RST REST Interface Programming Guide

03 OCT 2012

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

About the RST REST Interface

The REST (Representational State Transfer) interface allows you to control the RadBlue System Tester (RST) remotely to test a G2S host and analyze the G2S messages that are produced by that host.

For example, the REST interface can:

1. Request the current EGM ID.
2. Play a simple EGM game.
3. Insert coins.
4. Insert an ID.
5. Remove an ID.
6. Insert a note.
7. Insert a transcript marker.
8. Issue a voucher.
9. Redeem a voucher.
10. Check the status of an asynchronous action.

In addition to sending standard EGM-related commands, you can perform negative testing, request Message Transcript date, send EGM-initiated commands.

Method Overview

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

Method	Description	When to Use
cabinetEvents	The `cabinetEvent` method tells RST to generate one or more cabinet events.	Use this method to generate cabinet related events.
cancelCancelCreditHandpay	The `cancelCancelCreditHandpay` method lets you cancel a pending Cancel Credit Handpay transaction.	Use this method when you need to cancel a pending Cancel Credit handpay.
cashout	The `cashOut` method lets you cash out to a handpay.	Use this method to initiate a cash out - it creates a Cancel Credit transaction.
changeDoorState	The `changeDoorState` method lets you open and close the EGM cabinet doors (cabinet, logic or auxiliary).	Use this method to initiate changes to the cabinet doors.
changeEgmId	The `changeEgmId` method lets you change the EGM identifier used by RST.	Use this method to modify the EGM ID. Note that RST **must** be stopped in order to change the EGM Id.
coinAcceptorEvents	The `coinAcceptorEvents` method tells RST to create one or more events for a specified coin acceptor device.	Use this method to generate coin acceptor events.
coinDrop	The `coinDrop` method causes RST to open, then close the coin drop door.	Use this method to simulate the removal of the coin drop.
dispenseCoins	The `dispenseCoins` method lets you dispense coins from the hopper.	Use this method to dispense one or more coins from the hopper.
dispenseNotes	The `dispenseNotes` method lets you dispense note from the note dispenser device.	Use this method to dispense one or more notes from the note dispenser..
fetchAllModifications	The `fetchAllModifications` method is used to fetch the current set of message modifications (which may be empty).	Use this method to fetch all of the currently configured message modification definitions.
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 records in the current transcript.	This is the key method for building your test system. You will need to fetch transcript records, parse them and then dig into the resulting data. There are a lot of different ways of doing this. The option

Method	Description	When to Use
		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.
forceMsx003	The `forceMsx003` method lets you put the communications channel in a lost state by sending an `MSX003` command to the host.	This method causes the EGM to treat the host as an unknown host (rejecting every message). To clear this condition, you must stop and then restart RST.
getCountdownOverride	The `getCountdownOverride` method tells the RST to send the `getCountdownOverride` G2S command to the player host.	This method is used to request new countdown override information from the host that owns the player device.
getDescriptorList	The `getDescriptorList` method is used to fetch descriptors from the current EGM. This method is similar to the `getDescriptor` command in G2S. You can specify 1 or all classes, and 1 or all devices.	Use this method to view the RST's current list of devices (defined by the configuration file).
getEgmId	The `getEgmId` method gets the EGM ID of the currently loaded EGM in RST.	Use this method to confirm that RST is using the expected EGM ID.
getMcastKeyUpdate	The `getMcastKeyUpdate` method tells RST to request a new set of multicast keys from the host.	Using this method, RST will request a new set of multicast keys, using the specified communications device and the multicastId.
insertCoins	The `insertCoins` method inserts one or more coins of the same denomination, incrementing the cashable credit meter.	Use this method to put credits on the credit meter using coins.
insertId*	The `insertId` method inserts an ID into an ID reader device	Use this method to start a carded game play session.
insertNote	The `insertNote` method inserts a single note into the EGM, which increments the cashable credit meter.	Use this method to put credits on the credit meter using notes.
insertTranscriptMarker	The `insertTranscriptMarker`	Use this method to insert a transcript

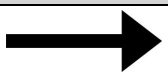
Method	Description	When to Use
	method is used to insert a transcript marker into the RST transcript.	marker. You can use transcript markers when fetching transcript records, to make it easier to find the records you want.
issueVoucher	The `issueVoucher` method creates a new voucher, moving credits from the game to the voucher system.	Use this method if you need to transfer funds from the specified credit meter to a voucher..
keyOff	The `keyOff` method is used to key off a pending handpay.	Use this method if you want to key off a pending handpay.
modificationsAddList	The `modificationsAddList` method is used to add a new set of message modifications to RST.	Messages modification commands allow you to add, remove, or change an attribute in an outbound G2S message.
modificationsClearAll	The `modificationsClearAll` method is used to clear the current set of message modifications.	This method is used to clear ALL of the active message modification records in RST.
modificationsDeleteList	The `modificationsDeleteList` method is used to delete one or more message modifications, by Id.	This method is used to clear SELECTED message modification records in RST.
noteAcceptorEvents	The `noteAcceptorEvents` method tells RST to generate one or more note acceptor events for a specified device.	Use this method to generate note acceptor events.
noteDrop	The `noteDrop` method causes RST to open the note drop door, remove and then reinsert the stacker, and then close the note drop door.	Use this method to simulate the removal of the note drop.
playPaytableGame*	The `playPaytableGame` method tells RST to play a paytable game (using the outcome database).	Use this method to play a paytable game on the EGM, generating a random or pre-determined outcome from the outcome database.
playSimpleGame*	The `playSimpleGame` method plays a simple game, in which you specify all of the details.	Use this method to simulate game play.
printerEvents	The `printerEvents` method tells RST to generate one or more printer events for the specified device.	Use this method to generate printer events.
progressiveGetHostInfo	The `progressiveGetHostInfo` method tells RST to request progressive host information from the owner of the specified progressive device.	This method is used to request progressive host information from the host that owns the device.
redeemVoucher	The `redeemVoucher` method redeems a voucher, moving credits from the voucher	Use this method to redeem a voucher to transfer funds onto the game.

Method	Description	When to Use
	system to the game.	
sendCommHostList	The `sendCommHostList` method tells RST to send a commHostList to the host that owns the specified device.	Use this method to cause RST to generate a `commHostList` to send its configuration information to the host owning the specified device.
sendMessage	The `sendMessage` method is an incredibly powerful tool for creating a message that the RST sends to the specified host.	With this method, you can specify a G2S message, including adding, removing or changing class level elements . The RST will fix up the message, if desired.
sendOptionList	The `sendOptionList` method tells RST to send an optionList to the host that owns the specified device.	Use this method to cause RST to generate an `optionList` to send the a set of options that you select to the host owning the specified device.
startEgm	The `startEGM` method tells the RST to start the engine.	Use this method whenever you want to start the RST engine.
stopEgm	The `stopEGM` method tells the RST to stop the engine.	Use this method whenever you want to stop the RST engine.
takeOutId	The `takeOutId` method removes the current ID from an ID reader device.	Use this method to end a carded session.
watGetAccounts	The `watGetAccounts` method tells RST to request a listing of the active player's WAT accounts.	Use this method to request the active player's WAT account list from the owner of the specified WAT device
watGetBalance	The `watGetBalance` method tells RST to request the balances for the specified WAT account.	Use this method to request the current balances from the specified WAT account.
watGetKeyPair	The `watGetKeyPair` method tells RST to request a new WAT keypair for the specified WAT device.	Use this method to request a new keypair from the owner of the specified WAT device.
watToEgm*	The `watToEgm` method tells RST to request a WAT transfer from the specified WAT account to the EGM.	Use this method to transfer funds from a selected account on the host to the EGM.
watToHost*	The `watToHost` method tells RST to request a WAT transfer from the EGM to the specified WAT account.	Use this method to transfer funds from the EGM to the specified WAT account on the host.

*Indicates that method is *asynchronous*.

Notes on Using this Document

- Methods in this document are grouped by function:
  - **Accessing the Transcript** - Includes methods that let you access Message Transcript data.
  - **Controlling the EGM** - Includes methods that let you start RST, stop RST, get the descriptor list and force RST into a lost state (MSX003).
  - **Playing the EGM** - Includes methods that simulate activities at an EGM.
  - **Sending Events** - Includes methods that let you send events.
  - **Sending WAT Messages** - Includes methods that let you send WAT information.
  - **Sending EGM-Initiated Requests** - Includes methods for sending EGM-initiated requests.
  - **Performing Negative Testing** - Includes methods that allow you to perform negative testing.
- 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:

RST REST		Host
Call method sent to host.		
		Host returns a ticket number.
`fetchStatus` sent to host.		
		Result of call method sent to RST 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

- G2S Message Protocol
- RemoteControl.zip (reference implementation)
- RST User Guide
- W3C XML Schema Definition Language (XSD)1.1 Part 2: Datatypes

Get Going with RST REST

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

1. Review the sample use case to understand the general message flow of RST messages.
2. Open the **RemoteControl.zip** reference implementation, which contains a full .NET project showing how to connect to the RST through a REST interface and issues sample calls.
3. Create a program, using the programming language of your choice, that connects to RST through the following URL: **http://127.0.0.1:38501/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.

View a Sample Use Case

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

1. insertTranscriptMarker - Insert transcript marker **Marker #1**.
2. insertId - Insert player ID **12345678**.
3. insertNote - Insert **$1** into the note acceptor.
4. insertCoins - Insert **$1** in coins into the coin acceptor.
5. playSimpleGame - Play a simple game, losing **$1**.
6. issueVoucher - Transfer credit meter to the voucher system.
7. takeOutId - Remove the player ID.
8. insertTranscriptMarker - Insert transcript marker **Marker #2**.
9. fetchTranscript - Fetch all of the G2S transcript entries between **Marker #1** and **Marker #2**.

Use the RST Scratch Pad for Reference

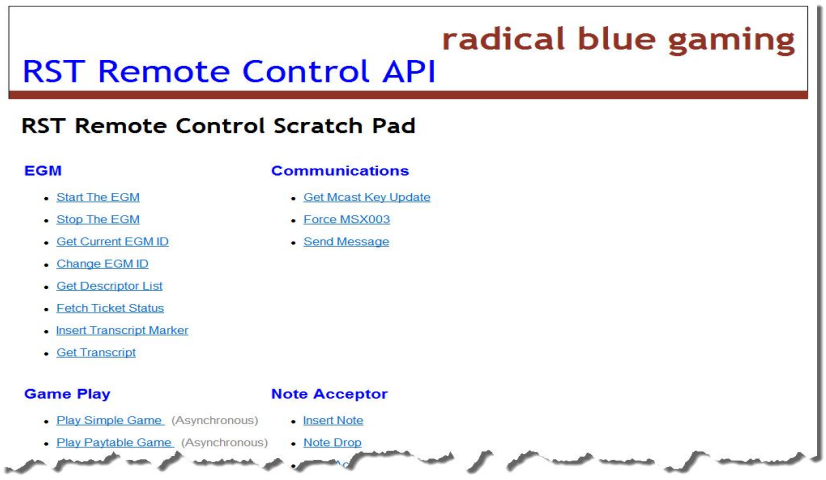
The RST Remote Control Scratch Pad is a demonstration of what an RST REST interface might look like. It contains all of the supported methods. Once you've sent a method, the XML for both the request and response displays as well as a summary of the response.

