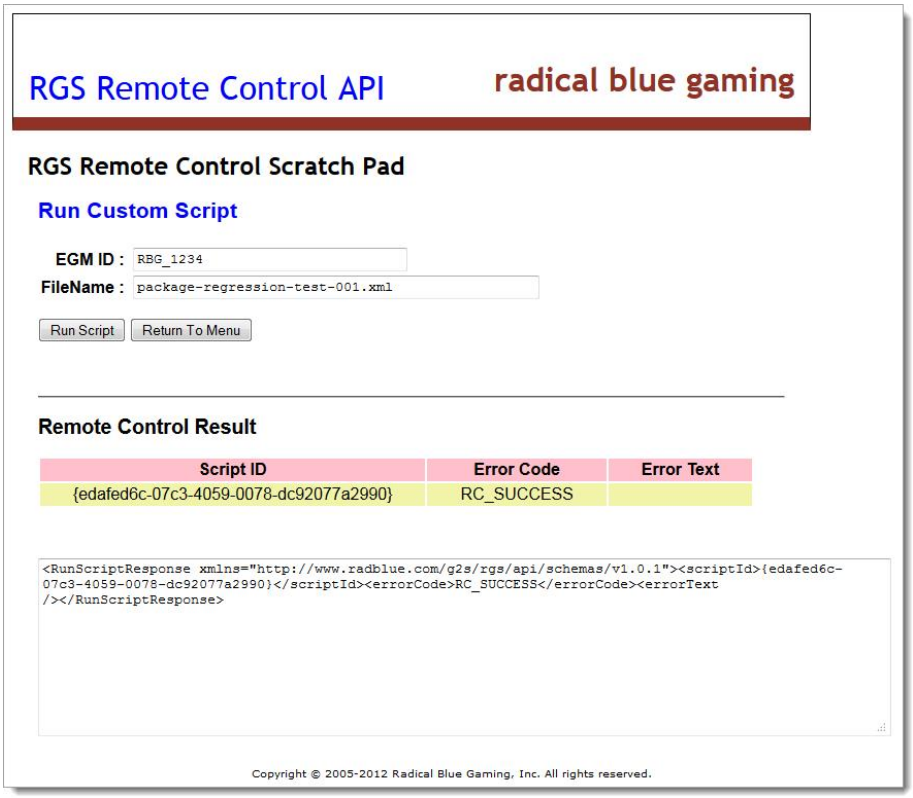
7. Paste the HTTP location into the address window, and press **Enter**.

Then, start sending methods:

1. Click any method to view its send options. In this example, we chose **Run Custom Script**.



2. Configure method attributes, and click **Run Script**.



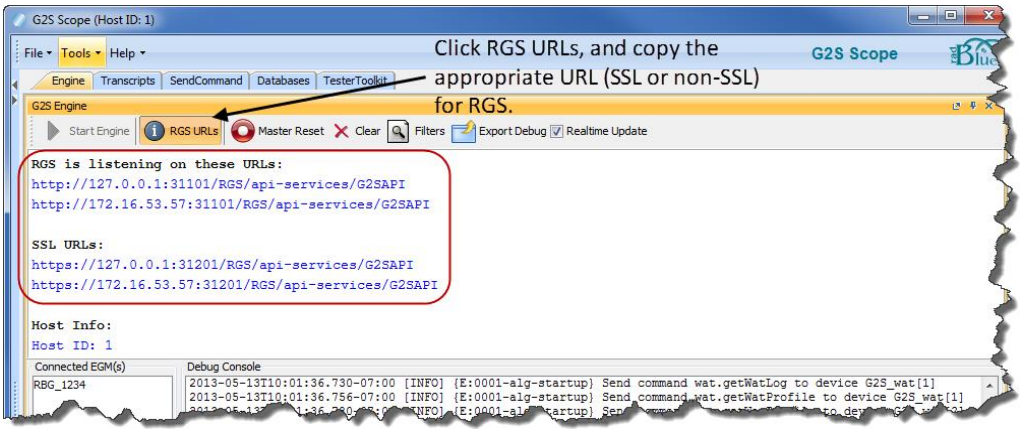
Once the message is sent, the **Remote Control Result** section displays a summary of the results along with the XML for the request and response messages.

3. Click **Return To Menu** to go back to the list of methods.

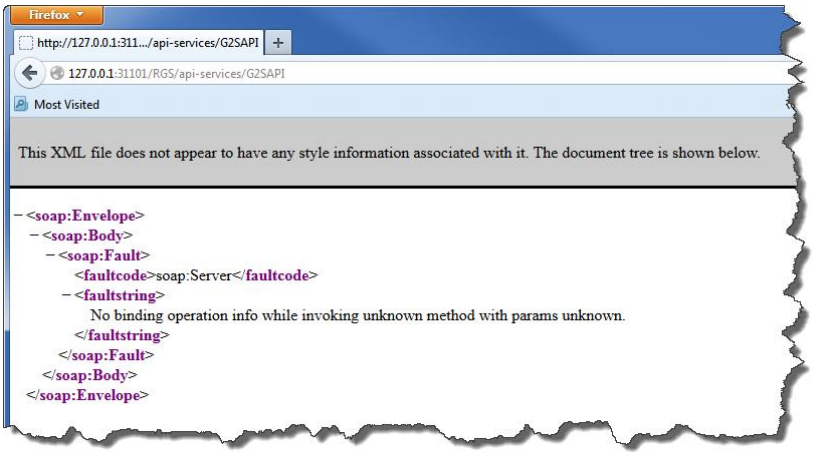
Connect to RGS REST

This procedure tests that your browser is communicating with RGS.

1. Launch **RGS**.
2. Click the **Engine** layout tab, and then click **RGS URLs**.



3. Highlight the appropriate URL (either SSL or non-SSL), and type **CTRL+c** to copy.
4. Open either a **Firefox** or **Safari** browser.
5. Paste the URL into the address bar, and press **Enter**.



6. If you see the above information in your browser, your browser has successfully connected to RGS. If not, your browser is not connecting with RGS. Contact support@radblue.com for troubleshooting assistance.

Method: connectedEgms

The `connectedEgms` method lists each currently connected EGM, its connection status and the G2S schema version it is running.

