radblue

Tiger Scripting Reference

10 DEC 2013 - Version 36

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 Tiger Scripting Language

Tiger is a scripting language developed specifically for use with RadBlue tools. Tiger scripts automate testing in RadBlue EGM simulators - either the RadBlue System Tester (RST)or RadBlue Load Tester (RLT). Tiger scripting allows you to perform regression and other extended testing on your G2S-driven applications.

How Does Tiger Work?

With Tiger, human actions (causing a series of actions in our tools) and GSA verbs can be easily expressed. Verbs denote high-level actions, such as inserting notes into an Electronic Gaming Machine (EGM), expressed in the verb `Human.insertNote`. You then specify the quantity, denomination, currency, and where the notes go (to the drop or the dispenser). When executed, this verb causes appropriate events to be sent, and meters and logs to be updated.

Assumptions

Tiger is intended for non-programmers, it is easy to use yet flexible enough to allow you to create robust test scenarios.

While it is helpful to have some general knowledge of scripting languages, it is not required to customize Tiger scripts. However, a good working knowledge of the Gaming Standards Association's (GSA) Game To System (G2S) protocol is required. The GSA G2S Message Protocol contains information on all G2S commands.

Tiger Script Usage Notes

- For GSA commands, see the G2S Message Protocol, available on the Gaming Standards Association's web site.
- A default attribute value of "-2" means that the first device is used. This value is often used as the default value for appropriate attributes.
- When no default value is assigned to an attribute, the attribute requires that you provide a value. If no value is provided, the script will fail.

Walk Through a Tiger Script

The following sample script describes the parts of a Tiger script to help you understand what needs to go into your custom script.

1. The first line of the file must contain standard XML information:

   ```
   <?xml version="1.0" encoding="UTF-8" ?>
   ```

2. Enter the namespaces that will be used in the script. For example:

   ```
   <tiger:tiger
   xmlns:tiger="http://www.radblue.com/g2s/egm-data-
   model/schemas/v1.0.0/"
   xmlns:core="http://www.radblue.com/tiger/core/schemas/v1.0.0/">
   ```

   See Prefix Namespace Bindings for information on selecting the correct namespace.

3. Create script metadata. Metadata provides information about the script and is displayed on the Tiger Scripting user interface in the tool each time the script is run. For example:

   ```
   <core:metadata>
     <core:name>smartegm-example-stress-test-001.xml</core:name>
     <core:author>Jane Smith</core:author>
     <core:summary>Our first Tiger script.</core:summary>
     <core:description>This script sends the host a stream of G2S
         commands.</core:description>
     <core:create-date>2009-02-1012T12:00:00.000</core:create-date>
     <core:last-modified-date>2009-02-1712T12:00:00.000</core:last-modified-date>
     <core:version>1.0.0</core:version>
     <core:gsa-protocol-name>G2S</core:gsa-protocol-name>
     <core:gsa-protocol-name>1.0.3</core:gsa-protocol-version>
     <core:tiger-dialect>G2S_egm</core:tiger-dialect>
   </core:metadata>
   ```

4. Enter the script parameters (optional). Parameters define script behavior. In this example, a parameter is used to tell the script to loop nine times:

   ```
   <core:parameters>
     <core:integerParameter core:name="loopDuration" core:default-value="9" />
   </core:parameters>
   <core:log>We are at the top of the script.</core:log>
   ```

5. Use the `tiger:assert` element to verify that key elements are present and enabled on the EGM. If any of the asserts fail, the script stops running and reports an error.

```
<tiger:assert>
        <tiger:device-exists tiger:device-class="G2S_cabinet" tiger:device-id="-2"
        />
        <tiger:device-exists tiger:device-class="G2S_eventHandler" tiger:device-
        id="-2" />
        <tiger:device-exists tiger:device-class="G2S_noteAcceptor" tiger:device-
        id="-2" />
        <tiger:device-exists tiger:device-class="G2S_meters" tiger:device-id="-2"
        />
        <tiger:money-in-enabled tiger:expected-state="true" />
        <tiger:money-out-enabled tiger:expected-state="true" />
        <tiger:game-play-enabled tiger:expected-state="true" />
  </tiger:assert>
```

6. Enter the script commands you want executed by the script.

```
<tiger>
  <tiger:repeat tiger:iterations="loopDuration + 1">
  <tiger:Human.insertID tiger:device-id="1" tiger:id-number="12345678" />
  <tiger:Human.insertCoins tiger:device-id="1" tiger:currency-id="USD"
        tiger:denom-id="10000" tiger:coin-action="HOPPER" tiger:count="4" />
  <tiger:Human.playSimpleGame tiger:device-id="1" tiger:denom-id="1000"
        tiger:primary-win="25000" tiger:credits-to-wager-cashable="5" />
  <tiger:Human.createVoucher tiger:device-id="1" tiger:credit-type="CASHABLE" />
  <tiger:Human.removeID tiger:device-id="1" />
  <tiger:repeat>
  <core:log>The script has now ended.</core:log>
</tiger:tiger>
```

In this example:

- the `tiger:repeat` verb contains all of the information that will be repeated in the script.
- `tiger:iterations` defines the number of times the script will execute. In this case, the script will run twice - the script will run once and then loop one time (`loopDuration` +1). In step 4, `loopDuration` equals **9**, so the script will run 10 times.
- several Human verbs simulate EGM activity: insertID, insertNote, insertCoins, playSimpleGame, createVoucher, and removeID. Note that there is a logical order to the verbs. For example, a player ID must be inserted before it can be removed.
- the log verb writes an entry to the debug log.

If you are using code snippets, paste them into this section.

How to Create a Tiger Script

If you are new to script writing, we suggest you use an existing RadBlue script and modify it as needed.

G2S scripts are located in the tool's directory under **scripts** > **smart-egm**. Be sure to save the file you are editing under a different name, so you don't overwrite the original file.

The process for creating a custom Tiger script is:

1. Save a copy of an existing Tiger script.
2. Modify the new Tiger script with the appropriate metadata.
3. Use Tiger verb commands and their associated code snippets (under **Examples** for each commands) to create the actions that your script will simulate.

Sample Tiger Script

```
<?xml version="1.0" encoding="UTF-8"?>
<tiger:tiger xmlns:tiger="http://www.radblue.com/g2s/egm-data-model/schemas/v1.0.0/"
      xmlns:core="http://www.radblue.com/tiger/core/schemas/v1.0.0/">
      <core:metadata>
            <core:name>smartegm-example-stress-test-001.xml</core:name>
            <core:author>Marty Wegner</core:author>
            <core:summary>Our first Tiger script for the SmartEGM.</core:summary>
            <core:description>
                  This script kicks the Host for a while, sending it a stream of G2S
                  commands.
            </core:description>
            <core:create-date>2007-03-12T12:00:00.000</core:create-date>
            <core:last-modified-date>2007-05-22T17:13:00.000</core:last-modified-date>
            <core:version>1.0.0</core:version>
            <core:gsa-protocol-name>G2S</core:gsa-protocol-name>
            <core:gsa-protocol-version>1.0.3</core:gsa-protocol-version>
            <core:tiger-dialect>G2S_egm</core:tiger-dialect>
      </core:metadata>
      <core:parameters>
            <core:integerParameter core:name="loopDuration" core:default-value="10" />
      </core:parameters>
      <core:log>We are at the top of the script.</core:log>
      <core:config core:name="engine.inter-verb-delay" core:value="0" />
      <tiger:assert>
            <tiger:device-exists tiger:device-class="G2S_cabinet" tiger:device-id="-2" />
            <tiger:device-exists tiger:device-class="G2S_eventHandler" tiger:device-id="-2"
            />
            <tiger:device-exists tiger:device-class="G2S_noteAcceptor" tiger:device-id="-2"
            />
            <tiger:device-exists tiger:device-class="G2S_meters" tiger:device-id="-2" />
            <tiger:money-in-enabled tiger:expected-state="true" />
            <tiger:money-out-enabled tiger:expected-state="true" />
            <tiger:game-play-enabled tiger:expected-state="true" />
      </tiger:assert>
      <tiger:DataModel.waitForDeviceState
            tiger:device-class="G2S_communications"
            tiger:timeout="600000"/>
      <tiger:repeat tiger:iterations="loopDuration">
            <tiger:Human.insertNotes
                  tiger:device-id="1"
                  tiger:currency-id="USD"
                  tiger:denom-id="100000"
                  tiger:note-action="DROP" />
            <tiger:Human.playSimpleGame
                  tiger:device-id="1"
                  tiger:denom-id="25000"
                  tiger:primary-win="0"
                  tiger:credits-to-wager-cashable="4" />
            <tiger:Human.dispenseCoins
                  tiger:currency-id="USD"
```

```
                 tiger:credit-type="CASHABLE"
                 tiger:denom-id="25000"/>
           <tiger:DataModel.waitForEventQueueToDrain
                 tiger:timeout="30000"
                 tiger:sleep-interval="500" />
      </tiger:repeat>
      <core:log>The script is now ended.</core:log>
</tiger:tiger>
```

Prefix Namespace Bindings

Prefix namespace bindings are used to associate elements and attribute names with the namespace name in the attribute value in the scope of the element to which the declaration is attached.

Sample Namespace Declaration

```
<?xml version="1.0" encoding="UTF-8"?>
<tiger:tiger xmlns:tiger="http://www.radblue.com/g2s/egm-data-model/schemas/v1.0.0/"
      xmlns:core="http://www.radblue.com/tiger/core/schemas/v1.0.0/">
```

Prefix	Namespace URI/Binding Location
core	**Location**: http://www.radblue.com/tiger/core/schemas/v1.0.0/ **File**: SmartEGMTigerScript.xsd **Element**: <xs:schema … >
core	**Location**: http://www.radblue.com/tiger/core/schemas/v1.0.0/ **File**: SmartEGMTigerScript.xsd **Element**: <xs:schema … >
tiger	**Location**: http://www.radblue.com/g2s/egm-data-model/schemas/v1.0.0/ **File**: SmartEGMTigerScript.xsd **Element**: <xs:schema … >
xhtml	**Location**: http://www.w3.org/1999/xhtml **File**: TigerScriptCore.xsd **Element**: <xs:schema … >
xs	**Location**: http://www.w3.org/2001/XMLSchema **File**: SmartEGMTigerScript.xsd **Element**: <xs:schema … >
xs	**Location**: http://www.w3.org/2001/XMLSchema **File**: TigerScriptCore.xsd **Element**: <xs:schema … >

References

The G2S Message Protocol is available from the Gaming Standards Association's website.

EGM Tiger Verb Commands List by Function

Branching Statements	
tiger:break	The `tiger:break` building block breaks out of an enclosing repeat loop.
core:continue	The `tiger:continue` building block continues to the top of the enclosing repeat loop.
core:exit	The `core:exit` verb exits the script immediately, reporting a successful result to the user through the user interface.
core:fail	The `core:fail` exits the script immediately, reporting an error result to the user through the user interface.
tiger:if-vendor	The `tiger:if-vendor` verb allows you to have condition logic based on the vendor of the current EGM. If the vendor the current EGM matches the given vendor then the verbs in the then block are executed. If the vendor of the current EGM does not match the given vendor then the verbs in the else block are executed.
tiger:random	The `tiger:random` verb executes verbs randomly. Here is how it works: • All of the weight attributes from the `tiger:<tiger:action/>` child elements are put into a sequence which generates a range of **[0,*n*>** where *n* is the total of all of the weights. • When the `tiger:<random/>` element is executed, a random number is generate in the range **[0,*n*>**. • The `tiger:<tiger:action/>` element that corresponds in the range with the random value is selected. • All of the child elements in the selected `<tiger:action/>` element are then executed one a-t a time.
tiger:repeat	The `tiger:repeat` verb is used to repeat a set of verbs zero or more times. The content of the verb is one or more Tiger/XML verbs. You can even have another tiger:repeat verb.
tiger:try-catch	The `tiger:try-catch` verb is a means for catching and handling verb errors.

Structural Statements	
core:include	The `core:include` verb allows a script to be included at runtime. When `core:include` is first executed, the contents of the filename are parsed and included into the current script. Then, the verbs are executed.

Structural Statements	
core:log	The `core:log` verb is used to send output to the logging facility. The output to be sent is the content of the element. As currently defined the content must be text only. Also, no parameter substitution is supported at this time. Depending on where Tiger is running and how the logging facility is configured this text may be written out to a Logger panel or to a log file. Check with your production configuration to determine where the logging information is sent.
core:parameters	The `core:parameters` verb defines parameters that are provided / can be modified by the end user. The parameters verb is used for defining parameters which must be provided by the end user (assuming there are no defaults).
core:pause	The `core:pause` verb is used to paused the script. The verb can wait a fixed number of seconds or a random number of seconds. Please refer to the type core:pause for further details.