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

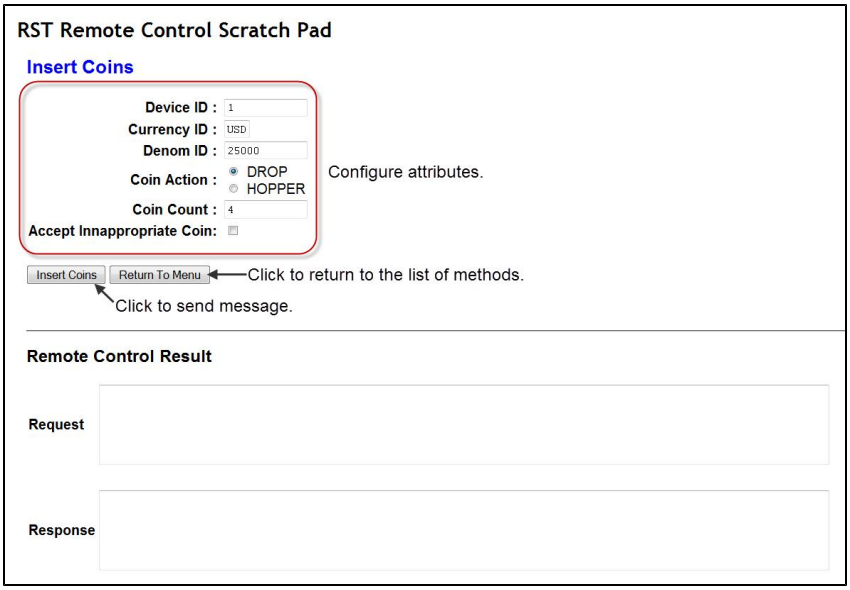
1. Launch RST.
2. Go to: **Tools** > **Configure** > **License Manager**.
3. Click **View Features**.
4. Verify that the **remoteControl** field is **true**. If not, contact RadBlue.

Now, to access the RST Scratch Pad:

1. Start RST by clicking **Start SmartEGM** on the **Main** tab of the **SmartEGM** layout.
2. Select the **Debug Log** layout.
3. Scroll up to the top of the log.
4. Scroll down until you see the following line:
   ```
   2012-01-26T09:07:26.385-08:00 [INFO] {svc-remote-control} *
   ScratchPad URL:
   http://localhost:38501/RST/pages/remotecontrol/ScratchPad.html
   ```
5. Highlight **http://localhost:38501/RST/pages/remotecontrol/ScratchPad.html**, and click **CTRL+c** to copy the selection.
6. Open either a **Firefox** or **Safari** browser.
7. Paste the HTTP location into the address window, and press **Enter**.



8. Click any method to view its send options. In this example, we chose **Insert Coins**.



9. Configure method attributes, if any, and click **Insert Coins**.

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.

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

Test the Remote Control URL

You can verify that the Remote Control URL is working in RST. Use this procedure if your program is not getting a response from RST *or* if your program is getting an error from RST.

1. Open **RST**.
2. Configure **RST** to communicate with either the RadBlue G2S Simulator (RGS) or the host system of your choice.
3. Start **RST**.
4. Open a browser.
5. In the address bar, type **http://127.0.0.1:38501/remote/egmId**, and press **Enter**.
   If your URL is valid, RST will return the following:

   <EgmIdResponse><egmId>RBG_1234</egmId></EgmIdResponse>

Method: changeEgmId

The `changeEgmId` method lets you change the EGM identifier used by RST.

Note: The EGM must be stopped to change the EGM identifier, or else an error is returned.

To change the EGM identifier through the RST user interface, select the **Main** tab on the **SmartEGM** layout, and click **Change** under the **Configuration Control** section. Note that RST must be stopped before the **Change** button is enabled.

HTTP Type: POST

Call Type: Synchronous

Request: ChangeEgmIdRequest

Element	Restrictions	Description
egmId	type:egmId	New EGM identifier.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request changes the EGM ID to **RBG_00:22:4F:37:56:4F**.

```
<ChangeEgmIdRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <egmId>RBG_00:22:4F:37:56:4F</egmId>
</ChangeEgmIdRequest>
```

Method: forceMsx003

The `forceMsx003` method lets you put the communications channel in a lost state by sending an MSX003 command to the host. Note that this behavior is *not* part of the official G2S specification. To clear this state, you must stop and then restart RST.

HTTP Type: POST

Call Type: Synchronous

Request: *Not Applicable*

There is no request information for this method.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Method: getDescriptorList

The `getDescriptorList` method is used to fetch descriptors from the RST. This method is similar to the `getDescriptor` command in G2S. You can specify one or all classes and one or all devices to be returned.

Use this method to confirm exactly what devices are loaded in the current EGM. The logic of your application can then change, depending on which G2S devices are present.

HTTP Type: POST

Call Type: Synchronous

Request: GetDescriptorRequest

Element	Restrictions	Description
deviceClass	type: deviceClass minOccurs: 1 maxOccurs: 1	Device class to use in the filtering of descriptors. A **G2S_all** value indicates all device classes.
deviceId	type: int minInclusive: -1	Device identifier to use in the filtering of descriptors. A **-1** value indicates all device instances.

Response: DescriptorList

Element	Restrictions	Description
descriptor	type: descriptor	List of requested device descriptors from RST.

Descriptor

Element	Restrictions	Description
configId	type: long minOccurs: 0 maxOccurs: 1	Host identifier of the host that configures the device.
configurationId	type: boolean minOccurs: 0 maxOccurs: 1	Configuration identifier most recently set by the configuration host (default=0).
deviceActive	type: boolean minOccurs: 0 maxOccurs: 1	Indicates whether the device is active.
deviceClass	type: deviceClass	G2S device class.

Element	Restrictions	Description
deviceConfig	type: boolean minOccurs: 0 maxOccurs: 1	This value is always **false** because the request is not from a G2S host.
deviceGuest	type: boolean minOccurs: 0 maxOccurs: 1	This value is always **false** because the request is not from a G2S Host.
deviceId	type: int minOccurs: 0 maxOccurs: 1	Identifies a specific device within the class.
deviceOwner	type: boolean minOccurs: 0 maxOccurs: 1	This value is always false because the request is not from a G2S host.
egmEnabled	type: boolean minOccurs: 0 maxOccurs: 1	Indicates whether the device is currently enabled by the EGM.
egmLocked	type: boolean minOccurs: 0 maxOccurs: 1	Indicates whether the device is currently locked by the EGM.
hostEnabled	type: boolean minOccurs: 0 maxOccurs: 1	Indicates whether the device is currently enabled by the owner host.
hostLocked	type: boolean minOccurs: 0 maxOccurs: 1	Indicates the device is currently locked by the owner host.
ownerId	type: long minOccurs: 0 maxOccurs: 1	Host identifier of the host that owns the device.
productId	type: string minOccurs: 0 maxOccurs: 1	Product identifier of the physical device.
productName	type: string minOccurs: 0 maxOccurs: 1	Product name of a physical device.
releaseNumber	type: string minOccurs: 0 maxOccurs: 1	Software release number of a physical device.
serialNumber	type: string minOccurs: 0 maxOccurs: 1	Serial number of a physical device.
vendorId	type: string minOccurs: 0	Vendor Identifier of a physical device.

Element	Restrictions	Description
	maxOccurs: 1	
vendorName	type: string minOccurs: 0 maxOccurs: 1	Vendor name of the manufacturer of a physical device.

Example

This request returns all RST devices.

```
<GetDescriptorRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceClass>G2S_all</deviceClass>
    <deviceId>-1</deviceId>
</GetDescriptorRequest>
```

This request returns all of the gamePlay devices in RST.

```
<GetDescriptorRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceClass>G2S_gamePlay</deviceClass>
    <deviceId>-1</deviceId>
</GetDescriptorRequest>
```

Method: getEgmId

The `getEgmId` method gets the EGM ID of the currently loaded EGM in RST. Use this method to confirm that the loaded EGM has the expected EGM ID.

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

HTTP Type: GET

Call Type: Synchronous

Request: *Not Applicable*

There is no request information for this method.

Response: EgmIdResponse

Attribute	Restrictions	Description
EgmId	type: egmId	EGM identifier.

Method: startEgm

The `startEgm` method starts the RST engine. The call returns immediately because bringing the EGM online can be time consuming.

To start the RST engine through the user interface, select the **Main** tab on the **SmartEGM** layout, and click **Start SmartEGM**.

HTTP Type: POST

Call Type: Synchronous

Request: *Not Applicable*

There is no request information for this method.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Method: stopEgm

The `stopEgm` method stops the RST engine. The call returns immediately because bringing down the RST can be time consuming.

To stop the RST engine through the user interface, select the **Main** tab on the **SmartEGM** layout, and click **Stop SmartEGM**.

HTTP Type: POST

Call Type: Synchronous

Request: *Not Applicable*

There is no request information for this method.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

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>12345678912345678912345678912345678912345678912345678912345678912345678912345678</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>123456789123456789123456
6789123456789</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>
```

Method: cancelCancelCreditHandpay

The `cancelCancelCreditHandpay` method is used to cancel the pending Cancel Credit handpay transaction, if one exists.

HTTP Type: POST

Call Type: Synchronous

Request: HandpayCancelCancelCreditHandpay

Element	Restrictions	Description
deviceId	type: int minInclusive: -1	Handpay device identifier.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes RST to request the cancellation of the cancel credit handpay.

```
<HandpayCancelCancelCreditHandpay xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
</HandpayCancelCancelCreditHandpay>
```

Method: cashOut

The `cashOut` method lets you cash out to a handpay. Use this method to create a Cancel Credit handpay transaction.

To initiate a cash out through the RST user interface, select the **Player Verbs** tab on the **SmartEGM** layout, and click **Cash Out to Handpay**.

HTTP Type: POST

Call Type: Synchronous

Request: CashOutRequest

Element	Restrictions	Description
deviceId	type: int minInclusive: -1 minOccurs: 1 maxOccurs: 1	EGM device identifier.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes RST to cashout the current player.