HTTP Type: GET

Request: connectedEgms

There are no elements for this method.

Response: egmList

Element

Element	Restrictions	Description
egm	type: complex minOccurs: "0" maxOccurs: "unbounded"	List of connected EGMs.

Elements

Element	Restrictions	Description
egmId	Type: string pattern:[A-Z, 0-9]{3}_[-~]{1,28}	Unique EGM identifier.
commsState	Type: string Enumeration: CommsState	Communications status of EGM.
connectionDate	Type: dateTime	Date and time
schemaVersion	Type: string	G2S schema used by the EGM.

Example

```
<EgmList xmlns="http://www.radblue.com/g2s/remote/api/schemas/v1.0.1"
         xmlns:ns2="http://www.radblue.com/g2s/transcript2/api/schemas/v1.0.1">
    <egm>
         <egmId>RBG_1234</egmId>
         <commsState>G2S_onLine</commsState>
         <connectionDate>2011-09-14T09:34:47.678-07:00</connectionDate>
         <schemaVersion>1.1.0</schemaVersion>
    </egm>
</EgmList>
```

Method: runScripts

The `runScripts` method lets you remotely run a custom script against one of the connected EGMs.

Note that the custom script must exist within RGS (created using the Custom Scripting feature in the Tester Toolkit module).

Any script that can be run from within RGS can be run using this method. However, the `prompt` verb cannot be used. If the `prompt` verb is included in a custom script, it will be skipped over when the script is run.

HTTP Type: Post

Request: runScriptRequest

Elements

Element	Restrictions	Description
egmId	type: string pattern:[A-Z, 0-9]{3}_[-~]{1,28}	Unique EGM identifier.
fileName	type: string	File name, including extension, for the file that is to be run.

Example

```
<RunScriptRequest xmlns="http://www.radblue.com/g2s/remote/api/schemas/v1.0.1">
    <egmId>RBG_1234</egmId>
    <fileName>BonusScript.xml</fileName>
</RunScriptRequest>
```

Response: runScriptResponse

Elements

Element	Restrictions	Description
errorCode	type: string minOccurs: 0 maxOccurs: 1	Error code.
errorText	type: string	Error description.
scriptId	type: string	Script identifier.

Example

```
<RunScriptResponse xmlns="http://www.radblue.com/g2s/remote/api/schemas/v1.0.1"
                   xmlns:ns2="http://www.radblue.com/g2s/transcript2/api/schemas/v1.0.1">
         <scriptId>{eadf9fc4-17e5-4772-008f-45a15ec65bd9}</scriptId>
         <errorCode>RC_SUCCESS</errorCode>
         <errorText>Success<errorText/>
</RunScriptResponse>
```

Method: runScriptStatus

The `runScriptStatus` method lets you receive information about the status of a custom script run against an EGM.

HTTP Type: Post

Request: runScriptStatusRequest

Element

Element	Restrictions	Description
scriptId	type: string	Script identifier.

Example

```
<RunScriptStatusRequest xmlns="http://www.radblue.com/g2s/remote/api/schemas/v1.0.1">
    <scriptId>{eadf9fc4-17e5-4772-008f-45a15ec65bd9}</scriptId>
</RunScriptStatusRequest>
```

Response: RunScriptStatusResponse

Elements

Element	Restrictions	Description
percent	type: string	If the script is currently running, indicates the percentage of completion.
scriptId	type: string	Script identifier.
status	type: string enumeration: status	Indicates the current status of the specified script.
text	type: string	Descriptive information.

Example


```
<RunScriptStatusResponse xmlns="http://www.radblue.com/g2s/remote/api/schemas/v1.0.1"
                         xmlns:ns2="http://www.radblue.com/g2s/transcript2/api/schemas/v1.0.
1">
    <scriptId>{eadf9fc4-17e5-4772-008f-45a15ec65bd9}</scriptId>
    <status>RC_success</status>
    <percent>100</percent>
    <text>Executing Verb: Transcript Marker: Script End: BonusScript</text>
</RunScriptStatusResponse>
```

Method: scheduleMsx003

The `scheduleMsx003` method schedules the sending of a `G2S_MSX003` (Communications Not Online) error that puts the communications channel for the specified EGM into a "lost" state. The EGM will then re-initialize its communications queues and restart communications by sending a `commsOnLine` command to RGS. This is the best way to get an EGM to reconnect and, therefore, execute a newly assigned start-up algorithm.

HTTP Type: POST

Request: EgmIdRequest

Element

Element	Restrictions	Description
egmId	type: string pattern:[A-Z, 0-9]{3}_[-~]{1,28}	Unique EGM identifier.

Example

```
<EgmIdRequest xmlns="http://www.radblue.com/g2s/remote/api/schemas/v1.0.1">
    <egmId>RBG_1234</egmId>
</EgmIdRequest>
```

Response: Status

Elements

Element	Restrictions	Description
errorCode	type: string enumeration: StatusErrorCode	Code for error type.
errorText	type: string	Error description.

Example

```
<Status xmlns="http://www.radblue.com/g2s/remote/api/schemas/v1.0.1"
        xmlns:ns2="http://www.radblue.com/g2s/transcript2/api/schemas/v1.0.1">
    <errorCode>RC_SUCCESS</errorCode>
    <errorText>Success</errorText>
</Status>
```

Method: setActiveResponseManager

The `setActiveResponseManager` method sets the selected response manager file as active in RGS.

Note that the response manager file must exist within RGS (created using the Response Manager feature in the Tester Toolkit module).

HTTP Type: POST

Request: FileNameRequest

Element

Element	Restrictions	Description
fileName	type: string	File name, including extension, for the response manager file that is to be set as active.

Example

```
<FileNameRequest xmlns="http://www.radblue.com/g2s/remote/api/schemas/v1.0.1">
    <fileName>response-manager-config-gsa-voucher.xml</fileName>
</FileNameRequest>
```

Response: Status

Elements

Element	Restrictions	Description
errorCode	type: string enumeration: StatusErrorCode	Code for error type.
errorText	type: string	Error description.