Metadata	
core:author	The `core:author` element is used to specify the author of the Tiger script.
core:create-date	The `core:create-date` element defines the date and time the script is created.
core:description	The `core:description` element provides a description of the script.
core:metadata	The `core:metadata` verb is used for defining information about the script that can be displayed by various tools that work with Tiger scripts.
core:gsa-protocol-name	The `core:gsa-protocol-name` element provides the GSA protocol used by the script.
core:gsa-protocol-version	The `core:gsa-protocol-version` element provides the version of the GSA protocol used by the script.
core:last-modified-date	The `core:last-modified-date` element provides the date and time of the latest script modification.
core:name	The `core:name` element denotes the script name.
core:summary	The `core:summary` element summarizes the content of the script.
core:transport	The `core:transport` element defines the type of transport used with the script.
core:version	The `core:version` element indicates the script version.

Data Model	
tiger:DataModel.snapshot	The `tiger:DataModel.snapshot` verb causes the SmartEGM to take a snapshot of the datamodel and store it under the given name.
tiger:DataModel.waitForDeviceState	The `tiger:DataModel.waitForDeviceState` verb causes the script to wait until the Host Enabled flag changes for the given device. A typical use for this verb is to have the script wait for the host to enable a device, say the Cabinet device. Note that not all devices allow their Host Enabled flag to be set or changed. These devices can be used with this verb but you will always get a time-out.
tiger:DataModel.waitForEventQueueToDrain	The `tiger:DataModel.waitForEventQueueToDrain` verb causes the script to wait until the outbound event queue has fully drained or a timeout occurs.
tiger:DataModel.waitForHandpayKeyOff	The `tiger:DataModel.waitForHandpayKeyOff` verb causes the script to wait until the outstanding handpay event has been keyed-off. This verb is intended to cause the EGM to wait until the host can send a `Handpay.setRemoteKeyOff` verb.

Assertions	
tiger:assert	The `tiger:assert` verb is used to assert or guarantee certain facts about the current EGM Data Model.
tiger:device-enabled	The `tiger:device-enabled` verb is used to check that a device is enabled in the data model.
tiger:device-exists	The `tiger:device-exists` assertion is used to check that a device exists in the data model.
tiger:egm-size-of	The `tiger:egm-size-of` assertion checks the current RAM footprint of the EGM data model.
tiger:game-play-enabled	The `tiger:game-play-enabled` assertion checks the current state of the *enableGamePlay* attribute in the cabinet status.
tiger:money-in-enabled	The `tiger:money-in-enabled` assertion checks the current state of the *enableMoneyIn* attribute of the cabinet status.
tiger:money-out-enabled	The `tiger:money-out-enabled` assertion checks the current state of the *enableMoneyOut*attribute of the cabinet status.

Custom Commands	
tiger:Transport.sendMyCommand	The `Transport.sendMyCommand` verb sends any valid G2S command to the owner of the specified device.

Custom Commands	
tiger:Transport.sendRawMessage	The `Transport.sendRawMessage` verb sends a raw message to a given host.

Physical Device Events	
tiger:Egm.resetEgm	The `tiger:Egm.resetEgm` verb resets the EGM.
tiger:Events.cabinetEvents	The `Events.cabinetEvents` verb simulates the generation of cabinet device related events.
tiger:Events.coinAcceptorEvents	The `tiger:Events.coinAcceptorEvents` verb simulates the generation of Coin Acceptor device related events.
tiger:Events.noteAcceptorEvents	The `tiger:Events.NoteAcceptorEvents` verb sets/clears a Note Acceptor event.
tiger:Events.printerEvents	The `tiger:Events.printerEvents` verb simulates the generation of Printer device related events.
tiger:Human.changeDoorState	The `tiger:Human.changeDoorState` verb simulates the opening and closing of EGM cabinet doors (cabinet, logic or auxiliary).

Player Activity	
tiger:Human.insertID	The `tiger:Human.insertID` verb simulates the insertion an ID into a reader.
tiger:Human.insertIDFromDatabase	The `tiger:Human.insertIDFromDatabase` verb simulates the insertion an ID into a reader. The ID number is read from the specified XML database.
tiger:Human.removeID	The `tiger:Human.removeID` verb simulates the removal of an ID from a reader.
tiger:Human.removeIDToDatabase	The `tiger:Human.removeIDToDatabase` verb simulates the removal of an ID from a reader. The ID number that is removed is unlocked if it was locked when it was inserted.

Currency Activity - Coin	
tiger:Human.insertCoins	The `tiger:Human.insertCoins` verb simulates the player inserting one or more coins (or tokens) into a Coin Acceptor device.
tiger:Human.dispenseCoins	The `tiger:Human.dispenseCoins` verb simulates the dispensing of coins from the hopper.
tiger:Human.doCoinDrop	The `tiger:Human.doCoinDrop` verb simulates the performing of a coin drop.

Currency Activity - Note	
tiger:Human.insertNote	The `tiger:Human.insertNote` verb simulates the player inserting a note (or bill) into a Note Acceptor device.
tiger.Human.dispenseNotes	The `tiger:Human.dispenseNotes` verb simulates the dispensing of notes from the note dispenser.
tiger:Human.doNoteDrop	The `tiger:Human.doNoteDrop` verb simulates the performing of a note drop.

Currency Activity - Voucher	
tiger:Human.createVoucher	The `tiger:Human.insertVoucher` verb simulates the insertion of a voucher.
tiger:Human.createVoucherToDatabase	
tiger:Human.insertVoucher	The `tiger:Human.insertVoucher` verb simulates the insertion of a voucher.
tiger:Human.insertVoucherFromDatabase	The `tiger:Human.insertVoucherFromDatabase` verb simulates the insertion of a voucher. The voucher validation ID comes from the specified database.
tiger:if-voucher-available	The `tiger:if-voucher-available` verb allows you to have condition logic based on if there is a voucher available in the database. If there is a voucher available then the verbs in the then block are executed. If there are no vouchers available in the database then the else block are executed.

Currency Activity - WAT	
tiger:Human.getWATAccounts	The `tiger:Human.getWATAccounts` verb simulates a human requesting the list of wagering accounts that are available to that person. This verb uses the player's account information from the ID Reader device that is configured in the wat.watProfile.
tiger:Human.getWATBalance	The `tiger:Human.getWATBalance` verb simulates a human requesting the available balance from one of the wagering accounts available to that person. This verb uses the player's account information from the ID Reader device that is configured in the `wat.watProfile`.
tiger:Human.watToEGM	The `tiger:Human.watToEGM` verb simulates a WAT Transfer from the Host to the EGM.
tiger:Human.watToHost	The `tiger:Human.watToHost` verb simulates a WAT Transfer from the EGM to the Host.
tiger:WAT.getKeyPair	The `tiger:WAT.GetKeyPair` verb sends a `wat.getKeyPair`

Currency Activity - WAT	
	command. The keyPairId is set to `wat.watStatus.keyPairId` plus one. The hashType is set to `wat.watProfile.hashType`.

Game Play Activity	
tiger:Human.playPaytableGame	The `tiger:Human.playPaytableGame` verb simulates game play using paytables to determine game outcomes.
tiger:Human.playSimpleGame	The `tiger:Human.playSimpleGame` verb simulates the player playing a simple, non-central determination game. A simple game is one where there is only a primary game, with no change to the original bet. Future releases of Tiger will contain more complex game play verb.
tiger:Human.playSimpleCentralGame	The `tiger:Human.playSimpleCentralGame` verb simulates the player playing a simple, central determination game. A simple game is one where there is only a primary game, no secondary game, and no change in the original bet. Future releases of Tiger will contain more complex game play verb.
tiger:Progressive.getHostInfo	The `tiger:Progressive.getHostInfo` verb sends the `progressive.getHostInfo` command.

Handpay	
tiger:DataModel.waitForHandpayKeyOff	The `tiger:DataModel.waitForHandpayKeyOff` verb causes the script to wait until the outstanding handpay event has been keyed off.
tiger:Human.cancelCancelCreditHandpay	The `tiger:Human.cancelCancelCreditHandpay` verb simulates a player canceling a cancel credit handpay.
tiger:Human.CashOut	The `tiger:Human.cashOut` verb simulates a player pressing the cash out button. A cancel credit handpay is currently the only supported pay out method.
tiger:Human.KeyOff	The `tiger:Human.keyOff` verb simulates an EGM attendant clearing a handpay condition by selecting the method to pay out the handpay.
tiger:if-cashable-credit-meter	The `tiger:if-cashable-credit-meter` verb allows you to add a condition based on the current value of the cashable credit meter.

About Metadata

Metadata data is information about the script. Metadata is intended for users of the script and does not affect script functionality. When you run a Tiger script, the metadata displays in the tool's user interface.

For any Tiger script you create, you can add the following metadata:

- author
- script creation date
- script description
- GSA protocol name
- GSA protocol version
- last date script was modified
- name of script
- summary of script content
- type of transport used by script
- script version number

core:metadata

The `core:metadata` verb is used for defining information about the script that can be displayed by various tools that work with Tiger scripts. Since Tiger scripts are often stored in databases, this verb provides a mechanism for retaining useful contextual information about the script. It is strongly suggested that all scripts contain metadata.

The `core:metadata` verb is used for defining information about the script that can be displayed by various tools that work with Tiger scripts. Since Tiger scripts are often stored in databases, this verb provides a mechanism for retaining useful contextual information about the script. It is strongly suggested that all scripts should contain metadata.

> **Note:** The Tiger schema allows for more than one `metadata` verb. Only the first `metadata` verb is guaranteed to be used by any Tiger tool. Subsequent `metadata` verbs are silently ignored.

This verb may contain the following elements: core:author, core:create-date, core:description, core:gsa-protocol-name, core:gsa-protocol-version, core:last-modified-date, core:name, core:summary, core:transport and core:version.

Elements

Element	Restrictions	Description
core:name	type: string use: required minLength: 1 maxLength: 64 default: "Unknown"	Name of the script. This name does not have to match the name of the script text file.
core:author	type: string use: optional minLength: 0 maxLength: 64 default: " "	Author of the script.
core:summary	type: string use: optional minLength: 0 maxLength: 64 default: "Default Summary"	Brief summary of the script.
core:description	type: string use: optional minLength: 0 maxLength: 256 default: "Default Description"	Description of the script.

Element	Restrictions	Description
core:create-date	type: dateTime use: optional default: "1966-06-23T00:00:00:000"	Date on which the script was created.
core:last-modified-date	type: string use: required default: "1966-06-23T00:00:00:000"	Date on which the script was last modified.
core:version	type: string use: optional minLength: 0 maxLength: 12 default: "1.0.0"	Version number of the script. Use it to track new releases of the script.
core:gsa-protocol-name	type: core:gsa-protocol-name use: optional default: "G2S"	Name of the GSA protocol used by the script.
core:gsa-protocol-version	type: string use: optional minLength: 0 maxLength: 12 G2S default: "1.0.3"	Version of the GSA protocol used by the script.
core:transport	type: string use: optional default: "standard"	Type of transport to use with the script. *This element has been deprecated*.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet defines the metadata for the script.

```
<core:metadata>
     <core:name>example-script-001.xml</core:name>
     <core:version>1.0.0</core:version>
     <core:author>Biff Brandon</core:author>
     <core:summary>The first demonstration script.</core:summary>
     <core:description>This script shows off the basics of a Tiger script. It
     demonstrates the basic commands in the Tiger
       scripting language.</core:description>
     <core:create-date>2007-03-21T09:39:00.000</core:create-date>
     <core:last-modified-date>2007-03-21T10:05:00.000</core:last-modified-date>
     <core:gsa-protocol-name>G2S</core:gsa-protocol-name>
     <core:gsa-protocol-version>1.0.0</core:gsa-protocol-version>
     <core:tiger-dialect>G2S_host</core:tiger-dialect>
     <core:transport>standard</core:transport>
</core:metadata>
```

core:author

The `core:author` element is used to specify the author of the Tiger script.

Content

Content	Restrictions	Description
text	type: string maxLength: 64 minLength: 0 default: "Unknown"	Name or identifier of Tiger script author.

This element may be included in: core:metadata.

Example

```
    <core:author>I. Write Scripts</core:author>
```

core:create-date

The `core:create-date` element defines the date and time the script is created.

Content

Content	Restrictions	Description
text	type: dateTime use: optional default: "1966-06-23T00:00:00:000"	Date on which the script was created.

This element may be included in: core:metadata.

Example

```
    <core:create-date>1966-06-23T00:02:30:000</core>
```

core:description

The `core:description` element provides a description of the script.

Content

Content	Restrictions	Description
text	type: string use: optional minLength: 0 maxLength: 256 default: "Default Description"	Description of the script.

This element may be included in: core:metadata.

Example

```
    <core:description> This script tests the WAT class.</core:description>
```

core:gsa-protocol-name

The `core:gsa-protocol-name` element provides the GSA protocol used by the script.

Content

Content	Restrictions	Description
text	type: core:gsa-protocol-name use: optional default: "G2S"	Name of the GSA protocol used by the script.

This element may be included in: core:metadata.

Example

```
    <core:gsa-protocol-name>G2S</core:gsa-protocol-name>
```

core:gsa-protocol-version

The `core:gsa-protocol-version` element provides the version of the GSA protocol used by the script.