```
<CashOutRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
</CashOutRequest>
```

Method: coinDrop

The `coinDrop` method lets you simulate the removal of the coin drop (Coin drop door opens and then closes).

To simulate the coin drop through the RST user interface, select the **PlayerVerbs** tab on the **SmartEGM** layout, and click **Perform Coin Drop**.

HTTP Type: POST

Call Type: Synchronous

Request: CoinDropRequest

Element	Restrictions	Description
deviceId	type: int minInclusive: -1 minOccurs: 1 maxOccurs: 1	Coin acceptor device that performs the coin drop.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes a coin drop.

```
<CoinDropRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
</CoinDropRequest>
```

Method: dispenseCoins

The `dispenseCoins` method lets you dispense coins from the hopper.

To dispense coins through the RST user interface, select the **PlayerVerbs** tab on the **SmartEGM** layout, and click **Dispense Coin(s)**.

HTTP Type: POST

Call Type: Synchronous

Request: DispenseCoinsRequest

Element	Restrictions	Description
all	type: boolean	If **false**, the value of the count element is honored. If **true**, the count element is ignored and RST dispenses as many coins as possible given the current credit meter.
count	type: int	Number of coins to dispense.
creditType	type:creditType	Type of credit to dispense (for example, cashable).
currencyId	type: string	Currency identifier of the coin dispensed.
denomId	type: long	Denomination identifier of the coin dispensed.
deviceId	type: int minInclusive: -1 minOccurs: 1 maxOccurs: 1	Coin acceptor device that performs the coin drop.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes RST to dispense five (5) US quarter coins.

```
<DispenseCoinsRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
    <currencyId>USD</currencyId>
    <denomId>25000</denomId>
    <creditType>CASHABLE</creditType>
    <count>5</count>
    <all>false</all>
</DispenseCoinsRequest>
```

Method: dispenseNotes

The `dispenseNotes` method lets you dispense note from the note dispenser device.

To dispense notes through the RST user interface, select the **PlayerVerbs** tab on the **SmartEGM** layout, and click **Dispense Note(s)**.

HTTP Type: POST

Call Type: Synchronous

Request: DispenseNotesRequest

Element	Restrictions	Description
all	type: boolean	If **false**, the value of the count element is honored. If **true**, the count element is ignored and RST dispenses as many notes as possible given the current credit meter.
count	type: int	Number of notes to dispense.
creditType	type:creditType	Type of credit to dispense (for example, cashable).
currencyId	type: string	Currency identifier of the note dispensed.
denomId	type: long	Denomination identifier of the note dispensed.
deviceId	type: int minInclusive: -1 minOccurs: 1 maxOccurs: 1	Note acceptor device that performs the note drop.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes RST to dispense five (5)US dollar bills.

```
<DispenseNotesRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
    <currencyId>USD</currencyId>
    <denomId>100000</denomId>
    <creditType>CASHABLE</creditType>
    <count>5</count>
    <all>false</all>
</DispenseNotesRequest>
```

Method: insertCoins

This method inserts one or more coins of the same denomination, incrementing the cashable credit meter. Use this method to put credits on the credit meter.

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

HTTP Type: POST

Call Type: Synchronous

Request: InsertCoinsRequest

Attribute	Restrictions	Description
acceptInappropriateCoin	type: boolean	If **true**, RST generates a `G2S_CAE105` (Inappropriate Coin Not Returned) event.
coinAction	type: coinAction	Determines whether the coin goes to the hopper or the drop when it is inserted.
coinCount	type: long	Number of coins inserted.
deviceId	type: int minInclusive: -1	Device identifier.
money	type: G2sMoney	Specifies the currency and denomination of the inserted coin.

G2sMoney Elements

Element	Restrictions	Description
currencyId	type: string minLength: 3 maxLength: 3	Currency identifier - ISO 4217 3-character Alpha.
denomId	type: long	Denomination identifier.

Example

```
<InsertCoinsRequest
xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1"><deviceId>1</deviceId>
<coinAction>DROP</coinAction><money><currencyId
xmlns="http://www.radblue.com/g2s/common/api/schemas/
v1.0.0">USD</currencyId><denomId
xmlns="http://www.radblue.com/g2s/common/api/schemas/v1.0.0">
25000</denomId></money><coinCount>4</coinCount><acceptInappropriateCoin>false
</acceptInappropriateCoin></InsertCoinsRequest>
```

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>
```

Method: insertId

This method inserts an ID into an ID reader device. It is an asynchronous method, so you should follow it with a `fetchStatus` method to determine the final status. Use this method for carded game play sessions.

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

To insert an ID through the RST user interface, select the **Player Verbs** tab on the **SmartEGM** layout, and click **Insert ID**.

HTTP Type: POST

Call Type: Asynchronous

> **Note:** The call result for this asynchronous method appears after the "Response" section. See `fetchStatus` for details on that method.

Request: InsertIdRequest

Attribute	Restrictions	Description
deviceId	type: int minInclusive: -1	EGM device identifier.
idNumber	type: string minLength: 1 maxLength: 64	Player or employee identifier.

Response: Ticket

Attribute	Restrictions	Description
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.
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

```
<InsertIdRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
<deviceId>1</deviceId><idNumber>12345678</idNumber></InsertIdRequest>
```

InsertIdResponse

Name	Value
fullName	Full name of player.
idType	G2S identifier type of the player.
idNumber	ID number for the player.
preferName	Player's preferred name.
playerId	Player identifier.

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>{461aace7-3b7b-40b3-80f8-cc4c5dfa29ca}</ticket>
    <responseType>InsertIdResponse</responseType>
    <code>RBG_OK</code>
    <message>ID has been inserted.</message>
    <attributes>
        <ns2:element>
          <ns2:name>fullName</ns2:name>
          <ns2:value>Elvis Presley</ns2:value>
        </ns2:element>
        <ns2:element>
          <ns2:name>idType</ns2:name>
          <ns2:value>G2S_player</ns2:value>
        </ns2:element>
        <ns2:element>
          <ns2:name>idNumber</ns2:name>
          <ns2:value>12345678</ns2:value>
        </ns2:element>
        <ns2:element>
          <ns2:name>preferName</ns2:name>
          <ns2:value>Elvis</ns2:value>
        </ns2:element>
        <ns2:element>
          <ns2:name>playerId</ns2:name>
          <ns2:value>P-12345678</ns2:value>
        </ns2:element>
    </attributes>
</CallResult>
```

Method: insertNote

This method inserts a single note into the EGM, which increments the cashable credit meter. Use this method to put credits on the credit meter.

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

HTTP Type: POST

Call Type: Synchronous

Request: InsertNoteRequest

Attribute	Restrictions	Description
deviceId	type: int minInclusive: -1	Device identifier.
money	type: G2sMoney	Specifies the currency and denomination of the inserted note.
noteAction	type: noteAction	Determines what happens to the note once it is inserted (goes to note dispenser, note drop, gets rejected or is returned).

Example

```
<InsertNoteRequest
xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1"><deviceId>1</deviceId><money>
<currencyId xmlns="http://www.radblue.com/g2s/common/api/schemas/v1.0.0">
USD</currencyId><denomId xmlns="http://www.radblue.com/g2s/common/api/schemas/v1.0.0">
100000</denomId></money><noteAction>DROP</noteAction></InsertNoteRequest>>
```

G2sMoney Elements

Element	Restrictions	Description
currencyId	type: string minLength: 3 maxLength: 3	Currency identifier - ISO 4217 3-character Alpha.
denomId	type: long	Denomination identifier.

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>
```

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>
```

Method: issueVoucher

This method creates a new voucher, moving credits from the game to the voucher system. Use this method if you need to transfer funds off of the game.

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

HTTP Type: POST

Call Type: Synchronous

Request: IssueVoucherRequest

Attribute	Restrictions	Description
creditType	type: creditType	Indicates whether the voucher is cashable, non-cashable or promotional.
idReaderDeviceId	type: int minInclusive: -1	Device identifier for the ID reader.
voucherDeviceId	type: int minInclusive: -1	Device identifier for the voucher.

Example

```
<IssueVoucherRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
<voucherDeviceId>1</voucherDeviceId><idReaderDeviceId>1</idReaderDeviceId>
<creditType>CASHABLE</creditType></IssueVoucherRequest>
```

Response: IssueVoucherResponse

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.
propertyLine1	type: string	First line of property information printed on voucher.
propertyLine2	type: string	Second line of property printed on voucher.
propertyName	type: string	Name of property printed on voucher.
title	type: string	Voucher profile (titlePromo, titleCash or titleNonCash).

Attribute	Restrictions	Description
validationId	type: string minLength: 18 maxLength: 18	18-digit number sequence used to identify a voucher.
voucherAmount	type: long minInclusive: 0 maxInclusive: 999999999999999	Amount of the voucher, in millicents.
voucherAuthenticationId	type: string	Voucher authentication identifier.

Example

```
<IssueVoucherResponse 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>All
is OK</message><title>Cash</title><propertyLine1>85 Keystone
Ave.</propertyLine1><propertyLine2>Reno,NV 89503</propertyLine2><propertyName>RadBlue Demo
System</propertyName><validationId>3000013273564142229</validationId>
<voucherAmount>200000</voucherAmount><voucherAuthenticationId>A93E-76F8-549F-21DA-9BC1-89AB-
887E-9CE1</voucherAuthenticationId></IssueVoucherResponse>
```

Method: KeyOff

The `Keyoff` method is used to key off a handpay.

To initiate a key off through the RST user interface, select the **Player Verbs** tab on the **SmartEGM** layout, and click **Key Off Handpay**.

HTTP Type: POST

Call Type: Synchronous

Request: KeyOffRequest

Element	Restrictions	Description
KeyOffAction	type: KeyOffAction	Type of key off to perform.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes RST to key off a handpay.

```
<KeyOffRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <keyOffAction>HANDPAY</keyOffAction>
</KeyOffRequest>
```

Method: noteDrop

The `noteDrop` method lets you perform a note drop operation for the specified note acceptor device. Note drop is a procedure that includes the note drop door opening, the stacker being removed and reinserted, and the door being closed.

To send note acceptor events through the RST user interface, select the **Player Verbs** tab on the **SmartEGM** layout, and click **Dispense Note(s)**.

HTTP Type: POST

Call Type: Synchronous

Request: NoteDropRequest

Element	Restrictions	Description
deviceId	type: int minInclusive: -1 minOccurs: 1 maxOccurs: 1	Note acceptor device that performs the note drop.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes a note drop.