Example

```
<Status xmlns="http://www.radblue.com/g2s/remote/api/schemas/v1.0.1"
        xmlns:ns2="http://www.radblue.com/g2s/transcript2/api/schemas/v1.0.1">
    <errorCode>RC_SUCCESS</errorCode>
    <errorText>Success</errorText>
</Status>
```

Method: setActiveStartupAlgorithm

The `setActiveStartupAlgorithm` method sets the selected start-up algorithm file as active in RGS.

Note that the start-up algorithm must exist within RGS (created using the Algorithms feature in the Tester Toolkit module).

HTTP Type: POST

Request: FileNameRequest

Element

Element	Restrictions	Description
fileName	type: string	File name, including extension, for the start-up algorithm file that is to be set as active.

Example

```
<FileNameRequest xmlns="http://www.radblue.com/g2s/remote/api/schemas/v1.0.1">
    <fileName>radblue-smarthost-def-gsa-002.xml</fileName>
</FileNameRequest>
```

Response: Status

Elements

Element	Restrictions	Description
errorCode	type: string enumeration: StatusErrorCode	Code for error type.
errorText	type: string	Error description.

Example

```
<Status xmlns="http://www.radblue.com/g2s/remote/api/schemas/v1.0.1"
xmlns:ns2="http://www.radblue.com/g2s/transcript2/api/schemas/v1.0.1">
    <errorCode>RC_SUCCESS</errorCode>
    <errorText>Success</errorText>
</Status>
```

Method: fetchStatus

This method checks the status of a previously submitted asynchronous request. The `CallResult` object contains an enumeration that indicates whether the request is still in progress or if it has completed (and what the outcome was). Use this method to find out the ultimate status of a previous asynchronous request.

Note that all elements and attributes are *required* for each method.

Details of the responses for asynchronous methods can be found with each method.

HTTP Type: POST

Call Type: Synchronous

Request: TicketStatusRequest

Element	Restrictions	Description
ticket	type: string minLength: 38 maxLength: 38	Tracking number for the submitted asynchronous operation. The ticket is passed to the `fetchStatus` method. RST then indicates whether the asynchronous method call has completed or is still in progress.

Example

```
<TicketStatusRequestxmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
<ticket>12345678912345678912345678912345678 9</ticket></TicketStatusRequest>
```

Response: CallResult

Element	Restrictions	Description
attributes	type: NameValueMap minOccurs: 0 maxOccurs: unbounded	Name-value pair that is specific to the requested information.
code	type: TicketCode minOccurs: 1 maxOccurs: 1	Outcome of the asynchronous API call.
message	type: string minOccurs: 1 maxOccurs: 1	Error or message associated with the asynchronous API call.
responseType	type: responseType minOccurs: 1 maxOccurs: 1	Indicates the type of information sent in the `attributes` element.
ticket	type: string minLength: 39 maxLength: 39	Tracking number for the submitted asynchronous operation. The ticket is passed to the `fetchStatus` method. RST then indicates whether the asynchronous method call has completed or is still in progress.

NameValueMap

Element	Restrictions	Description
name	type: string minOccurs: 1 maxOccurs: 1	Name associated with requested information.
value	type: string minOccurs: 1 maxOccurs: 1	Value associated with requested information.

Example

```
<CallResult xmlns:ns3="http://www.radblue.com/g2s/transcript2/api/schemas/v1.0.0"
xmlns:ns2="http://www.radblue.com/g2s/common/api/schemas/v1.0.0"
xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1"><ticket>123456789123456789123456789123456789</ticket><responseType>Unknown</responseType><code>RBG_BAD_ARGUMENTS</code><message>Never heard of ticket 123456789123456789123456789123456789</message></CallResult>
```

Method: fetchTranscript

The `fetchTranscript` method lets you fetch a set of transcript records, using the following filtering parameters:

- Specify Maximum Records to return
- Specify a specific EGM ID to match
- Specify a specific Host ID to match
- Specify a Date Range (start date/time, end date/time)
- Specify a Message ID Range (start/end)
- Specify a starting and ending Transcript Marker.

Note that all elements and attributes are *required* for each method.

HTTP Type: POST

Call Type: Synchronous

Request: G2sTranscript2SnippetRequest

Element	Restrictions	Description
commandClass	type: string maxLength:32 minLength: 0	Command class of requested messages. **Example**: "communications"
deviceId	type: int	EGM device identifier of requested messages. "-1" requests all devices for the specified command class. **Example**: "2"
egmId	type: string	Unique EGM identifier.
endDate	type: dateTime	End date and time of requested transcript records. **Example**: "2038-01-01T00:00:00.000-06:00"
endMessageId	type: long	Identifier of the last requested transcript record.
endTranscriptMarker	type: string	Name of last requested transcript record.
hostId	type: long maxOccurs: 12	Host ID associated with requested transcript records.

Element	Restrictions	Description
includeG2sAcks	type: boolean	"True" includes `g2sAck` messages in the response. "False" excludes all occurrences of this message from the response.
maxRecords	type: long restriction: zero (0) or greater	Maximum number of matching records to send in response. **Example**: "100"
startDate	type: dateTime	Starting date and time for requested transcript records. **Example**: "2038-01-01T00:00:00.000-06:00"
startMessageId	type: long	Identifier of first requested transcript record.
startTranscriptMarker	type: string	Name of first requested transcript record.

Example

```
<G2STranscript2SnippetRequest
xmlns="http://www.radblue.com/g2s/transcript2/api/schemas/v1.0.1">
    <egmId>RBG_1234<egmId/>
    <maxRecords>10</maxRecords>
    <startDate>1969-12-31T18:00:00.000-06:00</startDate>
    <endDate>2038-01-01T00:00:00.000-06:00</endDate>
    <startMessageId>0</startMessageId>
    <endMessageId>0</endMessageId>
    <startTranscriptMarker/>
    <endTranscriptMarker/>
</G2STranscript2SnippetRequest>
```

Response: G2sTranscript2SnippetResponse

Note: Query values are repeated at the start of the response.