Content

Content	Restrictions	Description
text	type: string use: optional minLength: 0 maxLength: 12 G2S default: "1.0.3" S2S default: "1.2.6"	Version of the GSA protocol used by the script.

This element may be included in: core:metadata.

Example

```
    <core:gsa-protocol-version>1.0.3</gsa-protocol-version>
```

core:last-modified-date

The `core:last-modified-date` element provides the date and time of the latest script modification.

Content

Content	Restrictions	Description
text	type: string use: required default: "1966-06-23T00:00:00:000"	Date on which the script was last modified.

This element may be included in: core:metadata.

Example

```
    <core:last-modified-date>1966-06-23T00:02:30:000</last-modified-date>
```

core:name

The `core:name` element denotes the script name.

Content

Attribute	Use	Restrictions	Default	Description
core:name	required	xs:string maxLength: 256 minLength:1	no default	Name of the script.

This element may be included in: core:metadata.

Example

```
    <core:name>WAT Test Script 1</core:name>
```

core:summary

The `core:summary` element summarizes the content of the script.

Content

Content	Restrictions	Description
text	type: string use: optional minLength: 0 maxLength: 64 default: "Default Summary"	Brief summary of the script.

This element may be included in: core:metadata.

Example

```
    <core:summary>This script exercises a host's WAT class implementation.</core:summary>
```

core:transport

The `core:transport` element defines the type of transport used with the script.

Content

Content	Restrictions	Description
text	type: string use: optional default: "standard"	Type of transport to use with the script. *This element has been deprecated*.

This element may be included in: core:metadata.

Example

```
    <core:transport>standard</core:transport>
```

core:version

The `core:version` element indicates the script version.

Content

Content	Restrictions	Description
text	type: string maxLength: 12 minLength: 0 default: "1.0.0"	Version of the script.

This element may be included in: core:metadata.

Example

```
<core.version>1.0.0</core.version>
```

tiger:assert

The `tiger:assert` verb is used to assert or guarantee certain facts about the current EGM Data Model. An assertion is something that must be true for the script to move forward.

When the `tiger:assert` verb executes, it executes each child element to determine if the assertion is true. If all of the child assertions are true, the verb returns with a success and the script continues. If any of the child assertions are not true, the verb returns with an error and the script terminates.

Uses of assertions include:

- Verifying that a device exists.
- Verifying that a device is enabled or disabled.
- Verifying that the cabinet device is in the right state.

You use an assertion to guarantee that the EGM data model is in a particular state. This is very useful for when a script has a set of data model requirements that must be true to proceed. If any of these requirements are not met, the script fails.

Elements

Element	Description
tiger:device-exists	This assertion checks that a device exists in the data model. This assertion checks whether the given device exists.
tiger:device-enabled	This assertion checks that a device is enabled in the data model. This assertion checks whether the given device exists *and* is enabled in the SmartEGM data model.
tiger:game-play-enabled	This assertion checks the current state of the *enableGamePlay* attribute in the cabinet status.
tiger:money-in-enabled	This assertion checks the current state of the *enableMoneyIn* attribute in the cabinet Status.
tiger:money-out-enabled	This assertion checks the current state of the *enableMoneyOut*attribute in the cabinet Status.
tiger:egm-size-of	This assertion checks the current RAM footprint of the EGM data model.

This element may contain the following elements: tiger:device-exists, tiger:device-enabled, tiger:game-play-enabled, tiger:money-in-enabled, tiger:money-out-enabled and tiger:egm-size-of.

This verb command may be included in the following elements: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet is a block of asserts for checking the current state of the EGM data model.

```
<tiger:assert>
     <tiger:device-exists tiger:device-class="G2S_cabinet"
       tiger:device-id="-2"/>
     <tiger:device-exists tiger:device-class="G2S_eventHandler"
       tiger:device-id="-2"/>
     <tiger:device-exists tiger:device-class="G2S_noteAcceptor"
       tiger:device-id="-2"/>
     <tiger:device-exists tiger:device-class="G2S_meters"
       tiger:device-id="-2"/>
</tiger:assert>
```

tiger:device-enabled

The `tiger:device-enabled` verb is used to check that a device is enabled in the data model.

For the purpose of this verb, "enabled" is defined as the device existing in the data model and the *hostEnabled* attribute from the status information for the given device. *Both of these conditions must be* **true** *for this assertion to pass.*

Child of `tiger:assert` element.

Attributes

Attribute	Restrictions	Description
tiger:device-class	type: tiger: device-class use: required	G2S device class of the device to use.
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first device from the specified device class. • A value of "-1" is legal, but should not be used because it will result in a failure.

Example

This snippet asserts that the cabinet device is enabled.

```
<tiger:assert>
     <tiger:device-enabled tiger:device-class="G2S_cabinet" tiger:device-id="-2" />
</tiger:assert>
```

tiger:device-exists

The `tiger:device-exists` assertion is used to check that a device exists in the data model. If the specified device exists in the data model, in any state, the assertion passes. If the specified device does not exist in the data model, the assertion fails.

Child of `tiger:assert` element.

Attributes

Attribute	Restrictions	Description
tiger:device-class	type: tiger: device-class use: required	G2S device class of the device to use.
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first device from the specified device class. • A value of "-1" is legal, but should not be used because it will result in a failure.

Example

This snippet asserts that the cabinet device exists.

```
<tiger:assert>
     <tiger:device-exists tiger:device-class="G2S_cabinet" tiger:device-id="-2" />
</tiger:assert>
```

tiger:egm-size-of

The `tiger:egm-size-of` assertion checks the current RAM footprint of the EGM data model. This assertion can be used to stop a script if the EGM data model consumes more than a certain amount of RAM. This is useful in a load testing situation where the script may be causing too much RAM to be consumed.

When the assertion executes, the data model of the EGM is measured. If it exceeds the given maximum, the assertion fails.

Child of `tiger:assert` element.

Attribute

Attribute	Restrictions	Description
tiger:max-size	type: integer use: default default: "3000000"	Maximum size, in bytes, for the data model.

Example

This snippet asserts that the EGM data model is using less than 10 MB of RAM.

```
<tiger:assert>
     <tiger:egm-size-of tiger:max-size="10485760 " />
</tiger:assert>
```

tiger:game-play-enabled

The `tiger:game-play-enabled` assertion checks the current state of the *enableGamePlay* attribute in the cabinet status. The current value of the attribute must match the value given in the assertion. If the two values match, the assertion passes. If the two values do not match, the assertion fails.

Child of `tiger:assert` element.

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first Cabinet device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:expected-state	type: boolean use: optional default: "true"	The expected value of the *enableGamePlay* attribute.

Example

This snippet asserts that gamePlay is enabled.

```
<tiger:assert>
     <tiger:game-play-enabled tiger:device-id="-2" />
</tiger:assert>
```

tiger:money-in-enabled

The `tiger:money-in-enabled` assertion checks the current state of the *enableMoneyIn* attribute of the cabinet status. The current value of the attribute must match the value given in the assertion. If the two values match, the assertion passes. If the two values do not match, the assertion fails.

Child of `tiger:assert` element.

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first cabinet device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger: expected-state	type: boolean use: optional default: "true"	The expected value of the *enableMoneyIn* attribute.

Examples

This snippet asserts that money-in is enabled.

```
<tiger:assert>
     <tiger:money-in-enabled tiger:device-id="-2" />
</tiger:assert>
```

This snippet asserts that money-in is disabled.

```
<tiger:assert>
     <tiger:money-in-enabled tiger:device-id="-2" tiger:expected-state="false" />
</tiger:assert>
```

tiger:money-out-enabled

The `tiger:money-out-enabled` assertion checks the current state of the *enableMoneyOut* attribute of the cabinet status. The current value of the attribute must match the value given in the assertion. If the two values match, the assertion passes. If the two values do not match, the assertion fails.

Child of `tiger:assert` element.

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default:"-2"	Device identifier. • A value of "-2" means the first cabinet device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger: expected-state	type: boolean use: optional default: "true"	The expected value of the *enableMoneyOut* attribute.

Examples

This snippet asserts that money-out is enabled.

```
<tiger:assert>
     <tiger:money-out-enabled tiger:device-id="-2" />
</tiger:assert>
```

This snippet asserts that money-out is disabled.

```
<tiger:assert>
     <tiger:money-out-enabled tiger:device-id="-2" tiger:expected-state="false" />
</tiger:assert>
```

tiger:action

The `tiger:action` building block defines the percentage of time (weight) a given action is executed in the script.

This building block may contain the following child elements:

core:log, core:metadata, core:parameters, core:pause, core:fail, core:exit, core:include, tiger:assert, tiger:repeat, tiger:break, tiger:continue, tiger:random, tiger:try-catch, tiger:if-vendor, tiger:if-voucher-available, tiger:DataModel.waitForDeviceState, tiger:DataModel.waitForEventQueueToDrain, tiger:DataModel.waitForHandpayKeyOff, tiger:DataModel.snapshot, tiger:Egm.resetEgm, tiger:Events.coinAcceptorEvents, tiger:Events.noteAcceptorEvents, tiger:Events.printerEvents, tiger:Human.cancelCancelCreditHandpay, tiger:Human.changeDoorState, tiger:Human.createVoucher, tiger:Human.createVoucherToDatabase, tiger:Human.dispenseCoins, tiger:Human.dispenseNotes, tiger:Human.doCoinDrop, tiger:Human.doNoteDrop, tiger:Human.getWATAccounts, tiger:Human.getWATBalance, tiger:Human.insertCoins, tiger:Human.insertID, tiger:Human.insertIDFromDatabase, tiger:Human.insertNote, tiger:Human.insertVoucher, tiger:Human.insertVoucherFromDatabase, tiger:Human.playSimpleGame, tiger:Human.playSimpleCentralGame, tiger:Human.playPaytableGame, tiger:Human.removeID, tiger:Human.removeIDToDatabase, tiger:Human.watToEGM, tiger:Human.watToHost, tiger:Human.cashOut, tiger:Human.keyOff, tiger:Progressive.getHostInfo, tiger:Transport.sendMyCommand, tiger:Transport.sendRawMessage, tiger:WAT.getKeyPair and tiger:Events.cabinetEvents.

Attribute

Attribute	Restrictions	Description
tiger:weight	type: integer use: optional default: "100"	Percent of time action is executed. The weight value is divided by the total weight.

This building block may be included in tiger:random.

Example

```
<tiger:random>
     <tiger:action tiger:weight="50">
       <tiger:Human.insertNote tiger:device-id="1" tiger:currency-id="USD" tiger:denom-
       id="100000" tiger:note-action="DROP"/>
     </tiger:action>
     <tiger:action tiger:weight="5">
       <tiger:pause/>
     </tiger:action>
     <tiger:action>
       <tiger:pause/>
     </tiger:action>
     <tiger:action>
       <tiger:Human.playSimpleGame tiger:device-id="7" tiger:denom-id="25000"
       tiger:primary-win="1" tiger:credits-to-wager-cashable="2"/>
     </tiger:action>
</tiger:random>
```

tiger:break

The `tiger:break` building block breaks out of an enclosing repeat loop.

Attribute

Attribute	Restrictions	Description
There are no attributes for this statement.		

This building block may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then , tiger:tiger and tiger:try.

Example

This snippet causes control to break out of the enclosing repeat loop.

```
<core:break/>
```

tiger:continue

The `tiger:continue` building block continues to the top of the enclosing repeat loop.

Attribute

Attribute	Restrictions	Description
There are no attributes for this statement.		

This building block may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then , tiger:tiger and tiger:try.

Example

This snippet causes control to pass to the top of the enclosing repeat loop.

```
<core:continue/>
```

core:exit

The `core:exit` building block is used to stop the Tiger script. The script returns a successful outcome, and no additional verbs are executed. Use this verb to prematurely terminate a script.

See also: core:fail

Attribute

Attribute	Restrictions	Description
core: message	type:string use: optional maxLength: 256 minLength:1 default: " "	Message is returned as the final result of the script. This attribute is useful when the Tiger script is running as part of a larger application.

This building block may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then , tiger:tiger and tiger:try.

Example

This snippet causes the script to exit.

```
    <core:exit/>
```

This snippet causes the script to exit and a specified return a message.

```
    <core:exit core:message="The script has ended."/>
```

core:fail

The `core:fail` building block is used to stop the script, and return a failed outcome. No additional verbs are executed once the `core:fail` verb has been executed. Use this verb to prematurely terminate a script with a failure.

See also: core:exit

Attribute

Attribute	Restrictions	Description
core:message	type:string use: optional maxLength: 256 minLength:1 default: " "	Message is returned as the final result of the script. This is useful when the Tiger script is running as part of a larger application.

This building block may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then , tiger:tiger and tiger:try.

Example

This snippet causes the script to fail.

```
    <core:fail/>
```

This snippet causes the script to fail and return a message.

```
    <core:fail core:message="The script has failed."/>
```

core:include

The `core:include` building block allows you to include another Tiger or XML script in the current script. The included content acts as if it were typed directly into the including script. You can nest `core:include` building blocks, but be careful not to invoke a circular chain.