```
<NoteDropRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
</NoteDropRequest>
```

Method: playPaytableGame

The `playPaytableGame` method plays a paytable game, generating either a random outcome or using a pre-determined outcome found in the outcome database.

To play a paytable game through the RST user interface, select the **Player Verbs** tab on the **SmartEGM** layout, and click **Play Paytable Game**.

HTTP Type: POST

Call Type: Asynchronous

> **Note:** The call result for this asynchronous method appears after the "Response" section. See `fetchStatus` for details on that method.

Request: PlayPaytableGameRequest

Element	Restrictions	Description
creditsToWagerCashable	type: long	Number of cashable credits to wager.
creditsToWagerNonCashable	type: long	Number of non-cashable credits to wager.
creditsToWagerPromo	type: long	Number of promotional credits to wager.
denomToBet	type: long minInclusive: -1 maxInclusive: 999999999999999	Denomination to bet.
finalHandCount	type: int	Number of final hands to include in game.
gamePlayDeviceId	type: int minInclusive: -1	Identifier of gamePlay device.
handpayKeyOffType	type: handpayKeyOffType	If handpay included, specifies key off type.
inGameDelay	type: long	A non-zero value defines how many milliseconds to wait at the end of the game. This element is used to simulate the completion of a real game.
outcomeRecord	type: outcomeRecord	Outcome record to use.
remoteKeyOffTimeOut	type: long	If remote key off included, defines the time to wait for the key off to be completed.
seed	type: long	Seed value of the random number generator used for this game. Zero (0) uses an RST-supplied seed.
winFinalSecondaryGame	type: boolean	If secondary games are included, indicates whether or not the final result is a win.
winToHandpay	type: boolean	Indicates whether or not the win is sent to a handpay.

outcomeRecord

Element	Restrictions	Description
exists	type: boolean minOccurs: 1 maxOccurs: 1	Indicates whether or not an outcome record exists.
outcomeName	type: string minOccurs: 1 maxOccurs: 1	If non-blank, name of the outcome record in the outcome database. RST looks up this outcome name in the database and uses the result.
randomOutcome	type: boolean minOccurs: 1 maxOccurs: 1	If **true**, RST uses the seed value to randomly compute the outcome.

Response: Ticket

Element	Restrictions	Description
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.
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

This request causes RST to play a paytable game.

```
<PlayPaytableGameRequest>
     <gamePlayDeviceId>1</gamePlayDeviceId>
     <denomToBet>100000</denomToBet>
     <creditsToWagerCashable>1</creditsToWagerCashable>
     <creditsToWagerPromo>0</creditsToWagerPromo>
     <creditsToWagerNonCashable>0</creditsToWagerNonCashable>
     <secondaryGameCount>0</secondaryGameCount>
     <winFinalSecondaryGame>false</winFinalSecondaryGame>
     <winToHandpay>false</winToHandpay>
     <handpayKeyOffType>G2S_localCredit</handpayKeyOffType>
     <remoteKeyOffTimeOut>600000</remoteKeyOffTimeOut>
     <inGameDelay>3000</inGameDelay>
     <seed>0</seed>
     <finalHandCount>1</finalHandCount>
     <outcomeRecord>
          <outcomeName/>
          <randomOutcome>true</randomOutcome>
          <exists>false</exists>
     </outcomeRecord>
</PlayPaytableGameRequest>
```

GamePlayResponse

Name	Value
denomId	Denomination of the bet.
totalCreditsWagered	Total number of credits wagered.
wageredAmount	Amount wagered, in millicents.
progressiveHit	**True** indicates a progressive hit. Otherwise, this value is **false**.
primaryWin	Amount of primary win, in millicents.

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>{03c6b676-3490-45e9-0008-53e01b5126f9}</ticket>
     <responseType>GamePlayResponse</responseType>
     <code>RBG_OK</code>
     <message>The game has completed.</message>
     <attributes>
     <ns2:element>
          <ns2:name>denomId</ns2:name>
          <ns2:value>100000</ns2:value>
     </ns2:element>
     <ns2:element>
```

```
          <ns2:name>totalCreditsWagered</ns2:name>
          <ns2:value>1</ns2:value>
     </ns2:element>
     <ns2:element>
          <ns2:name>wageredAmount</ns2:name>
          <ns2:value>100000</ns2:value>
     </ns2:element>
     <ns2:element>
          <ns2:name>progressiveHit</ns2:name>
          <ns2:value>false</ns2:value>
     </ns2:element>
     <ns2:element>
          <ns2:name>primaryWin</ns2:name>
          <ns2:value>0</ns2:value>
     </ns2:element>
     </attributes>
</CallResult>
```

Method: playSimpleGame

This method plays a simple game. It is asynchronous because some game play iterations can take a while to fully complete, so you should follow it with a `fetchStatus` method to determine the final status. Use this method if you need to simulate game play.

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

HTTP Type: POST

Call Type: Synchronous

> **Note:** The call result for this asynchronous method appears after the "Response" section. See `fetchStatus` for details on that method.

Request: PlaySimpleGameRequest

Attribute	Restrictions	Description
creditsToWagerCashable	type: int	Number of cashable, non-promotional credits to wager.
creditsToWagerNonCashable	type: int	Number of promotional, non-cashable credits you want to wager.
creditsToWagerPromo	type: int	Number of promotional, cashable credits you want to wager.
denomToBet	type: denomId	Value of a single credit.
gamePlayDeviceId	type: int	Game play device on which you want to play.
handpayKeyOffType	type: handpayKeyOffType	Type of handpay that should occur after game play has concluded. Method of paying handpay.
inGameDelay	type: int	Defines an in-game delay (in milliseconds) which is executed after GPE101 (Primary Game Escrow) and after GPE109 (Secondary Game Started). Use this delay to simulate complex game mechanics.
keyOffTimeOut	type: long	Time to wait for a `setRemoteKeyOff` command or a time out. If a time out occurs, the game win is paid according to the `handpayKeyOffType` attribute value.
primaryWin	type: primaryWin	Number of credits won as a result of game play. Zero credits are permissible.
progressiveHit	type: boolean	Indicates whether game play generates a progressive hit.

Attribute	Restrictions	Description
remoteKeyOffTimeOut	type: int	**DEPRECATED 03 OCT 2012 - Version 24** **Replaced by `KeyOffTimeOut`** Number of milliseconds before the EGM times out when waiting to receive a setRemoteKeyOff command.
secondaryGameCount	type: int	Number of double-or-nothing secondary games to attempt to play if the primary-win is not zero. For each secondary game play, the EGM wagers all the winnings to that point, and continues until either the amount won equals zero or the number of games to play.
winFinalSecondaryGame	type: boolean	All winnings from all secondary games are retained. If cleared, no secondary game winnings are retained.
winLevelIndex	type: int	Defines the win-level index for the progressive hit.
winToHandpay	type: boolean	Indicates whether the game win should go to handpay.

denomToBet Elements

Element	Restrictions	Description
denomId	type: long minInclusive: -1 maxInclusive: 999999999999999	Denomination identifier.

primaryWin Elements

Element	Restrictions	Description
G2sMeterValue	type: long minInclusive: 0 maxInclusive: 999999999999999	Number of credits to win.

Example

```
<PlaySimpleGameRequest
xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1"><gamePlayDeviceId>1
</gamePlayDeviceId><denomToBet>100000</denomToBet><creditsToWagerCashable>1
</creditsToWagerCashable>
<creditsToWagerPromo>0</creditsToWagerPromo><creditsToWagerNonCashable>0
</creditsToWagerNonCashable>
<primaryWin>0</primaryWin><secondaryGameCount>0
</secondaryGameCount><winFinalSecondaryGame>false
</winFinalSecondaryGame><winToHandpay>false</winToHandpay><handpayKeyOffType>G2S_
localCredit</handpayKeyOffType><keyOffTimeOut>600000</keyOffTimeOut>
<progressiveHit>false</progressiveHit><winLevelIndex>1</winLevelIndex><inGameDelay>3000
</inGameDelay></PlaySimpleGameRequest>
```

Response: Ticket

Attribute	Restrictions	Description
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.
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

```
<Ticket 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"><code>RBG_OK</code><message>Async
request submitted.</message><ticket>{6dc26648-ea9d-41cf-8089-30d7a5d2b62d}</ticket></Ticket>
```

GamePlayResponse

Name	Value
denomId	Denomination of the bet.
totalCreditsWagered	Total number of credits wagered.
wageredAmount	Amount wagered, in millicents.
progressiveHit	**True** indicates a progressive hit. Otherwise, this value is **false**.
primaryWin	Amount of primary win, in millicents.

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>{03c6b676-3490-45e9-0008-53e01b5126f9}</ticket>
     <responseType>GamePlayResponse</responseType>
     <code>RBG_OK</code>
     <message>The game has completed.</message>
     <attributes>
     <ns2:element>
          <ns2:name>denomId</ns2:name>
          <ns2:value>100000</ns2:value>
     </ns2:element>
     <ns2:element>
          <ns2:name>totalCreditsWagered</ns2:name>
          <ns2:value>1</ns2:value>
     </ns2:element>
     <ns2:element>
          <ns2:name>wageredAmount</ns2:name>
          <ns2:value>100000</ns2:value>
     </ns2:element>
     <ns2:element>
          <ns2:name>progressiveHit</ns2:name>
          <ns2:value>false</ns2:value>
     </ns2:element>
     <ns2:element>
          <ns2:name>primaryWin</ns2:name>
          <ns2:value>0</ns2:value>
     </ns2:element>
     </attributes>
</CallResult>
```

Method: progressiveGetHostInfo

The `progressiveGetHostInfo` method lets you request information from the progressive host that owns the specified progressive device.

To send this request through the RST user interface, select the **Player Verbs** tab on the **SmartEGM** layout, and click **Get Progressive Host Info**.

HTTP Type: POST

Call Type: Synchronous

Request: ProgressiveGetHostInfoRequest

Element	Restrictions	Description
deviceId	type: int minInclusive: -1 minOccurs: 1 maxOccurs: 1	Progressive device identifier.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes RST to request the current host information.