Element	Restrictions	Description
commandClass	type: string maxLength:32 minLength: 0	Class of requested messages. **Example**: "communications"
dateGenerated	type: dateTime	Date and time that the requested transcript was generated.
deviceId	type: int	EGM device identifier of requested messages. "-1" requests all devices for the specified command class. **Example**: "2"
egmId	type: string	Unique EGM identifier.
endDate	type: dateTime	End date and time of requested transcript records. **Example**: "2038-01-01T00:00:00.000-06:00"
endMessageId	type: long	Identifier of the last requested transcript record.
endTranscriptMarker	type: string	Name of last requested transcript record.
hostId	type: long maxOccurs: 12	Host ID associated with requested transcript records.
includeG2sAcks	type: boolean	"True" includes `g2sAck` messages in the response. "False" excludes all occurrences of this message from the response.
maxRecords	type: long	Maximum transcript records included in response.
message	type: G2sTranscript2MessageType	Message transcript details.
startDate	type: dateTime	Starting date and time for requested transcript records, in the format: "YYYY-MM-DDTHH:MM:SS:mmm-T" **Example**: "2038-01-01T00:00:00.000-06:00"
startMessageId	type: long	Identifier of first requested transcript record.
startTranscriptMarker	type: string	Name of first requested transcript record.
transcriptSize	type: long	Size of requested transcript.

G2sTranscript2MessageType

Element	Restrictions	Description
commandClass	type: string	G2S class for the specified command.
commandId	type: long	Command identifier associated with message.
commandName	type: string	Actual G2S command within the message. If more than one command is sent in a message, each command appears in its own message.
comment	type: string	Information entered by the user about a specific message. This field is *not* part of the actual message. Comments exist *only* in the tool in which they are entered.
content	type: string	Contains the XML message.
dateReceived	type: dateTime	Date and time message was received by the tool.
direction	type: string	Indicates whether message was sent or received by the tool. **Example**: "inbound"
end	type: string	Identifies the sender ("host" or "egm") of the message. **Example**: "host"
eventCode	type: string	Event code associated with message.
eventText	type: string	Descriptive text associated with specified event code.
fromLocation	type: string	Identifier of entity (for example, EGM or host) that sent the message.
messageId	type: long	Unique message identifier.
sessionId	type: long	Session ID associated with message.
sessionType	type: string default: "G2S_request"	Indicates how the message should be processed: as a request, response or notification. **Example**: G2S_request"
summary	type: string	Descriptive text for the command identifier.
timeToLive	type: long default: "30000"	Time, in milliseconds, to wait until message times out. **Example**: "30000" (equaling $0.30)
toLocation	type: string	Identifier of the intended target of the message.
transactionId	type: long	Unique transaction identifier.