There are three ways to specify the name, location and content of the Tiger or XML to include:

1. By filename: `<core:include core:filename="startup.xml" />`
   Includes a file based on a filename. The filename can be relative to the bin directory of the product or it can specify the full path (for example, C:\temp\startup.xml).
2. By URI: `<core:include core:uri=http://www.radblue.com/scripts/startup.xml/>`
   Includes a file through a valid URI. This filename can point to any resource reachable through a URI. If the URI is an HTTP resource, the product must have access to the network and/or to the Internet.
3. By resource: `<core:include core:resource="startup.xml" />`
   Includes a file relative to the class path of the product. The class path defines a list of directories in which the Tiger engine looks for scripts. The class path is defined when the product starts.

   If the included script is not found or is invalid Tiger or XML, the entire script fails with an error.

Attributes

Attribute	Restrictions	Description
core:filename	type:string use: optional maxLength: 256 minLength:1	The `core:filename` attribute points to where to find the Tiger script to include. The filename can be either relative to the current directory or absolute.
core:resource	type:string use: optional maxLength: 256 minLength:1	The `core:resource` attribute points to where to find the Tiger script to include. The filename is searched for on the current class path.
core:uri	type:anyURI use: optional	The `core:resource` attribute points to where to find the Tiger script to include. The filename is any valid URI.

Examples

This snippet includes the **Tiger script ../scripts/tiger/library.xml** relative to the current directory:

```
<core:include core:filename="../scripts/tiger/library.xml" />
```

This snippet includes a Tiger script from the Internet:

```
<core:include core:uri="http://www.radblue.com/scripts/tiger/library.xml" />
```

This snippet includes a Tiger script from the class path:

```
<core:include core:resource="library.xml" />
```

Usage

This element may be included in:

- tiger:action
- tiger:catch
- tiger:else
- tiger:repeat
- tiger:then
- tiger:tiger
- tiger:try

core:log

The `core:log` building block is used to send *output* to the logging facility. The output is the content of the element. As currently defined, the content must be text *only*. No parameter substitution is supported at this time. Depending on where Tiger is running and how the logging facility is configured, this text may be written out to a logger panel or to a log file. Check with your production configuration to determine where the logging information is sent.

> **Note:** Tiger is XML, so white space is always an issue. The XML standard specifies that white space is to be ignored. For most uses, this a good rule. However, in relation to this building block, it often *causes* issues. Be aware that it can be difficult to precisely format output when using this building block.

Attribute

Attribute	Restrictions	Description
core:level	type: core:LogLevelType use: optional default: "info"	The `core:level` attribute is used when logging a message. The level is applied against the logging configuration to determine if and where the log message is sent.

This building block may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then , tiger:tiger and tiger:try.

Example

This snippet sends a text string to the logging facility, using the INFO level.

```
<core:log>
```

This snippet sends a text string to the logging facility, using the WARN level.

```
<core:log core:level="warn">This is some text to be logged.</core:log>
```

tiger:else

The `tiger:else` element contains the verbs that are executed when a parent element performs some sort of logic.

This element may contain the following elements: core:exit, core:fail, core:include, core:log, core:metadata, core:parameters, core:pause, tiger:assert, tiger:DataModel.waitForDeviceState, tiger:DataModel.waitForEventQueueToDrain, tiger:DataModel.waitForHandpayKeyOff, tiger:Events.coinAcceptorEvents, tiger:Events.noteAcceptorEvents, tiger:Events.printerEvents, tiger:Human.cancelCancelCreditHandpay, tiger:Human.cashOut, tiger:Human.changeDoorState, tiger:Human.createVoucher, tiger:Human.dispenseCoins, tiger:Human.dispenseNotes, tiger:Human.doCoinDrop, tiger:Human.doNoteDrop, tiger:Human.getWATAccounts, tiger:Human.getWATBalance, tiger:Human.insertCoins, tiger:Human.insertID, tiger:Human.insertNote, tiger:Human.insertVoucher, tiger:Human.keyOff, tiger:Human.playSimpleGame, tiger:Human.removeID, tiger:Human.watToEGM, tiger:Human.watToHost, tiger:if-vendor, tiger:Progressive.getHostInfo, tiger:random, tiger:repeat, tiger:try-catch and tiger:WAT.getKeyPair.

This element may be included in the following elements: tiger:if-vendor and tiger:if-voucher-available.

Example

In this snippet, if the EGM is from `XYZ` company, log that fact.

```
<tiger:if-vendor tiger:vendor="XYZ">
     <tiger:then>
       <core:log>This is an XYZ EGM.</core:log>
     </tiger:then>
     <tiger:else>
       <core:log>This is not an XYZ EGM.</core:log>
     </tiger:else>
</tiger:if-vendor>
```

tiger:repeat

The `repeat` building block is used to repeat a set of verbs. You can repeat the verbs either a fixed number of iterations or for a fixed amount of time. Repeat this group of verbs this many times or for this long (Days HH:MM:SS).

Attributes

Attribute	Restrictions	Description
tiger:duration	type: string use: optional default: " "	**Constant Number of Iterations** If the value of the attribute is a string that defines a positive integer value, the loop is executed for exactly that many iterations. **Note**: The *tiger:duration* attribute is exactly the same as the *tiger:iterations* attribute except that the value of the *tiger:duration* attribute is a string and not an integer. **Example**: `tiger:duration="'15'"` **Random Number of Iterations** If the value of the attribute is a string of the pattern **"low, high"**, the loop will be executed a random number of times in the range **low to** (**high -1**). If the value of the attribute is a string of the pattern [**high**], the loop will be executed a random number of times in the range **low to (high -1)**. **Example**: "'[0,3]'" **Elapsed Time** This attribute specifies a duration in terms of elapsed time. The value of the attribute is a string value that defines the amount of time to execute the body of the loop. The value of the attribute must be a relative time value of the following format **"D HH:MM:SS"**, where *D* is the number of days (must be at least one digit), *HH* is the number of hours (must be two digits), *MM* is the number of minutes (must be two digits), *SS* is the number of seconds (must be two digits). The loop will iterate until the time has elapsed the given duration.The actual number of iterations processed is dependent on what the script does and how fast the script executes on the host machine. **Example**: `tiger:duration="'0. 00:30:00'"` repeats for 30 minutes.
tiger:iterations	type: integer use: optional default: " "	Specifies the total number of iterations to run the script. The value of this attribute is either a non-negative integer *or* the name of a parameter defined in the script. If the value is a parameter, the value of the parameter must be a non-negative integer. **Example integer**: `tiger:iterations="15"` **Example parameter**: `tiger:iterations="testCase1"`

Elements

Element	Description
core:exit	The `core:exit` verb exits the script immediately, reporting a successful result to the user through the user interface.
core:fail	The `core:fail` exits the script immediately, reporting an error result to the user through the user interface.
core:include	The `core:include` verb allows a script to be included at runtime. When `core:include` is first executed, the contents of the filename are parsed and included into the current script. Then, the verbs are executed.
core:log	The `core:log` verb is used to send output to the logging facility. The output to be sent is the content of the element. As currently defined the content must be text only. Also, no parameter substitution is supported at this time. Depending on where Tiger is running and how the logging facility is configured this text may be written out to a Logger panel or to a log file. Check with your production configuration to determine where the logging information is sent.
core:metadata	The `core:metadata` verb is used for defining information about the script that can be displayed by various tools that work with Tiger scripts.
core:parameters	The `core:parameters` verb defines parameters that are provided / can be modified by the end user. The parameters verb is used for defining parameters which must be provided by the end user (assuming there are no defaults).
core:pause	The `core:pause` verb is used to paused the script. The verb can wait a fixed number of seconds or a random number of seconds. Please refer to the type core:pause for further details.
tiger:assert	The `tiger:assert` verb is used to assert or guarantee certain facts about the current EGM Data Model.
tiger:DataModel.snapshot	The `tiger:DataModel.snapshot` verb causes the SmartEGM to take a snapshot of the datamodel and store it under the given name.
tiger:DataModel.waitForDeviceState	The `tiger:DataModel.waitForDeviceState` verb causes the script to wait until the Host Enabled flag changes for the given device. A typical use for this verb is to have the script wait for the host to enable a device, say the Cabinet device. Note that not all devices allow their Host Enabled flag to be set or changed. These devices can be used with this verb but you will always get a time-out.
tiger:DataModel.waitForHandpayKeyOff	The `tiger:DataModel.waitForHandpayKeyOff` verb causes the script to wait until the outstanding handpay event has been keyed-off. This verb is intended to cause the EGM to wait until the host can send a `Handpay.setRemoteKeyOff` verb.

Element	Description
tiger:Events.coinAcceptorEvents	The `tiger:Events.coinAcceptorEvents` verb simulates the generation of Coin Acceptor device related events.
tiger:Events.noteAcceptorEvents	The `tiger:Events.NoteAcceptorEvents` verb sets/clears a Note Acceptor event.
tiger:Events.printerEvents	The `tiger:Events.printerEvents` verb simulates the generation of Printer device related events.
tiger:Human.cancelCancelCreditHandpay	The `tiger:Human.cancelCancelCreditHandpay` verb simulates a player canceling a cancel credit handpay.
tiger:Human.cashOut	The `tiger:Human.cashOut` verb simulates a player pressing the cash out button. A cancel credit handpay is currently the only supported pay out method.
tiger:Human.changeDoorState	The `tiger:Human.changeDoorState` verb simulates the opening and closing of EGM cabinet doors (cabinet, logic or auxiliary).
tiger:Human.createVoucher	The `tiger:Human.insertVoucher` verb simulates the insertion of a voucher.
tiger:Human.createVoucherToDatabase	The `tiger:Human.insertVoucherFromDatabase` verb simulates the insertion of a voucher. The voucher validation ID comes from the specified database.
tiger:Human.dispenseCoins	The `tiger:Human.dispenseCoins` verb simulates the dispensing of coins from the hopper.
tiger:Human.dispenseNotes	The `tiger:Human.dispenseNotes` verb simulates the dispensing of notes from the note dispenser.
tiger:Human.doCoinDrop	The `tiger:Human.doCoinDrop` verb simulates the performing of a coin drop.
tiger:Human.doNoteDrop	The `tiger:Human.doNoteDrop` verb simulates the performing of a note drop.
tiger:Human.getWATAccounts	The Human.getWATAccounts verb simulates a human requesting the list of wagering accounts that are available to that person. This verb uses the player's account information from the ID Reader device that is configured in the wat.watProfile.
tiger:Human.getWATBalance	The `tiger:Human.getWATBalance` verb simulates a human requesting the available balance from one of the wagering accounts available to that person. This verb uses the player's account information from the ID Reader device that is configured in the wat.watProfile.
tiger:Human.insertCoins	The `tiger:Human.insertCoins` verb simulates the player inserting one or more coins (or tokens) into a Coin Acceptor device.
tiger:Human.insertID	The `tiger:Human.insertID` verb simulates the insertion an ID into a reader
tiger:Human.insertIDFromDatabase	The `tiger:Human.insertIDFromDatabase` verb simulates the

Element	Description
	insertion an ID into a reader. The ID number is read from the specified XML database.
tiger:Human.insertNote	The `tiger:Human.insertNote` verb simulates the player inserting a note (or bill) into a Note Acceptor device.
tiger:Human.insertVoucher	The `tiger:Human.insertVoucher` verb simulates the insertion of a voucher.
tiger:Human.insertVoucherFromDatabase	The `tiger:Human.insertVoucherFromDatabase` verb simulates the insertion of a voucher. The voucher validation ID comes from the specified database.
tiger:Human.KeyOff	The `tiger:Human.keyOff` verb simulates an EGM attendant clearing a handpay condition by selecting the method to pay out the handpay.
tiger:Human.playSimpleGame	The `tiger:Human.playSimpleGame` verb simulates the player playing a simple, non-central determination game. A simple game is one where there is only a primary game, with no change to the original bet. Future releases of Tiger will contain more complex game play verb.
tiger:Human.playSimpleCentralGame	The `tiger:Human.playSimpleCentralGame` verb simulates the player playing a simple, central determination game. A simple game is one where there is only a primary game, no secondary game, and no change in the original bet. Future releases of Tiger will contain more complex game play verb.
tiger:Human.removeID	The `tiger:Human.removeID` verb simulates the removal of an ID from a reader.
tiger:Human.removeIDToDatabase	The `tiger:Human.removeIDToDatabase` verb simulates the removal of an ID from a reader. The ID number that is removed is unlocked if it was locked when it was inserted.
tiger:Human.watToEGM	The `tiger:Human.watToEGM` verb simulates a WAT Transfer from the Host to the EGM.
tiger:Human.watToHost	The `tiger:Human.watToHost` verb simulates a WAT Transfer from the EGM to the Host.
tiger:if-vendor	The `tiger:if-vendor` verb allows you to have condition logic based on the vendor of the current EGM. If the vendor the current EGM matches the given vendor then the verbs in the then block are executed. If the vendor of the current EGM does not match the given vendor then the verbs in the else block are executed.
tiger:if-voucher-available	The `tiger:if-voucher-available` verb allows you to have condition logic based on if there is a voucher available in the database. If there is a voucher available then the verbs in the then block are executed. If there are no vouchers available in the database then the else block are executed.