```
<ProgressiveGetHostInfoRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
</ProgressiveGetHostInfoRequest>
```

Method: redeemVoucher

This method redeems a voucher, moving credits from the voucher system to the game. This is an asynchronous operation, so you should query RST for the status of the redemption, using the fetchStatus method. Use this method to transfer funds onto the game.

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

HTTP Type: POST

Call Type: Asynchronous

> **Note:** The call result for this asynchronous method appears after the "Response" section. See `fetchStatus` for details on that method.

Request: RedeemVoucherRequest

Attribute	Restrictions	Description
IdReaderDeviceId	type: int minInclusive: -1	Identifier for the ID reader device.
noteAcceptorDeviceId	type: int minInclusive: -1	Identifier for the note acceptor device.
validationId	type: string minLength: 18 maxLength: 18	18-digit number sequence used to identify a voucher.
voucherAction	type: voucherAction	Specifies what happens to the voucher once it is inserted into the EGM (for example, "drop" means it would be sent to the note drop).
voucherDeviceId	type: int	Identifier for the voucher device.

Example

```
<RedeemVoucherRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
<voucherDeviceId>1</voucherDeviceId><idReaderDeviceId>1</idReaderDeviceId>
<noteAcceptorDeviceId>1</noteAcceptorDeviceId>
<validationId>3000013273564142 29</validationId>
<voucherAction>DROP</voucherAction></RedeemVoucherRequest>
```

Response: Ticket

Attribute	Restrictions	Description
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.
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

```
<Ticket xmlns:ns3="http://www.radblue.com/g2s/transcript2/api/schemas/v1.0."
xmlns:ns2="http://www.radblue.com/g2s/common/api/schemas/v1.0."
xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1"><code>RBG_OK</code><message>Async
request submitted.</message><ticket>{ab1a7d44-5e0a-4726-009d-6a593d90952d}</ticket></Ticket>
```

RedeemVoucherResponse

Name	Value
voucherAmount	Value of the redeemed voucher, in millicents.

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>{7dada772-d9ff-4469-80c7-469fa062060c}</ticket>
    <responseType>RedeemVoucherResponse</responseType>
    <code>RBG_OK</code>
    <message>The voucher has been redeemed.</message>
    <attributes>
         <ns2:element>
           <ns2:name>voucherAmount</ns2:name>
           <ns2:value>1000000</ns2:value>
         </ns2:element>
    </attributes>
</CallResult>
```

Method: takeOutId

This method removes the current ID from an ID reader device. Use this method if you have to stop the carded session.

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

HTTP Type: POST

Call Type: Synchronous

Request: RemoveIdRequest

Attribute	Restrictions	Description
deviceId	type: int	Device identifier.

Example

```
<RemoveIdRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
<deviceId>1</deviceId></RemoveIdRequest>
```

Response: RemoveIdResponse

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

```
<RemoveIdResponse 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>ID has been removed.</message></RemoveIdResponse>
```

Method: cabinetEvents

The `cabinetEvent` method lets you send one or more cabinet events for a specified cabinet device. Use this method to generate or clear cabinet related events. You can report as many events as you want in a request with the exception of the CABINET_FAULTS_CLEARED event. That event must always be submitted alone.

To send cabinet events through the RST user interface, select the **Device Events** tab on the **SmartEGM** layout, and click **G2S_cabinet**.

HTTP Type: POST

Call Type: Synchronous

Request: CabinetEventsRequest

Element	Restrictions	Description
eventList	type:cabinetEventList minOccur: 1 maxOccur: 12	Container for cabinet events.

cabinetEventList

Element	Restrictions	Description
event	type:cabinetEvent	Specifies cabinet events.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request shows that all three door-open events have been set.

```
<CabinetEventsRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <eventList>
         <event>AUXILIARY_DOOR_OPENED</event>
         <event>CABINET_DOOR_OPENED</event>
         <event>LOGIC_DOOR_OPENED</event>
    </eventList>
</CabinetEventsRequest>
```

This request shows that all of the cabinet events have been cleared.

```
<CabinetEventsRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <eventList>
         <event>CABINET_FAULTS_CLEARED</event>
    </eventList>
</CabinetEventsRequest>
```

Method: changeDoorState

The `changeDoorState` method lets you open and close the EGM cabinet doors (cabinet, logic or auxiliary).

To change the state of any cabinet door through the RST user interface, select the **Device Events** tab on the **SmartEGM** layout, and click **G2S_cabinet**.

HTTP Type: POST

Call Type: Synchronous

Request: ChangeDoorStateRequest

Element	Restrictions	Description
action	type: DoorAction minOccurs: 1 maxOccurs: 1	Action to take on the door.
name	type: DoorName minOccurs: 1 maxOccurs: 1	Type of cabinet door.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates an error occurred in the request; **False** indicates that the request was completed successfully.
message	type: string	Description of the result.

Example

This request causes RST to report that the cabinet door is open.

```
<ChangeDoorStateRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <name>CABINET</name>
    <action>OPEN</action>
</ChangeDoorStateRequest>
```

This request causes RST to report that the cabinet door is closed.

```
<ChangeDoorStateRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <name>CABINET</name>
    <action>CLOSE</action>
</ChangeDoorStateRequest>
```

Method: coinAcceptorEvents

The `coinAcceptorEvents` method lets you send one or more coin acceptor events for a specified coin acceptor device. Use this method to generate or clear coin acceptor events.

To send coin acceptor events through the RST user interface, select the **Device Events** tab on the **SmartEGM** layout, and click **G2S_coinAcceptor**.

HTTP Type: POST

Call Type: Synchronous

Request: CoinAcceptorEventsRequest

Element	Restrictions	Description
deviceId	type: int minInclusive: -1 minOccurs: 1 maxOccurs: 1	EGM device identifier.
eventList	type:CoinAcceptorEventList minOccur: 1 maxOccur: 1	Container for coin acceptor events.

CoinAcceptorEventList

Element	Restrictions	Description
event	type:CoinAcceptorEvent	Coin acceptor events to report.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request shows that the coin drop door open event has been set.

```
<CoinAcceptorEventsRequest>
    <deviceId>1</deviceId>
    <eventList>
         <event>COIN_DROP_DOOR_OPEN</event>
    </eventList>
</CoinAcceptorEventsRequest>
```

This request shows that all of the coin acceptor events have been cleared.

```
<CoinAcceptorEventsRequest>
    <deviceId>1</deviceId>
    <eventList>
         <event>CLEAR_ALL_FAULTS</event>
    </eventList>
</CoinAcceptorEventsRequest>
```

Method: noteAcceptorEvents

The `noteAcceptorEvents` method lets you send one or more note acceptor events for a specified note acceptor device. Use this method to generate or clear note acceptor events.

To send note acceptor events through the RST user interface, select the **Device Events** tab on the **SmartEGM** layout, and click **G2S_noteAcceptor**.

HTTP Type: POST

Call Type: Synchronous

Request: NoteAcceptorEventsRequest

Element	Restrictions	Description
deviceId	type: int minInclusive: -1 minOccurs: 1 maxOccurs: 1	Note acceptor device identifier.
eventList	type:NoteAcceptorEventList minOccur: 1 maxOccur: 12	Container for note acceptor events.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request shows that the note acceptor door open event has been set.

```
<NoteAcceptorEventsRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
    <eventList>
         <event>STACKER_DOOR_OPEN</event>
    </eventList>
</NoteAcceptorEventsRequest>
```

This request shows that all of the note acceptor events have been cleared.

```
<NoteAcceptorEventsRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
    <eventList>
         <event>CLEAR_ALL_FAULTS</event>
    </eventList>
</NoteAcceptorEventsRequest>
```

Method: PrinterEvents

The `PrinterEvents` method lets you send one or more printer event for a specified printer device. Use this method to generate printer events.

To send printer events through the RST user interface, select the **Device Events** tab on the **SmartEGM** layout, and click **G2S_printer**.

HTTP Type: POST

Call Type: Synchronous

Request: PrinterEventsRequest

Element	Restrictions	Description
deviceId	type: int minInclusive: -1 minOccurs: 1 maxOccurs: 1	Printer device identifier.
eventList	type:PrinterEvent minOccur: 1 maxOccur: 12	Container for printer events.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request shows that the paper low event has been set.

```
<PrinterEventsRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
    <eventList>
         <event>PAPER_LOW</event>
    </eventList>
</PrinterEventsRequest>
```

This request shows that all of the printer events have been cleared.

```
<PrinterEventsRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
    <eventList>
         <event>CLEAR_ALL_FAULTS</event>
    </eventList>
</PrinterEventsRequest>
```

Method: watGetAccounts

The `watGetAccounts` method asks the RST to request the WAT accounts for the current player ID from the host that owns the specified WAT device.

To request WAT accounts for a player through the RST user interface, select the **Player Verbs** tab on the **SmartEGM** layout, and click the **Wager Account Transfer** bar. Then, click **Get Accounts** under the **Options** section.

HTTP Type: POST

Call Type: Synchronous

Request: WatGetAccountsRequest (from the host)

Element	Restrictions	Description
authentication	type: string minLength: 0 maxLength: 256 minOccur: 1 maxOccur: 1	The authentication string (password or personal identification number). Send an empty string if the WAT account does not need a password or the player did not provide a password.
deviceId	type: int minOccur: 1 maxOccur: 1	Identifier of the WAT device to use.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes RST to request the WAT accounts of the current player using the authentication string **PASSWORD**.

```
<WatGetAccountsRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
```

```
    <deviceId>1</deviceId>
    <authentication>PASSWORD</authentication>
</WatGetAccountsRequest>
```

Method: watGetBalance

The `watGetBalance` method requests from the host the WAT account balance(s) for the specified WAT account using the current player ID.

To request WAT account balances for a player through the RST user interface, select the **Player Verbs** tab on the **SmartEGM** layout, and click the **Wager Account Transfer** bar. Then, click **Get Balance** under the **Options** section.

HTTP Type: POST

Call Type: Synchronous

Request: WatGetBalanceRequest

Element	Restrictions	Description
accountId	type: string minLength: 0 maxLength: 32 minOccur: 1 maxOccur: 1	WAT account identifier
authentication	type: string minLength: 0 maxLength: 256 minOccur: 1 maxOccur: 1	The authentication string (password). Send an empty string if the WAT account does not need a password or the player did not provide a password.
deviceId	type: int minOccur: 1 maxOccur: 1	Identifier of the WAT device to use.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes RST to request the balance of WAT account **A-1111** with the authentication string **PASSWORD**.

```
<WatGetBalanceRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
    <accountId>A-1111</accountId>
    <authentication>PASSWORD</authentication>
</WatGetBalanceRequest>
```

Method: watGetKeyPair

The `watGetKeyPair` method requests from the host a new WAT key pair for the specified device.

To request a new WAT key pair through the RST user interface, select the **Player Verbs** tab on the **SmartEGM** layout, and click the **Wager Account Transfer** bar. Then, click **Get Key Pair** under the **Options** section.

HTTP Type: POST

Call Type: Synchronous

Request: WatGetBalanceRequest

Element	Restrictions	Description
deviceId	type: int minOccur: 1 maxOccur: 1	Identifier of the WAT device to use.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes RST to request a new WAT key pair from the host associated with WAT **device 1**.

```
<WatGetKeyPairRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
</WatGetKeyPairRequest>
```

Method: watToEgm

The `watToEgm` method initiates a transfer of funds from the WAT host to RST.

To request WAT account balances for a player through the RST user interface, select the **Player Verbs** tab on the **SmartEGM** layout, and click the **Wager Account Transfer** bar. Then, click **Transfer Funds** under the **Options** section.