Example


```
<G2STranscript2SnippetResponse xmlns="http://www.radblue.com/g2s/remote/api/schemas/v1.0.1"

xmlns:ns2="http://www.radblue.com/g2s/transcript2/api/schemas/v1.0.1">
    <ns2:dateGenerated>2011-09-14T09:42:24.030-07:00</ns2:dateGenerated>
    <ns2:maxRecords>10</ns2:maxRecords>
    <ns2:egmId/>
    <ns2:transcriptSize>10</ns2:transcriptSize>
    <ns2:message>
         <ns2:commandClass>undefined</ns2:commandClass>
         <ns2:commandId>0</ns2:commandId>
         <ns2:commandName>G2SACK</ns2:commandName>
         <ns2:comment/>
         <ns2:content>&lt;?xml version="1.0" encoding="UTF-8" standalone="yes"?&gt;
&lt;g2s:g2sMessage xmlns:g2s="http://www.gamingstandards.com/g2s/schemas/v1.0.3"&gt;
&lt;g2s:g2sAck g2s:dateTimeSent="2011-09-14T09:41:46.096-07:00" g2s:egmId="RBG_1234"
           g2s:hostId="1"/&gt;
&lt;/g2s:g2sMessage&gt;</ns2:content>
         <ns2:dateReceived>2011-09-14T09:41:46.097-07:00</ns2:dateReceived>
         <ns2:errorCode>G2S_none</ns2:errorCode>
         <ns2:errorText>G2S_none</ns2:errorText>
         <ns2:eventCode>G2S_none</ns2:eventCode>
         <ns2:eventText>G2S_none</ns2:eventText>
         <ns2:fromLocation>Host ID 1</ns2:fromLocation>
         <ns2:direction>Outbound</ns2:direction>
         <ns2:end>Host</ns2:end>
         <ns2:messageId>874</ns2:messageId>
         <ns2:sessionId>0</ns2:sessionId>
         <ns2:sessionType>N/A</ns2:sessionType>
         <ns2:summary>G2SACK</ns2:summary>
         <ns2:timeToLive>-1</ns2:timeToLive>
         <ns2:toLocation>RBG_1234</ns2:toLocation>
         <ns2:transactionId>-1</ns2:transactionId>
    </ns2:message>
    <ns2:message>
         <ns2:commandClass>bonus</ns2:commandClass>
         <ns2:commandId>217</ns2:commandId>
         <ns2:commandName>bonus.setBonusState</ns2:commandName>
         <ns2:comment>Custom Script: BonusScript</ns2:comment>
         <ns2:content>&lt;?xml version="1.0" encoding="UTF-8"
         standalone="yes"?&gt;&lt;g2s:g2sMessage
         xmlns:g2s="http://www.gamingstandards.com/g2s/schemas/v1.0.3"&gt;&lt;g2s:g2sBody
         g2s:dateTimeSent="2011-09-14T09:41:46.112-07:00" g2s:egmId="RBG_1234"
         g2s:hostId="1"&gt;&lt;g2s:bonus g2s:commandId="217" g2s:dateTime="2011-09-
         14T09:41:46.108-07:00" g2s:deviceId="1" g2s:errorCode="G2S_none" g2s:errorText=""
         g2s:sessionId="200140" g2s:sessionMore="false" g2s:sessionRetry="false"
         g2s:sessionType="G2S_request" g2s:timeToLive="30000"&gt;&lt;g2s:setBonusState
         g2s:disableText=""
         g2s:enable="true"/&gt;&lt;/g2s:bonus&gt;&lt;/g2s:g2sBody&gt;&lt;/g2s:g2sMessage&gt;
         </ns2:content>
         <ns2:dateReceived>2011-09-14T09:41:46.112-07:00</ns2:dateReceived>
         <ns2:errorCode>G2S_none</ns2:errorCode>
```

```
        <ns2:errorText>G2S_none</ns2:errorText>
        <ns2:eventCode>G2S_none</ns2:eventCode>
        <ns2:eventText>G2S_none</ns2:eventText>
        <ns2:fromLocation>Host ID 1</ns2:fromLocation>
        <ns2:direction>Outbound</ns2:direction>
        <ns2:end>Host</ns2:end>
        <ns2:messageId>875</ns2:messageId>
        <ns2:sessionId>200140</ns2:sessionId>
        <ns2:sessionType>G2S_request</ns2:sessionType>
        <ns2:summary>bonus.setBonusState</ns2:summary>
        <ns2:timeToLive>30000</ns2:timeToLive>
        <ns2:toLocation>RBG_1234</ns2:toLocation>
        <ns2:transactionId>-1</ns2:transactionId>
    </ns2:message>
    <ns2:message>
        <ns2:commandClass>undefined</ns2:commandClass>
        <ns2:commandId>0</ns2:commandId>
        <ns2:commandName>G2SACK</ns2:commandName>
        <ns2:comment/>
        <ns2:content>&lt;?xml version="1.0" encoding="UTF-8" standalone="yes"?&gt;
        &lt;g2s:g2sMessage
        xmlns:g2s="http://www.gamingstandards.com/g2s/schemas/v1.0.3"&gt;
    &lt;g2s:g2sAck g2s:dateTimeSent="2011-09-14T09:41:46.120-07:00" g2s:egmId="RBG_1234"
          g2s:hostId="1"/&gt;
&lt;/g2s:g2sMessage&gt;</ns2:content>
        <ns2:dateReceived>2011-09-14T09:41:46.126-07:00</ns2:dateReceived>
        <ns2:errorCode>G2S_none</ns2:errorCode>
        <ns2:errorText>G2S_none</ns2:errorText>
        <ns2:eventCode>G2S_none</ns2:eventCode>
        <ns2:eventText>G2S_none</ns2:eventText>
        <ns2:fromLocation>RBG_1234</ns2:fromLocation>
        <ns2:direction>Inbound</ns2:direction>
        <ns2:end>Client</ns2:end>
        <ns2:messageId>876</ns2:messageId>
        <ns2:sessionId>0</ns2:sessionId>
        <ns2:sessionType>N/A</ns2:sessionType>
        <ns2:summary>G2SACK</ns2:summary>
        <ns2:timeToLive>-1</ns2:timeToLive>
        <ns2:toLocation>Host ID 1</ns2:toLocation>
        <ns2:transactionId>-1</ns2:transactionId>
    </ns2:message>
    <ns2:message>
        <ns2:commandClass>bonus</ns2:commandClass>
        <ns2:commandId>120054</ns2:commandId>
        <ns2:commandName>bonus.bonusStatus</ns2:commandName>
        <ns2:comment/>
        <ns2:content>&lt;?xml version="1.0" encoding="UTF-8"
        standalone="yes"?&gt;&lt;g2s:g2sMessage
        xmlns:g2s="http://www.gamingstandards.com/g2s/schemas/v1.0.3"&gt;&lt;g2s:g2sBody
        g2s:dateTimeSent="2011-09-14T09:41:46.125-07:00" g2s:egmId="RBG_1234"
        g2s:hostId="1"&gt;&lt;g2s:bonus g2s:commandId="120054" g2s:dateTime="2011-09-