Element	Description
tiger:Progressive.getHostInfo	The `tiger:Progressive.getHostInfo` verb sends the `progressive.getHostInfo` command.
tiger:repeat	This verb is used to repeat a set of verbs zero or more times. The content of the verb is one or more Tiger/XML verbs. You can even have another tiger:repeat verb.
tiger:random	The `tiger:random` verb executes verbs randomly. Here is how it works: • All of the weight attributes from the `tiger:<tiger:action/>` child elements are put into a sequence which generates a range of **[0,*n*>** where *n* is the total of all of the weights. • When the `tiger:<random/>` element is executed, a random number is generate in the range **[0,*n*>**. • The `tiger:<tiger:action/>` element that corresponds in the range with the random value is selected. • All of the child elements in the selected `<tiger:action/>` element are then executed one a-t a time.
tiger:try-catch	The `tiger:try-catch` verb is a means for catching and handling verb errors.
tiger:WAT.getKeyPair	The `tiger:WAT.GetKeyPair` verb sends a `wat.getKeyPair` command. The *keyPairId* is set to `wat.watStatus.keyPairId` plus one. The *hashType* is set to `wat.watProfile.hashType`.

Examples

This snippet sends a block of Tiger commands ten times.

```
<tiger:repeat tiger:iterations="10">
     <tiger:Human.insertID tiger:device-id="1" tiger:id-number="12345678"/>
     <tiger:Human.insertNote tiger:device-id="1" tiger:currency-id="USD" tiger:denom-
     id="100000" tiger:note-action="DROP"/>
     <tiger:Human.insertCoins tiger:device-id="1" tiger:currency-id="USD" tiger:denom-
     id="10000"
     <tiger:Human.playerSimpleGmae tiger:device-id="1" tiger:denom-id="1000"
     tiger:primary-win="1000"/>
     <tiger:Human.createVoucher tiger:device-id="1" tiger:credit-type="CASHABLE"/>
     <tiger:Human.removeID tiger:device-id="1"/>
</tiger:repeat>
```

This snippet is similar to the previous example except the coins go to the hopper.

```
<tiger:repeat tiger:iterations="10">
     <tiger:Human.insertID tiger:device-id="1" tiger:id-number="12345678"/>
     <tiger:Human.insertNote tiger:device-id="1" tiger:currency-id="USD" tiger:denom-
     id="100000" tiger:note-action="DROP"/>
     <tiger:Human.insertCoins tiger:device-id="1" tiger:currency-id="USD" tiger:denom-
     id="10000" tiger:coin-action="HOPPER" tiger:count="4"/>
     <tiger:Human.playerSimpleGmae tiger:device-id="1" tiger:denom-id="1000"
     tiger:primary-win="1000"/>
     <tiger:Human.createVoucher tiger:device-id="1" tiger:credit-type="CASHABLE"/>
     <tiger:Human.removeID tiger:device-id="1"/>
</tiger:repeat>
```

tiger:then

The `tiger:then` building block contains the verbs that are executed when a parent element performs some sort of logic.

This building block may contain the following elements: core:exit, core:fail, core:include, core:log, core:metadata, core:parameters, core:pause, tiger:assert, tiger:DataModel.waitForDeviceState, tiger:DataModel.waitForEventQueueToDrain, tiger:DataModel.waitForHandpayKeyOff, tiger:Events.coinAcceptorEvents, tiger:Events.noteAcceptorEvents, tiger:Events.printerEvents, tiger:Human.cancelCancelCreditHandpay, tiger:Human.cashOut, tiger:Human.changeDoorState, tiger:Human.createVoucher, tiger:Human.dispenseCoins, tiger:Human.dispenseNotes, tiger:Human.doCoinDrop, tiger:Human.doNoteDrop, tiger:Human.getWATAccounts, tiger:Human.getWATBalance, tiger:Human.insertCoins, tiger:Human.insertID, tiger:Human.insertNote, tiger:Human.insertVoucher, tiger:Human.keyOff, tiger:Human.playSimpleGame, tiger:Human.removeID, tiger:Human.watToEGM, tiger:Human.watToHost, tiger:if-vendor, tiger:Progressive.getHostInfo, tiger:random, tiger:repeat, tiger:try-catch and tiger:WAT.getKeyPair.

This building block may be included in the following elements: `tiger:if-vendor` `and` `tiger:if-voucher-available`.

Example

In this snippet, if the EGM is from `XYZ` company, log that fact.

```
<tiger:if-vendor tiger:vendor="XYZ">
     <tiger:then>
       <core:log>This is an XYZ EGM.</core:log>
     </tiger:then>
</tiger:if-vendor>
```

core:parameters

The `core:parameters` verb lets you use pre-defined, references that can be changed from the **Execution Parameters** screen on the Tiger Script(s) user interface. Each parameter consists of a name-value pair that is referenced in one or more places in the script.

Loop Control Parameters

The Loop Control parameters can be used with the `tiger:repeat` statement and its *tiger:iterations* attribute *only*, effectively allowing you to repeat a block of commands as many times, or for as long as specified (format is Days, then HH:MM:SS, so "1 02:03:05" tells the loop to repeat for 1 day, 2 hours, 3 minutes, and 5 seconds).

Attribute	Restrictions	Description
loopIterations	type: integer	The number of times to repeat the loop
loopDuration	type: string	The length of time to execute the loop

Pre-defined Parameters

The following table lists the predefined parameters that are supported by Tiger:

Attribute	Restrictions	Description
toEmailAddressN	type: string default: " "	See Email Configuration
emailSubject	type: string default: " "	
smtpStatus	type: string	
smtpFromAddress	type: string	
smtpHost	type: string	
smtpPort	type: string	
smtpUsername	type: string	
smtpPassword	type: string	

Email Configuration

The RadBlue System Tester can be configured to send an e-mail *upon completing the script*. The e-mail is constructed by the tool and is sent out to users configured in the script.

Typically, you define the Simple Mail Transfer Protocol (SMTP) configuration in the user interface of the tool. However, these values can be supplied or overridden in the script. This makes it easy for an automated system to generate scripts that are completely configured.

The following table lists the parameters that are supported in the e-mail option:

Attribute	Restrictions	Description
toEmailAddressN	type: string default: " "	Defines one or more e-mail addresses. You can specify more than one e-mail address. This is done with a sequence number. The *N* is replaced by the integer sequence starting with **1**. When a gap is found the search for e-mail addresses stops. For example, this is how you would name the parameters to specify three e-mail addresses: • toEmail1 • toEmail2 • toEmail3 *This parameter is required*. At least one of these e-mail addresses must be defined in the script or the e-mail will not be sent.
emailSubject	type: string default: " "	Defines the subject of the e-mail. This can be any text. *This parameter is required*. If this parameter is missing, the e-mail will be sent without any subject.
smtpStatus	type: string default: " "	Defines the status of the e-mail feature. This text string must be one of the following values: • Never • Failure • Always If the value of the parameter is the empty string then the e-mail will not be sent. *This parameter is optional*. If it is present, the value of the parameter is used. If the parameter is

Attribute	Restrictions	Description
		not present, the value from the tool configuration is used. Use this parameter to override or supplant the value from the tool configuration.
smtpFromAddress	type: string	Defines the "from address" of the e-mail. This parameter may contain any valid e-mail address. If the value of the parameter is an empty string, the e-mail will not be sent. *This parameter is optional*. If it is present the value of the parameter is used. If the parameter is not present then the value from the tool configuration is used. Use this parameter to override or supplant the value from the tool configuration.
smtpHost	type: string	Defines the host name of the SMTP host. This parameter can be a DNS name or an IP address. The SMTP host will be used by the tool to deliver the e-mail. If the value of this parameter is an empty string, the e-mail will not be sent. *This parameter is optional*. If it is present, the value of the parameter is used. If the parameter is not present, the value from the tool configuration is used. Use this parameter to override or supplant the value from the tool configuration.
smtpPort	type: string	Defines the TCP/IP port of the SMTP host. If the value of this parameter is zero (0) or an illegal integer value, the e-mail will not be sent. *This parameter is optional*. If it is present, the value of the parameter is used. If the parameter is not present, the value from the tool configuration is used. Use this parameter to override or supplant the value from the tool configuration.
smtpUsername	type: string	Defines the user name to use when authenticating against the SMTP host. Some SMTP hosts require authentication. If the smtpHost specified requires authentication, use this parameter to specify the username. *This parameter is optional*. If it is present, the value of the parameter is used. If the parameter is not present, the value from the tool configuration

Attribute	Restrictions	Description
		is used. Use this parameter to override or supplant the value from the tool configuration.
smtpPassword	type: string	Defines the password to use when authenticating against the SMTP host. Some SMTP hosts require authentication. If the smtpHost specified requires authentication, use this parameter to specify the password. *This parameter is optional*. If it is present, the value of the parameter is used. If the parameter is not present, the value from the tool configuration is used. Use this parameter to override or supplant the value from the tool configuration.

Examples

This snippet defines a pair of loop control parameters for a script (4 times and 5 seconds). These values may be edited prior to running the script.

```
<core:parameters>
     <core:integerParameter core:name="loopIterations" core:default-value="4" />
     <core:stringParameter core:name="loopDuration" core:default-value="0 00:00:05" />
</core:parameters>
```

This snippet is a typical parameter set for a script that sends e-mail. In this example, the script is pre-defined to send e-mail to three users.

```
<core:parameters>
     <core:stringParameter core:name="toEmailAddress1"
       core:help-text="An email address" default-text="">
       user1@radblue.com
     </core:stringParameter>
     <core:stringParameter core:name="toEmailAddress2"
       core:help-text="An email address" default-text="">
       user2@radblue.com
     </core:stringParameter>
     <core:stringParameter core:name="toEmailAddress3"
       core:help-text="An email address" default-text="">
       user3@radblue.com
     </core:stringParameter>
     <core:stringParameter core:name="emailSubject"
       core:help-text="Subject of the email" default-text="The
       default email subject">
       The test has completed
     </core:stringParameter>
     <core:stringParameter core:name="smtpFromAddress"
       core:help-text="The from email address" default-text="">
       tool@radblue.com
     </core:stringParameter>
     <core:parameters>
```

This snippet is a full e-mail configuration that overrides whatever values were configured through the tool's user interface.

```
<core:parameters>
     <core:stringParameter core:name="toEmailAddress1"
       core:help-text="An email address" default-text="">
       user1@radblue.com
     </core:stringParameter>
     <core:stringParameter core:name="toEmailAddress2"
       core:help-text="An email address" default-text="">
       user2@radblue.com
     </core:stringParameter>
     <core:stringParameter core:name="toEmailAddress3"
       core:help-text="An email address" default-text="">
       user3@radblue.com
     </core:stringParameter>
     <core:stringParameter core:name="emailSubject"
       core:help-text="Subject of the email" default-text="The default
       email subject">
       The test has completed
     </core:stringParameter>
     <core:stringParameter core:name="smtpFromAddress"
       core:help-text="The from email address" default-text="">
       tool@radblue.com
     </core:stringParameter>
     <core:stringParameter core:name="smtpStatus"
       core:help-text="The SMTP Status" default-text="Never">
       Always
     </core:stringParameter>
     <core:stringParameter core:name="smtpHost" core:help-text="The
       SMTP Host" default-text="smtp.google.com">
       smtp.google.com
     </core:stringParameter>
     <core:stringParameter core:name="smtpPort" core:help-text="The
       SMTP Port" default-text="25">
       25
     </core:stringParameter>
     <core:stringParameter core:name="smtpUsername"
       core:help-text="The SMTP Authentication Username"
       default-text="username">
       billybob
     </core:stringParameter>
     <core:stringParameter core:name="smtpPassword"
       core:help-text="The SMTP Authentication Password"
       default-text="password">
       chevy
     </core:stringParameter>
<core:parameters>
```

core:pause

The `core:pause` verb is used to pause the script. The `core:pause` verb can wait a fixed number of seconds or a random number of seconds.

Attribute	Restrictions	Description
core:duration	type: string use: optional minLength: 1 default: "[0,1000]"	The `core:duration` attribute controls how long the script should pause. Accepted formats are: • **Positive integer value** - Sleep for the defined number of milliseconds. • **A value of the form [A,B]** - Sleep a random number of milliseconds in the range [A,B].

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet causes the script to pause for a random value between 0 and 1000 milliseconds.

```
<core:pause/>
```

This snippet causes the script to pause for 5000 milliseconds.

```
<core:pause core:duration="5000"/>
```

This snippet causes the script to pause for a random value between 1000 and 5000 milliseconds.

```
<core:pause core:duration="[1000,5000]"/>
```

tiger:action

The `tiger:action` is a child element of the `tiger:random` verb. This element contains the verbs that are to be executed if this action is randomly selected by the parent verb.