HTTP Type: POST

Call Type: Asynchronous

> **Note:** The call result for this asynchronous method appears after the "Response" section. See `fetchStatus` for details on that method.

Request: WatToEgmRequest

Element	Restrictions	Description
accountId	type: string minLength: 0 maxLength: 32 minOccur: 1 maxOccur: 1	WAT account identifier
cashableRequest	type: long minInclusive: 0 maxInclusive: 999999999999999 minOccur: 1 maxOccur: 1	Cashable amount to transfer.
deviceId	type: int minOccur: 1 maxOccur: 1	Identifier of the WAT device to use.
idReaderDeviceId	type: int minInclusive: -1 minOccur: 1 maxOccur: 1	ID reader device identifier.
nonCashableRequest	type: long minInclusive: 0 maxInclusive: 999999999999999 minOccur: 1 maxOccur: 1	Non-cashable amount to transfer.
promoRequest	type: long minInclusive: 0 maxInclusive: 999999999999999	Promotion amount to transfer.

Element	Restrictions	Description
	minOccur: 1 maxOccur: 1	

Response: Ticket

Element	Restrictions	Description
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.
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

This request causes RST to transfer **$10.00** to the credit meter.

```
<WatToEgmRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
    <idReaderDeviceId>1</idReaderDeviceId>
    <accountId>A-1111</accountId>
    <cashableRequest>1000000</cashableRequest>
    <promoRequest>0</promoRequest>
    <nonCashableRequest>0</nonCashableRequest>
</WatToEgmRequest>
```

WatToEgmResponse

Name	Value
accountId	Identifier of the account from which the funds will be transferred.
watDirection	Constant value **TO_EGM**.
cashableRequest	Amount of cashable funds to transfer, in millicents.
promoRequest	Amount of promotional funds to transfer, in millicents.
nonCashableRequest	Amount of non-cashable funds to transfer, in millicents.

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>{d25c206f-8f8c-4e7b-8038-ba18d893e074}</ticket>
    <responseType>WatToEgmResponse</responseType>
    <code>RBG_OK</code>
    <message>Transfer has completed successfully.</message>
    <attributes>
         <ns2:element>
           <ns2:name>nonCashableRequest</ns2:name>
           <ns2:value>0</ns2:value>
         </ns2:element>
         <ns2:element>
           <ns2:name>accountId</ns2:name>
           <ns2:value>A-1111</ns2:value>
         </ns2:element>
         <ns2:element>
           <ns2:name>cashableRequest</ns2:name>
           <ns2:value>2000000</ns2:value>
         </ns2:element>
         <ns2:element>
           <ns2:name>promoRequest</ns2:name>
           <ns2:value>0</ns2:value>
         </ns2:element>
         <ns2:element>
           <ns2:name>watDirection</ns2:name>
           <ns2:value>TO_EGM</ns2:value>
         </ns2:element>
    </attributes>
</CallResult>
```

Method: watToHost

The `watToHost` method initiates a transfer of funds from the WAT host to RST.

To request WAT account balances for a player through the RST user interface, select the **Player Verbs** tab on the **SmartEGM** layout, and click the **Wager Account Transfer** bar. Then, click **Transfer Funds** under the **Options** section.

HTTP Type: POST

Call Type: Asynchronous

> **Note:** The call result for this asynchronous method appears after the "Response" section. See `fetchStatus` for details on that method.

Request: WatToHostRequest

Element	Restrictions	Description
accountId	type: string minLength: 0 maxLength: 32 minOccur: 1 maxOccur: 1	WAT account identifier
cashableRequest	type: long minInclusive: 0 maxInclusive: 999999999999999 minOccur: 1 maxOccur: 1	Cashable amount to transfer.
deviceId	type: int minOccur: 1 maxOccur: 1	Identifier of the WAT device to use.
idReaderDeviceId	type: int minInclusive: -1 minOccur: 1 maxOccur: 1	ID reader device identifier.
nonCashableRequest	type: long minInclusive: 0 maxInclusive: 999999999999999 minOccur: 1 maxOccur: 1	Non-cashable amount to transfer.
promoRequest	type: long minInclusive: 0 maxInclusive: 999999999999999	Promotion amount to transfer.

Element	Restrictions	Description
	minOccur: 1 maxOccur: 1	

Response: Ticket

Element	Restrictions	Description
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.
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

This request causes RST to transfer **$10.00** from the credit meter to the WAT host.

```
<WatToHostRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
    <idReaderDeviceId>1</idReaderDeviceId>
    <accountId>A-1111</accountId>
    <cashableRequest>1000000</cashableRequest>
    <promoRequest>0</promoRequest>
    <nonCashableRequest>0</nonCashableRequest>
</WatToHostRequest>
```

WatToHostResponse

Name	Value
accountId	Identifier of the account from which the funds will be transferred.
watDirection	Constant value **FROM_EGM**.
cashableRequest	Amount of cashable funds to transfer, in millicents.
promoRequest	Amount of promotional funds to transfer, in millicents.
nonCashableRequest	Amount of non-cashable funds to transfer, in millicents.

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>{8077a701-ac0c-4475-80a5-a40f5b329295}</ticket>
    <responseType>WatToHostResponse</responseType>
    <code>RBG_OK</code>
    <message>Transfer has completed successfully.</message>
    <attributes>
         <ns2:element>
           <ns2:name>nonCashableRequest</ns2:name>
           <ns2:value>0</ns2:value>
         </ns2:element>
         <ns2:element>
           <ns2:name>accountId</ns2:name>
           <ns2:value>A-1111</ns2:value>
         </ns2:element>
         <ns2:element>
           <ns2:name>cashableRequest</ns2:name>
           <ns2:value>2000000</ns2:value>
         </ns2:element>
         <ns2:element>
           <ns2:name>promoRequest</ns2:name>
           <ns2:value>0</ns2:value>
         </ns2:element>
         <ns2:element>
           <ns2:name>watDirection</ns2:name>
           <ns2:value>FROM_EGM</ns2:value>
         </ns2:element>
    </attributes>
</CallResult>
```

Method: getCountdownOverride

The `getCountdownOverride` method tells RST to send the `getCountdownOverride` G2S command to the player host.

To send the `getCountdownOverride` command through the RST user interface, select the **Send Commands** tab on the **SmartEGM** layout, and click **Send getCountdownOverride**.

HTTP Type: POST

Call Type: Synchronous

Request: PlayerGetCountdownOverrideRequest

Element	Restrictions	Description
deviceId	type: int minInclusive: -1 minOccurs: 1 maxOccurs: 1	Player device identifier.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes RST to request the current countdown override from the player host.

```
<PlayerGetCountdownOverrideRequest
xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
</PlayerGetCountdownOverrideRequest>
```

Method: getMcastKeyUpdate

The `getMcastKeyUpdate` method tells RST to request a new set of multicast keys from the host.

To request new multicast keys through the RST user interface, select the **Send Commands** tab on the **SmartEGM** layout, and click **Send getMcastKeyUpdate**.

HTTP Type: POST

Call Type: Synchronous

Request: GetMcastKeyUpdateRequest

Element	Restrictions	Description
deviceId	type: int minOccur: 1 maxOccur: 1	Identifier of the communications device in use.
multicast	type: string minOccur: 1 maxOccur: 1	Identifier of the multicast group for which a new key is requested.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes RST to request new multicast keys for the multicast group `RBG_bonus`.

```
<GetMcastKeyUpdateRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
    <multicastId>RBG_Bonus</multicastId>
</GetMcastKeyUpdateRequest>
```

Method: sendCommsHostList

The `sendCommsHostList` method sends a `commsHostList` command to the host that owns the commConfig device.

To send the `commsHostList` command through the RST user interface, select the **Send Commands** tab on the **SmartEGM** layout, and click **Send CommsHostList**.

HTTP Type: POST

Call Type: Synchronous

Request: SendCommsHostListRequest

Element	Restrictions	Description
deviceId	type: int minInclusive: -1 minOccurs: 1 maxOccurs: 1	commConfig device identifier.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes RST to send the `commHostList` command to the host that owns the specified device.

```
<SendCommsHostListRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <deviceId>1</deviceId>
</SendCommsHostListRequest>
```

Method: sendOptionList

The `sendOptionList` method sends an `optionList` command to the host as a G2S request, for the specified class, device or option group.

To send the `optionList` command through the RST user interface, select the **Send Commands** tab on the **SmartEGM** layout, and click **Send Option List**.

HTTP Type: POST

Call Type: Synchronous

Request: SendOptionListRequest

Element	Restrictions	Description
deviceClass	type: deviceClass minInclusive: -1 minOccurs: 1 maxOccurs: 1	Device class of the device(s) to select when collecting the options to send. The value must be a valid G2S device class value. The special value **G2S_all** is used to indicate all device classes.
deviceId	type: int minInclusive: -1 minOccurs: 1 maxOccurs: 1	Device identifier of the device(s) to select when collecting the options to send. The value must be a valid device identifier or **-1**, which specifies all devices.
optionConfigDeviceId	type: int minOccurs: 1 maxOccurs: 1	Device identifier of the optionConfig device to which the request should be sent.
optionGroupId	type: string minInclusive: 5 minOccurs: 1 maxOccurs: 1	Identifier of the options that should be collected. The value must be a valid Option Group ID value. The special value**G2S_all** indicates all option group identifiers.
optionId	type: string minLength: 5 minOccurs: 1 maxOccurs: 1	This Option ID of the options that should be collected. This has to be a valid Option ID value. The special value **G2S_all** returns all option identifiers.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes RST to send all of the current options, for all devices, to the host associated with Option Config device **1**.

```
<SendOptionListRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <optionConfigDeviceId>1</optionConfigDeviceId>
    <deviceClass>G2S_all</deviceClass>
    <deviceId>-1</deviceId>
    <optionGroupId>G2S_all</optionGroupId>
    <optionId>G2S_all</optionId>
</SendOptionListRequest>
```

This request causes RST to send all of the current options, for all of the Game Play devices, to the host associated with Option Config device **1**.

```
<SendOptionListRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <optionConfigDeviceId>1</optionConfigDeviceId>
    <deviceClass>G2S_gamePlay</deviceClass>
    <deviceId>-1</deviceId>
    <optionGroupId>G2S_all</optionGroupId>
    <optionId>G2S_all</optionId>
</SendOptionListRequest>
```

About Negative Testing

Negative testing lets you see how a host system reacts to circumstances outside of its defined parameters (in other words, how the host handles unforeseen events).

You can do two types of negative testing through RST REST: *message modification* and *send a custom message*.

Modify Messages

Message modification lets you define a list of changes that RST will make to outgoing G2S messages. Message changes can include changing an attribute, deleting an attribute or adding a new attribute.