        14T09:41:46.122-07:00" g2s:deviceId="1" g2s:errorCode="G2S_none" g2s:errorText=""
        g2s:sessionId="200140" g2s:sessionMore="false" g2s:sessionRetry="false"
        g2s:sessionType="G2S_response" g2s:timeToLive="0"&gt;&lt;g2s:bonusStatus
        g2s:bonusActive="false" g2s:configurationId="0" g2s:delayGames="0"
        g2s:delayLater="false" g2s:delayTime="0" g2s:delayValue="0" g2s:egmEnabled="true"
```

```
        g2s:hostActive="true" g2s:hostEnabled="true"
        g2s:hostLocked="false"/&gt;&lt;/g2s:bonus&gt;&lt;/g2s:g2sBody&gt;&lt;
          /g2s:g2sMessage&gt;</ns2:content>
        <ns2:dateReceived>2011-09-14T09:41:46.133-07:00</ns2:dateReceived>
        <ns2:errorCode>G2S_none</ns2:errorCode>
        <ns2:errorText>G2S_none</ns2:errorText>
        <ns2:eventCode>G2S_none</ns2:eventCode>
        <ns2:eventText>G2S_none</ns2:eventText>
        <ns2:fromLocation>RBG_1234</ns2:fromLocation>
        <ns2:direction>Inbound</ns2:direction>
        <ns2:end>Client</ns2:end>
        <ns2:messageId>877</ns2:messageId>
        <ns2:sessionId>200140</ns2:sessionId>
        <ns2:sessionType>G2S_response</ns2:sessionType>
        <ns2:summary>bonus.bonusStatus</ns2:summary>
        <ns2:timeToLive>0</ns2:timeToLive>
        <ns2:toLocation>Host ID 1</ns2:toLocation>
        <ns2:transactionId>-1</ns2:transactionId>
    </ns2:message>
    <ns2:message>
        <ns2:commandClass>undefined</ns2:commandClass>
        <ns2:commandId>0</ns2:commandId>
        <ns2:commandName>G2SACK</ns2:commandName>
        <ns2:comment/>
        <ns2:content>&lt;?xml version="1.0" encoding="UTF-8" standalone="yes"?&gt;
&lt;g2s:g2sMessage xmlns:g2s="http://www.gamingstandards.com/g2s/schemas/v1.0.3"&gt;
&lt;g2s:g2sAck g2s:dateTimeSent="2011-09-14T09:41:46.144-07:00" g2s:egmId="RBG_1234"
          g2s:hostId="1"/&gt;
&lt;/g2s:g2sMessage&gt;</ns2:content>
          <ns2:dateReceived>2011-09-14T09:41:46.144-07:00</ns2:dateReceived>
          <ns2:errorCode>G2S_none</ns2:errorCode>
          <ns2:errorText>G2S_none</ns2:errorText>
          <ns2:eventCode>G2S_none</ns2:eventCode>
          <ns2:eventText>G2S_none</ns2:eventText>
          <ns2:fromLocation>Host ID 1</ns2:fromLocation>
          <ns2:direction>Outbound</ns2:direction>
          <ns2:end>Host</ns2:end>
          <ns2:messageId>878</ns2:messageId>
          <ns2:sessionId>0</ns2:sessionId>
          <ns2:sessionType>N/A</ns2:sessionType>
          <ns2:summary>G2SACK</ns2:summary>
          <ns2:timeToLive>-1</ns2:timeToLive>
          <ns2:toLocation>RBG_1234</ns2:toLocation>
          <ns2:transactionId>-1</ns2:transactionId>
    </ns2:message>
    <ns2:message>
        <ns2:commandClass>communications</ns2:commandClass>
        <ns2:commandId>0</ns2:commandId>
        <ns2:commandName>communications.transcriptMarker</ns2:commandName>
        <ns2:comment/>
        <ns2:content>&lt;?xml version="1.0" encoding="UTF-8"
        standalone="yes"?&gt;&lt;g2s:g2sMessage
        xmlns:g2s="http://www.gamingstandards.com/g2s/schemas/v1.0.3"&gt;&lt;g2s:g2sBody
        g2s:dateTimeSent="2011-09-14T09:41:46.156-07:00" g2s:egmId="RBG_1234"
        g2s:hostId="1"&gt;&lt;g2s:communications g2s:commandId="1" g2s:dateTime="2011-09-
        14T09:41:46.153-07:00" g2s:deviceId="1" g2s:errorCode="G2S_none" g2s:errorText=""
```

```
        g2s:sessionId="0" g2s:sessionMore="false" g2s:sessionRetry="false"
        g2s:sessionType="G2S_notification" g2s:timeToLive="0"&gt;&lt;rbg:transcriptMarker
        xmlns:rbg="http://www.radblue.com/gsa/g2s/extensions/1.0.0" rbg:marker-name="Script
        End: BonusScript"/&gt;&lt;/g2s:communications&gt;&lt;/g2s:g2sBody&gt;&lt;/g2s:
        g2sMessage&gt;</ns2:content>
        <ns2:dateReceived>2011-09-14T09:41:46.159-07:00</ns2:dateReceived>
        <ns2:errorCode>G2S_none</ns2:errorCode>
        <ns2:errorText>G2S_none</ns2:errorText>
        <ns2:eventCode>G2S_none</ns2:eventCode>
        <ns2:eventText>G2S_none</ns2:eventText>
        <ns2:fromLocation>Host ID 1</ns2:fromLocation>
        <ns2:direction>Outbound</ns2:direction>
        <ns2:end>Host</ns2:end>
        <ns2:messageId>879</ns2:messageId>
        <ns2:sessionId>0</ns2:sessionId>
        <ns2:sessionType>N/A</ns2:sessionType>
        <ns2:summary>TM: Script End: BonusScript</ns2:summary>
        <ns2:timeToLive>0</ns2:timeToLive>
        <ns2:toLocation>RBG_1234</ns2:toLocation>
        <ns2:transactionId>-1</ns2:transactionId>
    </ns2:message>
    <ns2:message>
        <ns2:commandClass>communications</ns2:commandClass>
        <ns2:commandId>120055</ns2:commandId>
        <ns2:commandName>communications.keepAlive</ns2:commandName>
        <ns2:comment/>
        <ns2:content>&lt;?xml version="1.0" encoding="UTF-8"
        standalone="yes"?&gt;&lt;g2s:g2sMessage
        xmlns:g2s="http://www.gamingstandards.com/g2s/schemas/v1.0.3"&gt;&lt;g2s:g2sBody
        g2s:dateTimeSent="2011-09-14T09:42:17.055-07:00" g2s:egmId="RBG_1234"
        g2s:hostId="1"&gt;&lt;g2s:communications g2s:commandId="120055" g2s:dateTime="2011-
        09-14T09:42:17.049-07:00" g2s:deviceId="1" g2s:errorCode="G2S_none"
        g2s:errorText="" g2s:sessionId="3000192" g2s:sessionMore="false"
        g2s:sessionRetry="false" g2s:sessionType="G2S_request"