See also `tiger:random`

Attribute

Attribute	Restrictions	Description
tiger:weight	type: integer use: optional default: "100"	The weight value to assign to this action. The weights of all of the actions are used by the parent element to make a random choice.

This element may contain the following elements: core:config, core:exit, core:fail, core:include, core:log, core:metadata, core:parameters, core:pause, tiger:assert, tiger:Cabinet.getCashableCreditMeter, tiger:DataModel.waitForDeviceState, tiger:DataModel.waitForEventQueueToDrain, tiger:DataModel.waitForHandpayKeyOff, tiger:Events.coinAcceptorEvents, tiger:Events.noteAcceptorEvents, tiger:Events.printerEvents, tiger:Human.cancelCancelCreditHandpay, tiger:Human.cashOut, tiger:Human.changeDoorState, tiger:Human.createVoucher, tiger:Human.dispenseCoins, tiger:Human.dispenseNotes, tiger:Human.doCoinDrop, tiger:Human.doNoteDrop, tiger:Human.getWATAccounts, tiger:Human.getWATBalance, tiger:Human.insertCoins, tiger:Human.insertID, tiger:Human.insertNote, tiger:Human.insertVoucher, tiger:Human.keyOff, tiger:Human.playSimpleGame, tiger:Human.removeID, tiger:Human.watToEGM, tiger:Human.watToHost, tiger:if-vendor, tiger:Progressive.getHostInfo, tiger:random, tiger:repeat, tiger:try-catch and tiger:WAT.getKeyPair.

Example

This snippet randomly selects one of the action blocks to execute.

```
<tiger:random>
     <tiger:action tiger:weight="50">
       <tiger:Human.insertNote tiger:device-id="1" tiger:currency-id="USD" tiger:denom-
       id="100000" tiger:note-action="DROP"/>
     </tiger:action>
     <tiger:action tiger:weight="5">
       <tiger:pause/>
     </tiger:action>
     <tiger:action>
       <tiger:pause/>
     </tiger:action>
     <tiger:action>
       <tiger:Human.playSimpleGame tiger:device-id="7" tiger:denom-id="25000"
       tiger:primary-win="1" tiger:credits-to-wager-cashable="2"/>
     </tiger:action>
</tiger:random>
```

tiger:DataModel.clearVoucherDatabase

The `tiger:DataModel.clearVoucherDatabase` verb causes the SmartEGM to clear the voucher database (remove all vouchers) for the specified database name.

IMPORTANT NOTE!
If you use your own voucher database file, you must place the file in the **/user-data/id-<name>.xml** location.

For RST, the user-data directory **will** be deleted when the tool is uninstalled.
For RLT, the user-data directory **will not** be deleted when the tool is uninstalled.

Attribute

Attribute	Restrictions	Description
tiger:database-name	type: string use: required	Specifies the database name that will be used to retrieve IDs from. The database maps to a **../conf/id-${database-name}.xml** in the local file system.

Examples

This snippet causes the SmartEGM to clear the voucher database named "rst" of all voucher records.

```
<tiger:DataModel.clearVoucherDatabase tiger:database-name="rst"/>
```

tiger:DataModel.snapshot

The `tiger:DataModel.snapshot` verb causes the SmartEGM to take a snapshot of the EGM's data model and store it under the specified name. The snapshot can then be viewed in the tool's snapshot viewer.

Attribute

Attribute	Restrictions	Description
tiger:name	type: string use: required	Name of the snapshot.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet causes the SmartEGM to store the current data model to a snapshot with the name of "shapshot1". The named snapshot can be viewed in the tool's snapshot viewer.

```
<tiger:DataModel.snapshot tiger:name="snapshot1"/>
```

tiger:DataModel.waitForDeviceState

The `tiger:DataModel.waitForDeviceState` verb causes the script to wait until the *`hostEnabled`* attribute changes for the given device. A typical use for this verb is to have the script wait for the host to enable a device, say the cabinet device.

Note that not all devices allow their *hostEnabled* attribute to be set or changed. These devices can be used with this verb, but you will always get a time-out.

Attributes

Attribute	Restrictions	Description
tiger:device-class	type: tiger:device-class use: required	G2S device class of the device in use.
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first device from the specified device class. • A value of "-1" is legal but should not be used because it will result in a failure.
tiger:expected-value	type: boolean use: optional default: "true"	Expected value of the *hostEnabled* attribute for the selected device.
tiger:timeout	type: integer use: optional default: "60000"	Number of milliseconds to wait for the device to achieve the expected value.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Examples

This snippet causes the script to wait for the host to set the *hostEnabled* attribute of the first cabinet device to true. The script waits for a maximum of 60 seconds.

```
<tiger:DataModel.waitForDeviceState tiger:device-class="G2S_Cabinet"/>
```

This snippet causes the script to wait for the host to set the *hostEnabled* attribute of the hopper with device ID 2 to true. The script waits for a maximum of 60 seconds.

```
<tiger:DataModel.waitForDeviceState tiger:device-class="G2S_hopper" tiger:device-
id="2"/>
```

tiger:DataModel.waitForEventQueueToDrain

The `tiger:DataModel.waitForEventQueueToDrain` verb causes the script to wait until the outbound event queue has fully drained or a timeout occurs.

Per the G2S specification, the SmartEGM has an outbound queue of events. Depending on various factors (script complexity, event subscriptions, CPU and memory availability, network bandwidth and host performance) the queue may build up. It often makes sense to make the script wait for the event queue to drain. This verb is used to perform such a wait.

As a general guideline, this verb should be invoked in any script that runs a long time and generates a lot of events. The more events in the event queue the slower the SmartEGM will run.

Attributes

Attribute	Restrictions	Description
tiger:timeout	type: integer use: optional default: "60000"	Number of milliseconds to wait for the queue to drain.
tiger:post-delay	type: integer use: optional default: "0"	Delay, in milliseconds, to wait after he queue has drained.
tiger:pre-delay	type: integer use: optional default: "0"	Delay, in milliseconds, to wait before the tiger:timeout delay begins.
tiger:sleep-interval	type: integer use: optional default: "1000"	Frequency, in millisecond, for checking whether the event queue has drained.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Examples

This snippet causes the script to wait for the event queue to drain. The script waits a maximum of 60 seconds.

```
<tiger:DataModel.waitForEventQueueToDrain/>
```

This snippet causes the script to wait 3 seconds, then it waits a maximum of 30 seconds for the event queue to drain (checking every 0.5 seconds), and then afterwards, it waits an additional 1 second before the script execution continues.

```
<tiger:DataModel.waitForEventQueueToDrain tiger:timeout="30000" tiger:pre-delay="3000"
     tiger:post-delay="1000" tiger:sleep-interval="500" />
```

tiger:DataModel.waitForHandpayKeyOff

The `tiger:DataModel.waitForHandpayKeyOff` verb causes the script to wait until the outstanding handpay event has been keyed off. This verb is intended to cause the EGM to wait until the host can send a `Handpay.setRemoteKeyOff` command.

Attribute

Attribute	Restrictions	Description
tiger:timeout	type: integer use: optional default: "60000"	Number of milliseconds to wait for the handpay keyoff.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Examples

This snippet causes the script to wait for the current handpay event to be keyed off. The script waits a maximum of 60 seconds.

```
<tiger:DataModel.waitForHandpayKeyOff/>
```

This snippet causes the script to wait for the current handpay event to be keyed off. The script waits a maximum of 30 seconds.

```
<tiger:DataModel.waitForEventQueueToDrain tiger:timeout="30000"/>
```

tiger:Egm.resetEgm

The `tiger:Egm.resetEgm` verb resets the EGM.

Attribute

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first cabinet device. • A value of "-1" is legal, but should not be used because it will result in a failure.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet inserts one $1.00 USD note into the first note acceptor device. The note is sent to the drop.

```
<tiger:Human.insertNote tiger:device-id="-2" tiger:currency-id="USD" tiger:denom-
id="100000" tiger:note
      action="DROP"/>
```

tiger:Events.cabinetEvents

The `tiger:Events.cabinetEvents` verb simulates the generation of Cabinet device related events.

Attribute

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	EGM device identifier.
tiger:setEvent	type: setCabinetEvent	Cabinet door event to be simulated.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Examples

This snippet clears the Service Lamp On on the first Cabinet device.

```
<tiger:Events.cabinetEvents>
     <tiger:setEvent tiger:event="SERVICE_LAMP_ON"/>
</tiger:Events.cabinetEvents>
```

This snippet generates the SERVICE_LAMP_ON event in the first Cabinet device.

```
<tiger:Events.cabinetEvents>
     <tiger:setEvent tiger:event="SERVICE_LAMP_OFF"/>
</tiger:Events.cabinetEvents>
```

This snippet generates the CABINET_DOOR_CLOSED event in the first Cabinet device.

```
<tiger:Events.cabinetEvents>
     <tiger:setEvent tiger:event="CABINET_DOOR_OPENED"/>
</tiger:Events.cabinetEvents>
```

This snippet generates the CABINET_DOOR_OPENED event in the first Cabinet device.

```
<tiger:Events.cabinetEvents>
     <tiger:setEvent tiger:event="CABINET_DOOR_CLOSED"/>
</tiger:Events.cabinetEvents>
```

tiger:Events.coinAcceptorEvents

The `tiger:Events.coinAcceptorEvents` verb simulates the generation of events related to coin acceptor devices.

This verb may contain the `tiger:setEvent` element. Note that the child elements of `tiger:setEvent` map directly to the coin acceptor events from the G2S specification.

Attribute

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first coin acceptor device. • A value of "-1" is legal, but should not be used because it will result in a failure.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Element

Element	Restrictions	Description
tiger:setEvent	type:SetCoinAcceptorEvent	Coin acceptor event to be simulated.

Examples

This snippet generates the ACCEPTOR_JAMMED event in the first coin acceptor device.

```
<tiger:Events.coinAcceptorEvents>
     <tiger:setEvent tiger:event="ACCEPTOR_JAMMED"/>
</tiger:Events.coinAcceptorEvents>
```

This snippet clears all outstanding events on the first coin acceptor first device.

```
<tiger:Events.coinAcceptorEvents>
     <tiger:setEvent tiger:event="CLEAR_ALL_FAULTS"/>
</tiger:Events.coinAcceptorEvents>
```

This generates the COIN_DROP_DOOR_OPEN event in the first coin acceptor device.

```
<tiger:Events.coinAcceptorEvents>
     <tiger:setEvent tiger:event="COIN_DROP_DOOR_OPEN"/>
</tiger:Events.coinAcceptorEvents>
```

This generates the COIN_DROP_DOOR_CLOSED event in the first coin acceptor device.

```
<tiger:Events.coinAcceptorEvents>
     <tiger:setEvent tiger:event="COIN_DROP_DOOR_CLOSED"/>
</tiger:Events.coinAcceptorEvents>
```

tiger:Events.noteAcceptorEvents

The `tiger:Events.NoteAcceptorEvents` command defines the note acceptor event. This verb allows you to generate events related to a note acceptor device. In G2S, this would be events such as door open or mechanical fault.

This verb may contain the `tiger:setEvent` element. Note that the child elements of `tiger:setEvent` map directly to the note acceptor events from the G2S specification.

Attribute

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first note acceptor device. • A value of "-1" is legal, but should not be used because it will result in a failure.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Element

Element	Restrictions	Description
tiger:setEvent	type:SetNoteAcceptorEvent	Note acceptor event to be simulated.

Examples

This snippet generates a stacker jam event for the first note acceptor device.

```
<tiger:Events.noteAcceptorEvents>
     <tiger:setEvent tiger:event="STACKER_JAM" />
</tiger:Events.noteAcceptorEvent>
```

This snippet clears all existing faults on the first note acceptor device.

```
<tiger:Events.noteAcceptorEvents>
     <tiger:setEvent tiger:event="CLEAR_ALL_FAULTS" />
</tiger:Events.noteAcceptorEvent>
```

tiger:Events.printerEvents

The `tiger:Events.printerEvents` verb simulates the generation of events related to printer devices.

This verb may contain the `tiger:setEvent` statement. Note that the child elements of `tiger:setEvent` map directly to the printer events from the G2S specification.

Attribute

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first printer device. • A value of "-1" is legal, but should not be used because it will result in a failure

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Element

Element	Restrictions	Description
tiger:setEvent	type:SetPrinterEvent	Printer event to be simulated.