RST maintains a first-in, first-out list of modifications. For each G2S message RST sends, RST checks whether the message matches a modification in the list. As the RST matches modifications, it decrements the modification's count. Note that if the count is **-1**, RST will always perform the specified modification. When the modification's count reaches zero, that modification is removed from list of modifications.

When RST processes a message, it first iterates over the entire list of modifications, looking for all modifications that modify the `g2s:g2sBody` element. Since there can more than one G2S command in one message, RST checks all of the class-level elements (child elements of the `g2s:g2sBody` element) next. For each class-level element, RST checks the entire list of modifications looking for matches. When a modification matches, RST either sets (SET) an attribute value (adds if not present) or removes (REMOVE) an attribute. For example:

Initial List of Modifications

ID	Action	Element	Attribute Namespace	Attribute Name	Attribute Value	Count
1	SET	wat	http://www.radblue.com/	special		-1
2	REMOVE	g2sBody	http://www.gamingstandards.com/g2s/schemas/v1.0.3	dateTimeSent		1
3	SET	g2sBody	http://www.gamingstandards.com/g2s/schemas/v1.0.3	egmId	Name	2
4	SET	eventReport	http://www.gamingstandards.com/g2s/schemas/v1.0.3	eventId	ALPHA	1
5	REMOVE	commsOnLine	http://www.gamingstandards.com/g2s/schemas/v1.0.3	egmLocation		1

Method: fetchAllModifications

The `fetchAllModifications` method is used to fetch the current set of message modifications (which may be empty). Use this method to fetch all of the currently configured message modification definitions.

HTTP Type: POST

Call Type: Synchronous

Request: *Not Applicable*

There is no request information for this method.

Response: GetModificationsResponse

Element	Restrictions	Description
modification	type: ModificationDefinition minOccurs: 1 maxOccurs: unbounded	Message modification definition.

ModificationDefinition

Element	Restrictions	Description
action	type: ModificationAction minOccur: 1 maxOccur: 1	Action to take for the definition.
attributeName	type: string minLength: 0 maxLength: 256 minOccur: 1 maxOccur: 1	Name of the attribute for adding and searching.
attributeNamespaceURI	type: string minLength: 1 maxLength: 256 minOccur: 1 maxOccur: 1	Namespace of the attribute, for both adding and searching. When you use `<ANY-NAMESPACE>`, all attributes are searched regardless of namespace. This value is illegal if the action element is **SET** .
attributeValue	type: string minLength: 0 maxLength: 512	If the `action` element is SET, this is the new value of the attribute, whether it is added or replaced. If the `action` element is REMOVE,

Element	Restrictions	Description
	minOccur: 1 maxOccur: 1	the value of this element is ignored.
count	type: int minInclusive: 1	Controls how many times this definition is applied. When the count reaches zero, it is deleted by RST.
elementName	type: string minLength: 1 maxLength: 64 minOccur: 1 maxOccur: 1	The name of the element to search for. The special value **<ANY-CLASS>** matches any class level element, regardless of namespace. The special value **<ANY-COMMAND>** matches any command level element, regardless of namespace.
id	type: string minLength: 1 maxLength: 64 minOccur: 1 maxOccur: 1	Modification identifier. This value does not have to be unique.

Method: modificationsAddList

The `modificationsAddList` method lets you add one or more message modification definitions to the RST transport layer. These modifications are immediately active once the method returns.

Use this method to configure the message modification mechanism in the RST transport layer.

HTTP Type: POST

Call Type: Synchronous

Request: AddModificationsRequest

Element	Restrictions	Description
modification	type: ModificationDefinition minOccurs: 1 maxOccurs: 1	Message modification definition to add.

ModificationDefinition

Element	Restrictions	Description
action	type: ModificationAction minOccur: 1 maxOccur: 1	Action to take for the definition.
attributeName	type: string minLength: 0 maxLength: 256 minOccur: 1 maxOccur: 1	Name of the attribute for adding and searching.
attributeNamespaceURI	type: string minLength: 1 maxLength: 256 minOccur: 1 maxOccur: 1	Namespace of the attribute, for both adding and searching. When you use `<ANY-NAMESPACE>`, all attributes are searched regardless of namespace. This value is illegal if the action element is **SET** .
attributeValue	type: string minLength: 0 maxLength: 512 minOccur: 1 maxOccur: 1	If the `action` element is SET, this is the new value of the attribute, whether it is added or replaced. If the `action` element is REMOVE, the value of this element is ignored.
count	type: int minInclusive: 1	Controls how many times this definition is applied. When the count reaches zero, this

Element	Restrictions	Description
		modification record is deleted by RST.
elementName	type: string minLength: 1 maxLength: 64 minOccur: 1 maxOccur: 1	The name of the element to search for. The special value **<ANY-CLASS>** matches any class level element, regardless of namespace. The special value **<ANY-COMMAND>** matches any command level element, regardless of namespace.
id	type: string minLength: 1 maxLength: 64 minOccur: 1 maxOccur: 1	Modification identifier. This value does not *have to* be unique, but it *should be* to more easily manage active modification records.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request adds a new modification record that removes the *hostId* attribute from the `g2sBody` element of the next two G2S messages sent by RST.

```
<AddModificationsRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <modification>
        <id>ID-1</id>
        <action>REMOVE</action>
        <count>2</count>
        <elementName>g2sBody</elementName>
        <attributeNamespaceURI>http://www.gamingstandards.com/g2s/schemas/v1.0.3</attribute
        NamespaceURI>
        <attributeName>hostId</attributeName>
        <attributeValue></attributeValue>
    </modification>
</AddModificationsRequest>
```

Method: modificationsClearAll

The `modificationsClearAll` method lets you clear all current message modification definitions. These modifications are immediately active once the method returns.

Use this method at the beginning of negative testing that requires message modification, guaranteeing that the definitions you load next are the only definitions in use.

HTTP Type: POST

Call Type: Synchronous

Request: *Not Applicable*

There is no request information for this method.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Method: modificationsDeleteList

The `modificationsDeleteList` method lets you delete one or more message modification definitions.

HTTP Type: POST

Call Type: Synchronous

Request: DeleteModificationsRequest

Element	Restrictions	Description
id	type: string minLength: 1 maxLength: 64	Message modification definitions to delete (this element can be repeated).

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes RST to delete the message modification definitions with the identifier of **ID-1** and **ID-2**.

```
<DeleteModificationsRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <id>ID-1</id>
    <id>ID-2</id>
</DeleteModificationsRequest>
```

Send Custom Messages

You can send a custom message to test the SOAP layer and whether the host system can handle missing required values or illegal values by verifying that the host generated the correct error in response. You can change the following elements of any G2S message:

- **EGM ID for SOAP** – EGM identifier to put in the SOAP message. If there is no value, the current EGM ID is used. This is normally a fixed value.
- **Fix Up** – Indicates whether RST should add valid XML around a custom message or send it exactly as entered.
- **Host ID** – Identifier of host system to which you are sending the message.
- **Host ID for SOAP** – Host identifier to put in the SOAP message. If there is no value, use the current host ID. This is normally a fixed value.
- **Message XML** – Message XML that is changed if Fix Up value is **true**.

About Fix Up Values

Optionally, RST can modify, insert or delete (*fix up)* any child element of `g2s:g2sBody` by assuming that the element is a class-level attribute. With fix up, RST modifies the three attributes in `g2s:g2sBody` as well as the class-level attributes of the G2S command. For each attribute:

- If a value is supplied by the user (see **Present** column below), RST uses that value.
- If a value is not supplied by the user, RST uses the missing value.
- If the value is **-1**, the required attribute is removed from the message (missing).

For example:

Fix Up Rules

Attribute	Required	Present	Missing	-1
g2s:hostId	Yes	Use value	Supply argument value	Remove attribute
g2s:egmId	Yes	Use value	Supply current EGM	Remove attribute
g2s:dateTimeSent	Yes	Use value	Supply current date and time	Remove attribute
g2s:deviceId	Yes	Use value		Remove attribute
g2s:dateTime	Yes	Use value	Supply current date and time	Remove attribute
g2s:commandId	Yes	Use value	Supply next command ID for the specified host in the method arguments.	Remove attribute
g2s:sessionType	No	Use value	Assume sessionType is G2S_request	Remove attribute
g2s:sessionId	No	Use value	If session type is G2S_request, add	Remove attribute

Attribute	Required	Present	Missing	-1
			next session Id for the specified host in the message arguments.	
g2s:sessionRetry	No	Use value	Ignore	Remove attribute*
g2s:timeToLive	No	Use value	Ignore	Remove attribute*
g2s:sessionMore	No	Use value	Ignore	Remove attribute*
g2s:errorCode	No	Use value	Ignore	Remove attribute*
g2s:errorText	No	Use value	Ignore	Remove attribute*

* This value is not needed because the missing attribute will be ignored.

The following is a Send Custom Message example to an EGM with the identifier **RBG_1234**:

- **Host ID** = 1
- **Fix Up** = True
- **EGM ID for SOAP** = [blank]
- **Host ID for SOAP** = [blank]
- **Message** =

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<g:g2sMessage xmlns:g="http://www.gamingstandards.com/g2s/schemas/v1.0.3">
    <g:g2sBody g:hostId="2">
         <r:pebbleDispenser xmlns:r="http://www.radblue.com/schema/pebble"
             g:deviceId="1">
           <r:dropPebble r:count="15"/>
         </r:pebbleDispenser>
         <g:communications g:deviceId="1"
             g:sessionId="12000105"
             g:sessionType="G2S_response">
          <g:keepAliveAck/>
         </g:communications>
    </g:g2sBody>
</g:g2sMessage>
```

When the above message is sent, RST generates a SOAP message with:

- **g2sEgmId** = RBG_1234
- **g2sHostId** = 1
- **g2sRequest** =

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<g:g2sMessage xmlns:g="http://www.gamingstandards.com/g2s/schemas/v1.0.3">
    <g:g2sBody g:dateTimeSent="2012-02-21-T12:50:43.790-08:00"
         g:egmId="RBG_1234" g:hostId="2">
         <r:pebbleDispenser xmlns:r="http://www.radblue.com/schema/pebble"
             g:deviceId="1"
             g:commandId="1726"
           g:dateTime="2012-02-21-T12:50:43.765-08:00"
             g:sessionId="17293">
           <r:dropPebble r:count="15"/>
         </r:pebbleDispenser>
         <g:communications g:deviceId="1"
             g:sessionId="12000105"
             g:sessionType="G2S_response"
           g:commandId="1727">
           <g:keepAliveAck/>
         </g:communications>
    </g:g2sBody>
</g:g2sMessage>
```

Method: sendMessage

The `sendMessage` method sends a G2S command to a host. With this method, you can override the SOAP layer arguments, which allows you to craft illegal SOAP requests.

Use this method to perform negative tests of the host's ability to handle SOAP parameters.