        g2s:timeToLive="30000"&gt;&lt;g2s:keepAlive/&gt;&lt;/g2s:communications&gt;&lt;
          /g2s:g2sBody&gt;&lt;/g2s:g2sMessage&gt;</ns2:content>
        <ns2:dateReceived>2011-09-14T09:42:17.068-07:00</ns2:dateReceived>
        <ns2:errorCode>G2S_none</ns2:errorCode>
        <ns2:errorText>G2S_none</ns2:errorText>
        <ns2:eventCode>G2S_none</ns2:eventCode>
        <ns2:eventText>G2S_none</ns2:eventText>
        <ns2:fromLocation>RBG_1234</ns2:fromLocation>
        <ns2:direction>Inbound</ns2:direction>
        <ns2:end>Client</ns2:end>
        <ns2:messageId>881</ns2:messageId>
        <ns2:sessionId>3000192</ns2:sessionId>
        <ns2:sessionType>G2S_request</ns2:sessionType>
        <ns2:summary>communications.keepAlive</ns2:summary>
        <ns2:timeToLive>30000</ns2:timeToLive>
        <ns2:toLocation>Host ID 1</ns2:toLocation>
        <ns2:transactionId>-1</ns2:transactionId>
    </ns2:message>
    <ns2:message>
        <ns2:commandClass>undefined</ns2:commandClass>
        <ns2:commandId>0</ns2:commandId>
        <ns2:commandName>G2SACK</ns2:commandName>
```

```
        <ns2:comment/>
        <ns2:content>&lt;?xml version="1.0" encoding="UTF-8" standalone="yes"?&gt;
        &lt;g2s:g2sMessage
        xmlns:g2s="http://www.gamingstandards.com/g2s/schemas/v1.0.3"&gt;
        &lt;g2s:g2sAck g2s:dateTimeSent="2011-09-14T09:42:17.077-07:00" g2s:egmId="RBG_
        1234"
        g2s:hostId="1"/&gt;
        &lt;/g2s:g2sMessage&gt;</ns2:content>
        <ns2:dateReceived>2011-09-14T09:42:17.077-07:00</ns2:dateReceived>
        <ns2:errorCode>G2S_none</ns2:errorCode>
        <ns2:errorText>G2S_none</ns2:errorText>
        <ns2:eventCode>G2S_none</ns2:eventCode>
        <ns2:eventText>G2S_none</ns2:eventText>
        <ns2:fromLocation>Host ID 1</ns2:fromLocation>
        <ns2:direction>Outbound</ns2:direction>
        <ns2:end>Host</ns2:end>
        <ns2:messageId>882</ns2:messageId>
        <ns2:sessionId>0</ns2:sessionId>
        <ns2:sessionType>N/A</ns2:sessionType>
        <ns2:summary>G2SACK</ns2:summary>
        <ns2:timeToLive>-1</ns2:timeToLive>
        <ns2:toLocation>RBG_1234</ns2:toLocation>
        <ns2:transactionId>-1</ns2:transactionId>
    </ns2:message>
    <ns2:message>
        <ns2:commandClass>communications</ns2:commandClass>
        <ns2:commandId>218</ns2:commandId>
        <ns2:commandName>communications.keepAliveAck</ns2:commandName>
        <ns2:comment/>
        <ns2:content>&lt;?xml version="1.0" encoding="UTF-8"
        standalone="yes"?&gt;&lt;g2s:g2sMessage
        xmlns:g2s="http://www.gamingstandards.com/g2s/schemas/v1.0.3"&gt;&lt;g2s:g2sBody
        g2s:dateTimeSent="2011-09-14T09:42:17.083-07:00" g2s:egmId="RBG_1234"
        g2s:hostId="1"&gt;&lt;g2s:communications g2s:commandId="218" g2s:dateTime="2011-09-
        14T09:42:17.076-07:00" g2s:deviceId="1" g2s:errorCode="G2S_none" g2s:errorText=""
        g2s:sessionId="3000192" g2s:sessionMore="false" g2s:sessionRetry="false"
        g2s:sessionType="G2S_response"
        g2s:timeToLive="0"&gt;&lt;g2s:keepAliveAck/&gt;&lt;/g2s:communications&gt;&lt;
          /g2s:g2sBody&gt;&lt;/g2s:g2sMessage&gt;</ns2:content>
        <ns2:dateReceived>2011-09-14T09:42:17.083-07:00</ns2:dateReceived>
        <ns2:errorCode>G2S_none</ns2:errorCode>
        <ns2:errorText>G2S_none</ns2:errorText>
        <ns2:eventCode>G2S_none</ns2:eventCode>
        <ns2:eventText>G2S_none</ns2:eventText>
        <ns2:fromLocation>Host ID 1</ns2:fromLocation>
        <ns2:direction>Outbound</ns2:direction>
        <ns2:end>Host</ns2:end>
        <ns2:messageId>883</ns2:messageId>
        <ns2:sessionId>3000192</ns2:sessionId>
        <ns2:sessionType>G2S_response</ns2:sessionType>
        <ns2:summary>communications.keepAliveAck</ns2:summary>
        <ns2:timeToLive>0</ns2:timeToLive>
        <ns2:toLocation>RBG_1234</ns2:toLocation>
        <ns2:transactionId>-1</ns2:transactionId>
    </ns2:message>
    <ns2:message>
```

```
        <ns2:commandClass>undefined</ns2:commandClass>
        <ns2:commandId>0</ns2:commandId>
        <ns2:commandName>G2SACK</ns2:commandName>
        <ns2:comment/>
        <ns2:content>&lt;?xml version="1.0" encoding="UTF-8" standalone="yes"?&gt;
        &lt;g2s:g2sMessage
        xmlns:g2s="http://www.gamingstandards.com/g2s/schemas/v1.0.3"&gt;
        &lt;g2s:g2sAck g2s:dateTimeSent="2011-09-14T09:42:17.098-07:00" g2s:egmId="RBG_
        1234"
        g2s:hostId="1"/&gt;
        &lt;/g2s:g2sMessage&gt;</ns2:content>
        <ns2:dateReceived>2011-09-14T09:42:17.108-07:00</ns2:dateReceived>
        <ns2:errorCode>G2S_none</ns2:errorCode>
        <ns2:errorText>G2S_none</ns2:errorText>
        <ns2:eventCode>G2S_none</ns2:eventCode>
        <ns2:eventText>G2S_none</ns2:eventText>
        <ns2:fromLocation>RBG_1234</ns2:fromLocation>
        <ns2:direction>Inbound</ns2:direction>
        <ns2:end>Client</ns2:end>
        <ns2:messageId>884</ns2:messageId>
        <ns2:sessionId>0</ns2:sessionId>
        <ns2:sessionType>N/A</ns2:sessionType>
        <ns2:summary>G2SACK</ns2:summary>
        <ns2:timeToLive>-1</ns2:timeToLive>
        <ns2:toLocation>Host ID 1</ns2:toLocation>
        <ns2:transactionId>-1</ns2:transactionId>
    </ns2:message>
</G2STranscript2SnippetResponse>
```