Examples

This snippet generates a printer jam event in the first printer device.

```
<tiger:Events.printerEvents>
     <tiger:setEvent tiger:event="PRINTER_JAM"/>
</tiger:Events.printerEvents>
```

This snippet clears all outstanding events on the first printer first device.

```
<tiger:Events.printerEvents>
       <tiger:setEvent tiger:event="CLEAR_ALL_FAULTS"/>
</tiger:Events.printerEvents>
```

This generates the PRINTER_DISCONNECTED event in the first printer device.

```
    <tiger:Events.printerEvents>
         <tiger:setEvent tiger:event="PRINTER_DISCONNECTED"/>
    </tiger:Events.printerEvents>
```

This generates the PRINTER_CONNECTED event in the first printer device.

```
    <tiger:Events.printerEvents>
         <tiger:setEvent tiger:event="PRINTER_CONNECTED"/>
    </tiger:Events.printerEvents>
```

tiger:Human.cancelCancelCreditHandpay

The `tiger:Human.cancelCancelCreditHandpay` verb simulates a player canceling a pending cancel credit handpay.

Attribute

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device Identifier. • A value of "-2" means the first Handpay device. • A value of "-1" is legal, but should not be used because it will result in a failure.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet cancels the pending cancel credit handpay on the first handpay device.

```
    <tiger:Human.cancelCancelCreditHandpay/>
```

tiger:Human.cashOut

The `tiger:Human.cashOut` verb simulates a player pressing the cash out button. A cancel credit handpay is currently the only supported pay out method.

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device Identifier. • A value of "-2" means the first handpay device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:method	type: tiger:method use: optional default: "HANDPAY"	Defines the method for cashing out to a Cancel Credit Handpay.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet sends all funds on the credit meter to the first handpay device.

```
<tiger:Human.cashOut tiger:method="HANDPAY"/>
```

tiger:Human.changeDoorState

The `Human.changeDoorState` verb simulates the opening and closing of EGM cabinet doors (cabinet, logic or auxiliary).

Attributes

Attribute	Restrictions	Description
tiger:action	type: tiger:doorAction use: optional default: "CLOSE"	Action to take on the door (open or close).
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the cabinet device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:doorName	type: tiger:door-name use: optional default: "CABINET"	Door to open or close.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet opens the logic door of the first cabinet device.

```
<tiger:Human.changeDoorState tiger:doorName="LOGIC" tiger:action="OPEN"/>
```

tiger:Human.createVoucher

The `tiger:Human.createVoucher` verb simulates the creation of a voucher.

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first voucher device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:id-reader-device-id	type: integer use: optional default: "-2"	The ID of the ID reader device where an ID may be found. A value of "-2" means the first ID Reader device. A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:credit-type	type: tiger:credit-type use: optional default: "CASHABLE"	The type of credits to be used in creating the voucher. The value is removed from the appropriate credit meter.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet takes all of the current credits on the cashable credit meter, using the first voucher device, and creates a voucher with that value.

```
    <tiger:Human.createVoucher/>
```

tiger:Human.createVoucherToDatabase

The `tiger:Human.createVoucherToDatabase` verb simulates the creation of a voucher. The voucher's validation number is then stored in a database for later processing.

This verb allows you to specify the name of the voucher database to use. This allows you to have more than one database and switch between them on a verb-by-verb basis.

IMPORTANT NOTE!
If you use your own voucher database file, you must place the file in the **/user-data/voucher-<name>.xml** location.

For RST, the user-data directory **will** be deleted when the tool is uninstalled.
For RLT, the user-data directory **will not** be deleted when the tool is uninstalled.

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first Voucher device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:id-reader-device-id	type: integer use: optional default: "-2"	ID Reader Device identifier. • A value of "-2" means the first ID reader device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:credit-type	type: tiger:credit-type use: optional default: "cashable"	Type of credits to be used in creating the voucher. This value is taken off of the appropriate credit meter.
tiger:database-name	type: string use: required	Specifies the database name where validation numbers are stored. . The database maps to a **../conf/voucher-${database-name}.xml** in the local file system.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet takes all of the current credits on the cashable credit meter, using the first voucher device, and creates a voucher with that value. The voucher's validation number is then stored in a voucher database called `../conf/voucher-test.xml`.

```
    <tiger:Human.createVoucherToDatabase tiger:database-name="test"/>
```

tiger:Human.dispenseCoins

The `tiger:Human.dispenseCoins` verb dispenses *x* coins of the specified denomination from a hopper.

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first Hopper device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:all	type: boolean use: optional default: "true"	If true, as many coins as possible are dispensed (in the same way a real EGM functions).
tiger:count	type: integer use: optional default: "1"	Number of coins to be dispensed.
tiger:credit-type	type: tiger:credit-type use: optional default: "CASHABLE"	Credit meter to reference when dispensing coins. The value is removed from the appropriate credit meter.
tiger:currency-id	type: string use: optional default: "USD"	Currency identifier of the dispensed coin.
tiger:denom-id	type: long use: required	Denomination identifier of the dispensed coin.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Examples

This snippet dispenses as many $0.25 USD cashable coins as possible, using the first hopper device. The cashable credit meter is updated accordingly.

```
<tiger:Human.dispenseCoins tiger:currency-id="USD" tiger:credit-type="CASHABLE"
tiger:denom-id="25000"/
```

This snippet dispenses five $0.25 USD cashable coins, using the first hopper device. The cashable credit meter is updated accordingly.

```
<tiger:Human.dispenseCoins tiger:currency-id="USD" tiger:credit-type="CASHABLE"
tiger:denom-id="25000" tiger:count="5"/
```

tiger:Human.dispenseNotes

The `tiger:Human.dispenseNotes` verb dispenses *x* notes of the specified denomination from a note dispenser.

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first note dispenser device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:all	type: boolean use: optional default: "true"	If true, as many notes as possible are dispensed (in the same way a real EGM functions).
tiger:count	type: integer use: optional default: "1"	Number of notes to be dispensed.
tiger:credit-type	types: tiger:credit-type use: optional default: "CASHABLE"	Type of credits to be used when dispensing notes. The value is removed from the appropriate credit meter.
tiger:currency-id	type: string use: optional default: "USD"	Currency identifier of the dispensed note.
tiger:denom-id	type: long use: required	Denomination identifier of the dispensed note.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Examples

This snippet dispenses as many $1.00 USD cashable notes as possible, using the first note dispenser device. The cashable credit meter is updated accordingly.

```
<tiger:Human.dispenseNotes tiger:currency-id="USD" tiger:credit-type="CASHABLE"
tiger:denom-id="100000"/>
```

This snippet dispenses five $1.00 USD cashable notes, using the first note dispenser device. The cashable credit meter is updated accordingly.

```
<tiger:Human.dispenseNotes tiger:currency-id="USD" tiger:credit-type="CASHABLE"
tiger:denom-id="100000" tiger:count="5"/>
```

tiger:Human.doCoinDrop

The `tiger:Human.doCoinDrop` verb executes a coin drop sequence: drop door open and door closed events. The `tiger:Human.doCoinDrop` verb simulates the performing of a coin drop.

Attribute

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first coin acceptor device. • A value of "-1" is legal, but should not be used because it will result in a failure.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet performs a coin drop using the first coin acceptor device.

```
    <tiger:Human.doCoinDrop/>
```

tiger:Human.doNoteDrop

The `tiger:Human.doNoteDrop` verb executes a note drop sequence: Note door open, stacker out, stacker in, and door closed events.

Attribute

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first note acceptor device. • A value of "-1" is legal, but should not be used because it will result in a failure.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet performs a note drop for the first note acceptor.

```
    <tiger:Human.doNoteDrop/>
```

tiger:Human.enterPIN

The `tiger:Human.enterPIN` verb allows you to enter a smart card PIN. If the smart card is negotiable, the verbs in the `then` block are executed. If the smart card is not negotiable, the verbs in the `else` block are executed.

Attribute

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first ID reader device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:pin	type: string use: optional default: " "	PIN for the smart card. If a value is not specified, the correct value for the smart card will be used.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet enters the smart card PIN if the smart card is not negotiable.

```
<tiger:if-smart-card-negotiable>
     <tiger:then>
     </tiger:else> <tiger:Human.enterPIN/>
</tiger:if-smart-card-negotiable>
```

tiger:Human.getWATAccounts

The `tiger:Human.getWATAccounts` verb simulates a request by a player for a list of wagering accounts that are available to that player. This command uses the player's account information from the ID reader device that is configured in the `wat.watProfile` command.

See also: `tiger:Human.GetWATBalance`

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first WAT device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:authentication	type: string use: optional default: " "	Authentication used by the player, for example, a Personal Identification Number (PIN). This value is un-encrypted. The authentication attribute will be encrypted using the *hashType* as specified in the `wat.watProfile` command.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet requests all wagering accounts for the player associated with the WAT device's ID reader, using the first WAT device. A player PIN of 1234 is encrypted, based on the `wat.watProfile` hash type.

```
    <tiger:Human.getWATAccounts tiger:authentication="1234"/>
```

tiger:Human.getWATBalance

The `tiger:Human.getWATBalance` verb simulates a request by a player for the available balance from the specified wagering account. This command uses the player's account information from the ID Reader device that is configured in the `wat.watProfile`.

See also: `tiger:Human.GetWATAccounts`

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first WATdevice. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:accountId	type: string use: optional default: "-2"	Player's wagering account identifier. If no account is specified, the first account returned by Human.getWATAccounts is used.
tiger:authentication	type: string use: optional default: " "	Authentication used by the player, for example, a Personal Identification Number (PIN). This value is un-encrypted. The authentication attribute will be encrypted using the *hashType* as specified in the `wat.watProfile` command.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Examples

This snippet requests the balance of the first account for the player associated with the WAT device's ID reader, using the first WAT device. The player PIN of 1234 is encrypted, based on the `wat.watProfile` command's *hashType* attribute.

```
    <tiger:Human.getWATBalance tiger:authentication="1234"/>
```

This snippet requests the balance of account number "12345678" for the player associated with the WAT device's ID reader. The player PIN of 1234 is encrypted, based on the `wat.watProfile` command's *hashType* attribute.

```
    <tiger:Human.getWATAccounts tiger:accountId="12345678" tiger:authentication="1234"/>
```

tiger:Human.insertCoins

The `tiger:Human.insertCoins` verb inserts coins into the EGM. This verb simulates the player inserting one or more coins (or tokens) into a coin acceptor device.

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first coin acceptor device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:accept-inappropriate-coin	type: integer use: optional default: "-2"	Determines if these coins should be considered inappropriate and handled accordingly.
tiger:coin-action	type: tiger:coin-action use: optional default: "DROP"	Authentication used by the player (for example, a PIN). This value is un-encrypted. The authentication attribute is encrypted using the *hashType* attribute as specified in the `wat.watProfile` command.
tiger:count	type: integer use: optional default: "1"	Number of notes to be dispensed.
tiger:currency-id	type: string use: optional default: "USD"	Currency identifier of the inserted coin.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet inserts four $0.10 USD coins into the first coin acceptor device. The coins are sent to the hopper.

```
<tiger:Human.insertCoins tiger:device-id="-2" tiger:currency-id="USD" tiger:denom-
id="10000" tiger:coin
     action="HOPPER" tiger:count="4"/>
```

tiger:Human.insertID

The `tiger:Human.insertID` verb inserts an ID into the specified ID reader.

See also: `tiger:Human.removeID`

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first ID reader device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:id-number	type: string use: required	This value can come from a magnetic card, RFID chip, etc.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet inserts the ID with the value of "12345678", using the first ID reader device.

```
<tiger:Human.insertID tiger:device-id="-2" tiger:id-number="12345678"/>
```

tiger:Human.insertIDFromDatabase

The `tiger:Human.insertIDFromDatabase` verb simulates the insertion of an ID into an ID reader device. The ID number used is read from the specified external XML database.

> **IMPORTANT NOTE!**
> If you use your own ID database file, you must place the file in the **/user-data/id-<name>.xml** location.
>
> For RST, the user-data directory **will** be deleted when the tool is uninstalled.
> For RLT, the user-data directory **will not** be deleted when the tool is uninstalled.

See also: `tiger:Human.removeIDToDatabase`

Attributes

Attribute	Restrictions	Description
See "tiger:Human.insertVoucherFromDatabase" on page 111	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first ID Reader device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:database-name	type: string use: required	Specifies the database name that will be used to retrieve IDs from. The database maps to a **../conf/id-${database-name}.xml** in the local file system.
tiger:lock-id	type: boolean use: optional default: "true"	Indicates whether the specified identifier is locked. A locked ID cannot be reused until it is removed from the ID reader. If you set `tiger:lock-id` to **false**, the Tiger script may reuse that ID in a different EGM when the script is run from the RadBlue Load Tester (RLT).

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet inserts the ID number read from the `../conf/id-goldCards.xml` database, using the first ID reader device. The ID number is locked in the database and cannot be reused until it is removed from the ID reader device.

```
<tiger:Human.insertIDFromDatabase tiger:device-id="-2" tiger:database-name="goldCards"
tiger:lock="true"/>
```

tiger:Human.insertNote

The `tiger:Human.insertNote` verb simulates the player inserting a note (or bill) into a note acceptor device.