HTTP Type: POST

Call Type: Synchronous

Request: SendMessageRequest

Element	Restrictions	Description
fixUp	type: boolean minOccur: 0 maxOccur: 1	If the value of this element is **true**, the message rules are applied to the content of the message element. If the value of the element is **false**, the message rules are not applied. Use a **false** value if you are sending *invalid* XML.
hostId	type: long minOccur: 1 maxOccur: 1	Host identifier.
message	type: string minOccur: 1 maxOccur: 1	Actual message sent.
soapEgmId	type: egmId minOccur: 0 maxOccur: 1	Value of the SOAP EGM ID to pass in the SOAP stack. If the value is missing, the proper value is used.
soapHostId	type: string minLength: 1 maxLength: 8 minOccur: 0 maxOccur: 1	Value of the SOAP Host ID to pass in the SOAP stack. If the value is missing, the proper value is used.

Response: SimpleCallResult

Element	Restrictions	Description
error	type: boolean	**True** indicates that an error occurred while processing the request; **False** indicates that the request was completed successfully.
message	type: string	Regardless of the value of the error element, the `message` element describes the outcome of the method call.

Example

This request causes RST to send a standard G2S keep alive command using the current SOAP Host ID and SOAP EGM ID.

```
<SendMessageRequest xmlns="http://www.radblue.com/g2s/rst/api/schemas/v1.0.1">
    <hostId>1</hostId>
    <fixUp>true</fixUp>
    <message>
         <g:g2sMessage xmlns:g="http://www.gamingstandards.com/g2s/schemas/v1.0.3">
             <g:g2sBody g:hostId="1">
               <g:communications g:deviceId="1">
             <g:keepAlive />
           </g:communications>
           </g:g2sBody>
         </g:g2sMessage>
    </message>
</SendMessageRequest>
```

cabinetEvent

Enumeration	Description
AUXILIARY_DOOR_CLOSED	Auxiliary door has been closed.
AUXILIARY_DOOR_OPENED	Auxiliary door has been opened.
BACKUP_BATTERY_LOW	Backup battery is low.
CABINET_DOOR_CLOSED	Cabinet door has been closed.
CABINET_DOOR_OPENED	Cabinet door has been opened.
CABINET_FAULTS_CLEARED	Cabinet faults have been cleared.
CASH_OUT	Cash out.
EGM_POWER_LOST	EGM has lost power.
EGM_POWER_UP	EGM has powered up.
EGM_STATE_AUDIT_MODE	EGM is in audit mode.
EGM_STATE_DEMO_MODE	EGM is in demonstration mode.
EGM_STATE_OPERATOR_DISABLED	EGM has been disabled by the operator.
EGM_STATE_OPERATOR_LOCKED	EGM has been locked by the operator.
EGM_STATE_OPERATOR_MODE	EGM is in operator mode.
EXIT_OPERATOR_MENU	EGM has exited the operator menu.
GENERAL_CABINET_TILT	General cabinet tilt.
GENERAL_MEMORY_FAILURE	General memory failure.
HARD_METERS_DISCONNECTED	Hard meters have been disconnected.
HARD_METERS_RECONNECTED	Hard meters have been reconnected.
LOGIC_DOOR_CLOSED	Logic door has been closed.
LOGIC_DOOR_OPENED	Logic door has been opened.
NVM_FAILURE	Non-volatile memory failure.
OPERATOR_RESET_CABINET	Cabinet reset by operator.
OPERATOR_CHANGED_CABINET_CONFIG	Cabinet configuration changed by operator.
POWER_OFF_AUXILIARY_DOOR_OPEN	Auxiliary door was opened while the EGM was powered off.
POWER_OFF_CABINET_DOOR_OPEN	Cabinet door was opened while the EGM was powered off.
POWER_OFF_LOGIC_DOOR_OPEN	Logic door was opened while the EGM was powered off.
SERVICE_LAMP_OFF	Service lamp has been turned off.
SERVICE_LAMP_ON	Service lamp has been turned on.
VIDEO_DISPLAY_ERROR	Error in video display.

CoinAcceptorEvent

Enumeration	Description
ACCEPTOR_FAULT	Coin acceptor device error.
ACCEPTOR_JAMMED	Coin acceptor device is jammed.
CLEAR_ALL_FAULTS	Clear all errors for the coin acceptor device.
COIN_ACCEPTOR_CONNECTED	Coin acceptor device is connected.
COIN_DROP_DOOR_OPEN	Coin drop door has been opened.
COMPONENT_FAULT	Component error has been detected in the device.
DIVERTER_FAULT	Diverter error has been detected in the device.
FIRMWARE_FAULT	Firmware error has been detected in the device.
ILLEGAL_ACTIVITY_DETECTED	Illegal activity has been detected in the device.
LOCKOUT_MALFUNCTION	Lockout malfunction has been detected in the device.
MECHANICAL_FAULT	Error in the mechanical components of the device.
NON-VOLATILE_MEMORY_FAULT	Error in the non-volatile memory of the device.
OPTICAL_FAULT	Error in the optical component of the device.
COIN_ACCEPTOR_DISCONNECTED	Coin acceptor device has been disconnected from the EGM.
COIN_DROP_DOOR_CLOSED	Coin drop door has been closed.
OPERATOR_CHANGED_COIN_ACCEPTOR_CONFIG	Coin acceptor device configuration has been changed by the operator.

coinAction Enumeration

Enumeration	Description
drop	Inserted coin(s) go to the coin drop.
hopper	Inserted coin(s) go to the coin hopper.

creditType Enumeration

Enumeration	Description
cashable	Cashable funds.
non-cashable	Non-Cashable funds.
promo	Promotional funds.

deviceClass

Enumeration	Description
pattern	[A-Z0-9]{3}_[-~]{1,60}

doorAction

Enumeration	Description
CLOSE	Close the cabinet door.
OPEN	Open the cabinet door.

DoorName

Enumeration	Description
AUXILIARY	Auxiliary door.
CABINET	Main cabinet door.
LOGIC	Logic door.

egmId

Enumeration	Description
pattern	[A-Z 0-9] {3})_[-~]{1,28}

handpayKeyOffType

Enumeration	Description
G2S_cancelled	Cancel handpay keyoff request.
G2S_localCredit	Keyoff handpay at EGM to local credit meter.
G2S_localHandpay	Keyoff handpay at EGM.
G2S_localVoucher	Keyoff handpay at EGM to voucher.
G2S_localWat	Keyoff handpay at EGM to WAT (Wagering Account Transfer).
G2S_remoteCredit	Keyoff handpay to local credit meter from a remote system.
G2S_remoteHandpay	Keyoff handpay from a remote system.
G2S_remoteVoucher	Keyoff handpay to a voucher from a remote system.
G2S_remoteWat	Keyoff handpay to WAT from a remote system.
G2S_unknown	Handpay keyoff method is not known.

handpayType

Enumeration	Description
G2S_bonusPay	Bonus handpay.
G2S_cancelCredit	Cancel credit handpay.
G2S_gameWin	Game win handpay.

KeyOffAction

Enumeration	Description
HANDPAY	Key off game to a local handpay.
CREDIT	Key off game to the credit meter.
VOUCHER	Key off game to a voucher.
WAT	Key off game to a WAT account.
REMOTE	Key off game using a remote server. The method will wait for the `remote-key-off-timeout` period. If the remote key off is not received in that period, the method will fail.

ModificationAction

Enumeration	Description
SET	Apply modification definition(s).
REMOVE	Remove modification definition(s).

noteAcceptorEvent

Enumeration	Description
ACCEPTOR_FAULT	Note acceptor error.
ACCEPTOR_JAMMED	Note acceptor jammed.
CLEAR_ALL_FAULTS	Clear all errors for note acceptor device.
COMPONENT_FAULT	General note acceptor device error.
FIRMWARE_FAULT	Error in firmware for the note acceptor device.
ILLEGAL_ACTIVITY_DETECTED	Illegal activity detected in the note acceptor device.
MECHANICAL_FAULT	Error in one of the mechanical components of the note acceptor device.
NON_VOLATILE_MEMORY_FAULT	Error in the non-volatile memory of the note acceptor device.
NOTE_ACCEPTOR_CONNECTED	Note acceptor has been connected.
OPTICAL_FAULT	Error in the optical component of the note acceptor device.
STACKER_DOOR_OPEN	Note acceptor's stacker door has been opened.
STACKER_FAULT	General error in the note acceptor's stacker.
STACKER_FULL	Note acceptor stacker is full.
STACKER_JAMMED	Note acceptor stacker is jammed.
STACKER_NEARLY_FULL	Note acceptor stacker is close to capacity.
STACKER_REMOVED	Note acceptor stacker has been removed.
VALIDATOR_TOTALS_RESET	Bill validator totals have been reset.
NOTE_ACCEPTOR_DISCONNECTED	Note acceptor device has been disconnected.
OPERATOR_CHANGED_NOTE_ ACCEPTOR_CONFIG	Operator has changed the note acceptor configuration.
STACKER_DOOR_CLOSED	Note acceptor stacker door has been closed.
STACKER_INSERTED	Note acceptor stacker has been inserted.

noteAction

Enumeration	Description
dispenser	Note is sent to the note dispenser.
drop	Note is sent to the note drop.
reject	Note is rejected by the EGM.
return	Note is returned by the EGM.

PrinterEvent

Enumeration	Description
CLEAR_ALL_FAULTS	Clear all printer device errors.
COMPONENT_FAULT	General printer device errors.
FIRMWARE_FAULT	Error in the printer device's firmware.
MECHANICAL_FAULT	Error in the printer device's mechanical components.
NON_VOLATILE_MEMORY_FAULT	Error in the printer device's non-volatile memory.
OPTICAL_FAULT	Error in the printer device's optical component.
PAPER_EMPTY	Printer is out of paper.
PAPER_JAM	Printer has a paper jam.
PAPER_LOW	Printer paper is low.
PRINT_HEAD_OPEN	Print device's print head is open.
PRINTER_CHASSIS_OPEN	Print device's print chassis is open.
PRINTER_CONNECTED	Printer has been connected.
OPERATOR_CHANGED_PRINTER_CONFIG	Operator has changed the printer device's configuration.
PRINT_HEAD_CLOSED	Printer device's print head has been closed.
PRINTER_CHASSIS_CLOSED	Printer device's chassis has been closed.
PRINTER_DISCONNECTED	Printer device has been disconnected from the EGM.

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.

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.

voucherAction

Enumeration	Description
drop	Indicates that the voucher goes to the note drop.
reject	Indicates that the voucher has been rejected by the EGM.
return	Indicates that the voucher has been returned to the player by the EGM.

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

D

E

F

G

H

I

J

K

R

S

T

U

V

W

X