Method: insertTranscriptMarker

This method is used to insert a transcript marker into the transcript. You can use transcript markers when fetching transcript records (for example, get all records between a starting marker and an ending marker), making it easier to find the records you want.

Note that all elements and attributes are *required* for each method.

HTTP Type: POST

Call Type: Synchronous

Request: TranscriptMarkerRequest

Attribute	Restrictions	Description
transcriptMarker	type: string minLength: 0 maxLength: 128	Identifier for the transcript marker to be inserted.

Example

```
<TranscriptMarkerRequest
xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1"><transcriptMarker>Marker
#1</transcriptMarker></TranscriptMarkerRequest>
```

Response: SimpleCallResult

Attribute	Restrictions	Description
error	type: boolean minOccur: 1 maxOccur:1	Error associated with the asynchronous API call.
message	type: string minOccur: 1 maxOccur:1	Message associated with the asynchronous API call.

Example

```
<SimpleCallResult xmlns:ns3="http://www.radblue.com/g2s/transcript2/api/schemas/v1.0.0"
xmlns:ns2="http://www.radblue.com/g2s/common/api/schemas/v1.0.0"
xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1"><error>false</error><message>The
API call finished successfully.</message></SimpleCallResult>
```

CommsState Enumeration

Enumeration	Description
G2S_closed	Communications between the G2S devices have been closed.
G2S_closing	Communications between the G2S devices are in the process of closing.
G2S_onLine	Communications between the EGM and host system have been established and they are currently communicating.
G2S_opening	Communications between the G2S devices are being established.
G2S_overflow	Outbound queue of one of the G2S devices is in an overflow state, and messages cannot be queued.
G2S_sync	Communications between the G2S devices are currently synchronizing.
RBG_lost	Communications between the G2S devices have been lost.

responseType

Enumeration	Description
GamePlayResponse	Indicates that the `attributes` element's name-value pair reflects game play information.
InsertIdResponse	Indicates that the `attributes` element's name-value pair reflects player identifier information.
RedeemVoucherResponse	Indicates that the `attributes` element's name-value pair reflects voucher redemption information.
WatToEgmResponse	Indicates that the `attributes` element's name-value pair reflects WAT-to-EGM information.
WatToHostResponse	Indicates that the `attributes` element's name-value pair reflects WAT-to-host information.
Unknown	Indicates that the `attributes` element's name-value pair reflects information of an undetermined type.

Status Enumeration

Enumeration	Description
RC_success	Indicates the script has completed successfully.
RC_preparingToRun	Indicates the script has been executed and is preparing to run.
RC_running	Indicates the script is currently running.
RC_error	Indicates that the script was executed, but resulted in an error.

StatusErrorCode Enumeration

Enumeration	Description
RC_success	Script has completed successfully.
RC_RM001	Active response manager file name is missing.
RC_RM100	Unknown response manager exception.
RC_RS001	EGM ID is missing.
RC_RS002	File name is missing.
RC_RS003	EGM ID is not online.
RC_RS004	Script ID is missing.
RC_RS005	Script contains invalid verbs.
RC_RS006	Could not locate script file on disk.
RC_RS007	EGM is not in G2S_online state.
RC_RS100	Unknown run script exception.
RC_SA001	Active start-up algorithm file name is missing.
RC_SA100	Unknown start-up algorithm exception.
RC_SH100	Unknown schedule exception.

TicketCode

Enumeration	Description
OK	Ticket accepted.
ERROR	Invalid ticket information.
IN_PROGRESS	Ticket status in progress. Check status at a later time.
BAD_ARGUMENT	Invalid code.

XML Data Types

RadBlue uses standard XML data types as defined by the W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes section of version 1.1 of the XML schema.

- boolean
- dateTime
- int
- long
- string

Index

A

accessing the transcript

fetchStatus 33

fetchTranscript 35

insertTranscriptMarker 45

attributes 34

B

boolean 49

C

callResult 34

code 34

commandClass 35, 37-38

commandId 38

commandName 38

comment 38

commsState 21, 47

communications not online 12, 27

connectedEgms 21

connection URL 15

connectionDate 21

content 38

D

data types 49

boolean 49

dateTime 49

long 49

string 49

date types

int 49

dateGenerated 37

dateReceived 38

dateTime 49

deviceId 35, 37

direction 38

E

egm 21

egmId 21, 23, 27, 35, 37

egmIdRequest 27

egmList 21

end 38

endDate 35, 37

endMessageId 35, 37

endTranscriptMarker 35, 37

enumeration
- commsState 47
- responseType 47
- status 48
- statusErrorCode 48
- ticketCode 48

error 45
errorCode 23, 27, 29, 31
errorText 23, 27, 29, 31
eventCode 38
eventText 38

F

fetchStatus 33
fetchTranscript 35
fileName 23, 29, 31
fileNameRequest 29, 31
fromLocation 38

G

g2s_closed 47
g2s_closing 47
g2s_onLine 47
g2s_opening 47
g2s_overflow 47
g2s_sync 47
g2sTranscript2SnippetRequest 35
g2sTranscript2SnippetResponse 37
gamePlayResponse 47
getting started 15

H

hostId 35, 37

I

includeG2sAcks 36-37
insertIdResponse 47
insertTranscriptMarker 45
int 49
interface example 16

L

long 49

M

maxRecords 36-37
message 34, 37, 45
messageId 38
method
- connectedEgms 21
- fetchStatus 33
- fetchTranscript 35
- insertTranscriptMarker 45
- runScripts 23
- runScriptStatus 25
- scheduleMsxOO3 27

setActiveResponseManager 29
setActiveStartupAlgorithm 31

N

name 34

P

percent 25

R

rbg_lost 47
rc_error 48
rc_preparingToRun 48
rc_rm001 48
rc_rm100 48
rc_rs001 48
rc_rs002 48
rc_rs003 48
rc_rs004 48
rc_rs005 48
rc_rs006 48
rc_rs007 48
rc_rs100 48
rc_running 48
rc_sa001 48
rc_sa100 48
rc_sh100 48
rc_success 48
redeemVoucherResponse 47
remote control api 16
request
connectedEgms 21
egmIdRequest 27
fileNameRequest 29, 31
g2sTranscript2SnippetRequest 35
runScriptRequest 23
runScriptStatusRequest 25
TicketStatusRequest 33
transcriptMarkerRequest 45
response
CallResult 34
egmList 21
g2sTranscript2SnippetResponse 37
runScriptResponse 23
runScriptStatusResponse 25
simpleCallResult 45
status 27, 29, 31
responseType 34, 47
rst rest
sample interface 16
use case 15
runScripts 23
runScriptStatus 25

S

sample interface 16

scheduleMsx003 27

schemaVersion 21

scratch pad 16

scriptId 23, 25

sessionId 38

sessionType 38

setActiveResponseManager 29

setActiveStartupAlgorithm 31

simpleCallResult 45

startDate 36-37

startMessageId 36-37

startTranscriptMarker 36-37

status 25, 29, 31, 48

statusErrorCode 48

string 49

summary 38

T

test browser connection 19

text 25

ticket 33-34

ticketCode 48

ticketStatusRequest 33

timeToLive 38

toLocation 38

transactionId 38

transcriptMarker 45

transcriptMarkerRequest 45

transcriptSize 37

U

use case 15

using programming guide 13

V

value 34

W

w3c xml schema definition language 49

watToEgmResponse 47

watToHostResponse 47

X

xml data types 49

xml schema definition language 49

xsd 49