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first note acceptor device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:currency-id	type: string use:optional default: "USD"	Currency identifier of inserted note.
tiger:denom-id	type: long use: required	Denomination identifier of inserted note.
tiger:note-action	type: tiger:noteAction use: optional default: "DROP"	Action to be taken on inserted note.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet inserts one $1.00 USD note into the first note acceptor device. The note is sent to the drop.

```
<tiger:Human.insertNote tiger:currency-id="USD" tiger:denom-id="100000"
    tiger:note-action="DROP"/>
```

tiger:Human.insertVoucher

The `tiger:Human.insertVoucher` verb simulates the insertion of a voucher for redemption.

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first Voucher device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:id-reader-device-id	type: integer use: optional default: "-2"	The ID of the ID reader device where an ID may be found. A value of "-2" means the first ID Reader device. A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:note-acceptor-device-id	type: integer use: optional default: "-2"	Identifier of the note acceptor device where the voucher was inserted. • A value of "-2" means the first note acceptor device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:voucher-action	type: tiger:voucherAction use: optional default: "DROP"	Action to be taken on the inserted voucher.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet inserts the voucher with validation ID "123456789012345678", using the first voucher device, the first note acceptor device and the first ID reader device. The voucher will go to the drop.

```
<tiger:Human.insertVoucher tiger:validationId="123456789012345678"/>
```

tiger:Human.insertVoucherFromDatabase

The `tiger:Human.insertVoucherFromDatabase` verb simulates the insertion of a voucher for redemption. The voucher validation ID comes from the specified database.

IMPORTANT NOTE!
If you use your own voucher database file, you must place the file in the **/user-data/voucher-<name>.xml** location.

For RST, the user-data directory **will** be deleted when the tool is uninstalled.
For RLT, the user-data directory **will not** be deleted when the tool is uninstalled.

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first Voucher device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:id-reader-device-id	type: integer use: optional default: "-2"	The ID of the ID reader device where an ID may be found. A value of "-2" means the first ID Reader device. A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:note-acceptor-device-id	type: integer use: optional default: "-2"	Identifier of the note acceptor device where the voucher was inserted. • A value of "-2" means the first note acceptor device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:voucher-action	type: tiger:voucherAction use: optional default: "DROP"	Action to be taken on the inserted note.
tiger:database-name	type: string use: required	Specifies the database name that will be used to retrieve validation numbers from. The database maps to a **../conf/voucher-${database-name}.xml** in the local file system.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet inserts a voucher using a validation number obtained from the voucher database (`../conf/voucher-test.xml`), using the first voucher device, the first note acceptor device and the first ID reader device. The voucher will go to the drop.

```
    <tiger:Human.insertVoucherFromDatabase tiger:database-name="test"/>
```

tiger:Human.KeyOff

The `tiger:Human.keyOff` verb simulates an EGM attendant clearing a handpay condition by selecting the method to pay out the handpay.

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first Handpay device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:method	enumeration: tiger:KeyOffAction use: optional default: "CREDIT"	Method used to key off a handpay.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet keys off to vouchers all outstanding handpay transactions.

```
<tiger:Human.keyOff tiger:method="VOUCHER"/>
```

tiger:Human.playPaytableGame

The `tiger:Human.playPaytableGame` verb simulates game play using paytables to determine game outcomes. This verb is used primarily to test the Game Outcome extension of the G2S protocol. Attributes and elements related to the Game Outcome extension appear in **bold**.

Attributes

Attribute	Restrictions	Description
credits-to-wager-cashable	type: positive integer use: optional default: "0"	Number of cashable credits to wager.
credits-to-wager-non-cashable	type: positive integer use: optional default: "0"	Number of non-cashable credits to wager.
credits-to-wager-promo	type: positive integer use: optional default: "0"	Number of promotional credits to wager.
denom-id	type: long use: required	Identifier of the denomination to be wagered.
tiger:device-id	type: integer use: optional default: "-2"	Device Identifier. • A value of "-2" means the first gamePlay device. • A value of "-1" is legal, but should not be used because it will result in a failure.
final-hand-count	**type: positive integer** **use: optional** **default: "1"**	**Number of final hands to generate.**
handpay-action	type: tiger:KeyOffAction use: optional default: "HANDPAY"	Indicates how the handpay should be paid.
in-game-delay	type: positive integer use: optional default: "0"	Defines an in-game delay, in milliseconds, that is executed after `GPE101` and after `GPE109`. Use this delay to simulate complex game mechanics.
tiger:key-off-timeout	type: integer use: optional default: "0"	How long to wait for a `setRemoteKeyOff` command before timing out. If a time out occurs, the `handpay-action` value is used.
play-secondary-game-count	type: positive integer use: optional default: "0"	Number of double-or-nothing secondary games to play if the game's calculated initial win is not zero. The first secondary game takes the game's initial win value as the amount wagered. For all but the

Attribute	Restrictions	Description
		last secondary game, the secondary amount won will equal twice the amount wagered.
remote-key-off-timeout	type:positive integer use: optional default: "60000"	**DEPRECATED 03 OCT 2012 - Version 24** **Replaced by `tiger:key-off-timeout`** Indicates how long to wait for the remote key off.
seed	**type: integer** **use: optional** **default: "-1"**	**Seed for the random number generator. Specifying a seed number provides a consistent series of game outcomes.**
win-final-secondary-game	type: boolean use: optional default: "false"	Indicates whether the final secondary game is a win or a loss. If `win-final-secondary-game` is set to **false**, the final win is zero (0).
win-to-handpay	type: boolean use: optional default: "false"	Indicates whether the game win goes to a handpay.

Element

Note: If this element is not included, the SmartEGM will always create random outcomes.

Attribute	Restrictions	Description
outcome	**type: tiger:OutcomeRecordType** **minOccurs: 0** **maxOccurs: 1**	**Indicates the use of the outcome.xml file to determine the game outcome.**

tiger:OutcomeRecordType Attributes

Attribute	Restrictions	Description
outcome-random	type: boolean use: optional default: "false"	Indicates whether the outcome record is selected randomly from the outcome.xml file. If this value is true then outcome-record-name is ignored.
outcome-record-name	type: string optional default: " "	Name of selected outcome record from the outcome.xml file.

This verb may be included in: tiger:action, tiger:catch, , tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Examples

1. **Poker game with a specified outcome.**

```
<tiger:Human.playPaytableGame
       tiger:credits-to-wager-cashable=”50”
       tiger:denom-id=”25000”
       tiger:device-id=”2”
       tiger:final-hand-count=”50”>
     <tiger:outcome tiger:outcome-record-name=”outcome-001” />
</tiger:Human.playPaytableGame>
```

Wager $12.50 on game play device 2. The initial and first final hands are specified in the outcome-list.xml file under the name “outcome-001”. The second through fiftieth final hands are randomly generated from a standard 52 card deck where the initial cards have been removed. See discussion about final hand generation.

The primary win is calculated by apportioning the total credits-to-wager by the number of outcome hands to each individual final hand and looking up the wager multiplier for each winning hand.

2. **Poker game with a randomly selected outcome.**

```
<tiger:Human.playPaytableGame
       tiger:credits-to-wager-cashable=”5”
       tiger:denom-id=”100000”
       tiger:device-id=”2” >
     <tiger:pokerGame tiger:outcome-random=”true” />
</tiger:Human.playPaytableGame>
```

Wager $5.00 on game play device 2. The initial hand and final hand is randomly selected from the outcome-list.xml file. Each record in the outcome-list.xml file has a relative weight that is used in the selection of records from outcome-list.xml file.

The primary win is calculated by determining if the final hand is a winning hand and looking up the wager multiplier for that hand.

3. **Poker game where no outcomes are specified.**

```
<tiger:Human.playPaytableGame
     tiger:credits-to-wager-cashable=”5”
     tiger:denom-id=”100000”
     tiger:device-id=”2”
     tiger:final-hand-count=”50” />
```

Wager $5.00 on game play device 2. The initial hand is randomly generated from a standard 52 card deck. If the initial hand is a winning hand, those cards that make up the winning hand are held. The first through fiftieth final hands are then randomly generated. See discussion about final hand generation.

tiger:Human.playSimpleCentralGame

The `tiger:Human.playSimpleCentralGame` verb simulates the player playing a central determination game, in which the outcome comes from a central determination server.

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first gamePlay device that is configured for central determination. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:credits-to-wager-cashable	type: integer use: optional default: "0"	Number of cashable credits to wager.
tiger:credits-to-wager-non-cashable	type: integer use: optional default: "0"	Number of non-cashable credits to wager.
tiger:credits-to-wager-promo	type: integer use: optional default: "0"	Number of promotional credits to wager.
tiger:denom-id	type: long use: required	Identifier of denomination to be wagered.
tiger:handpay-action	type: tiger:KeyOffAction use: optional default: "HANDPAY"	Indicates how handpay should be paid.
tiger:in-game-delay	type: integer use: optional default: "0"	Defines an in-game delay (in milliseconds) that is executed after GPE101 (Primary Game Escrow) and after GPE109 (Secondary Game Started). Use this delay to simulate complex game mechanics.
tiger:key-off-timeout	type: integer use: optional default: "0"	How long to wait for a `setRemoteKeyOff` command before timing out. If a time out occurs, the `handpay-action` value is used.
tiger:play-secondary-game-count	type: integer use: optional default: "0"	Number of double-or-nothing secondary games to attempt to play if the primary-win is not zero. For each secondary game play, the SmartEGM will wager all of the winnings up to that point, and will continue until either the amount won equals zero or the number of games to play.

Attribute	Restrictions	Description
tiger:remote-key-off-timeout	type: integer use: optional default: "60000"	**DEPRECATED 03 OCT 2012 - Version 24** **Replaced by `tiger:key-off-timeout`** Time, in milliseconds, to wait for a remote keyoff.
tiger:win-to-handpay	type: boolean use: optional default: "false"	Indicates whether the game win should go to a handpay.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet plays a simple game on Game Play Device 7, where two $0.25 cashable credits are wagered. The outcome of the game is determined by the central host.

```
    tiger:Human.playSimpleCentralGame tiger:device-id="7" tiger:denom-id="25000"
    tiger:credits-to-wager-cashable="2"/>
```

tiger:Human.playSimpleGame

The `tiger:Human.playSimpleGame` verb simulates a player playing a game at a standard EGM.

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first gamePlay device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:credits-to-wager-cashable	type: integer use: optional default: "0"	Number of cashable credits to wager.
tiger:credits-to-wager-non-cashable	type: integer use: optional default: "0"	Number of non-cashable credits to wager.
tiger:credits-to-wager-promo	type: integer use: optional default: "0"	Number of promo credits to wager.
tiger:denom-id	type: long use: required	Denomination ID to be wagered.
tiger:handpay-action	type: tiger:KeyOffAction use: optional default: "HANDPAY"	Method of paying handpay.
tiger:in-game-delay	type: integer use: optional default: "0"	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.
tiger:key-off-timeout	type: integer use: optional default: "0"	How long to wait for a `setRemoteKeyOff` command before timing out. If a time out occurs, the `handpay-action` value is used.
tiger:play-secondary-game-count	type: integer use: optional default: "0"	Number of double-or-nothing secondary games to attempt to play if the primary-win is not zero. For each secondary game play, the SmartEGM wagers all the winnings to that point, and continues until either the amount won equals zero or the number of games to play.
tiger:primary-win	type: integer use: optional	Number of credits to be awarded for the handle pull.

Attribute	Restrictions	Description
tiger:progressive-hit	type: boolean use: optional default: "false"	Indicates whether game play generates a progressive hit.
tiger:remote-key-off-timeout	type: integer use: optional default: "60000"	**DEPRECATED 03 OCT 2012 - Version 24** **Replaced by `tiger:key-off-timeout`** Time, in milliseconds, to wait for a remote keyoff.
tiger:win-final-secondary-game	type: boolean use: optional default: "false"	Indicates whether the final secondary game is a win or a loss. If `tiger:win-final-secondary-game` is set to **false**, the final win will equal zero.
tiger:win-level-index	type: integer use: optional default: "-1"	Defines the win-level index for the progressive hit.
tiger:win-to-handpay	type: boolean use: optional default: "false"	Indicates whether the game win should go to handpay.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Examples

This snippet plays a game on game play device 7, where two $0.25 cashable credits are wagered and one $0.25 cashable credit won.

```
    <tiger:Human.playSimpleGame tiger:device-id="7" tiger:denom-id="25000" tiger:primary-
    win="1" tiger:credits-to
         wager-cashable="2"/>
```

This snippet plays a game on game play device 7, where two $0.25 cashable credits are wagered and one $0.25 cashable credit is won. The game play causes a handpay event that is paid to a voucher.

```
    <tiger:Human.playSimpleGame tiger:device-id="7" tiger:denom-id="25000"
         tiger:primary-win="1" tiger:credits-to-wager-cashable="2"
         tiger:win-to-handpay="true" tiger:handpay-action="VOUCHER"/>
```

This snippet plays a game on game play device 7, where two $0.25 cashable credits are wagered and one $0.25 cashable credit won. The game play causes a handpay event. The game play waits until a remote key off event is received from the host. If the remote key off is not received in 10 minutes, the `Human.playSimpleGame` verb fails.

```
<tiger:Human.playSimpleGame tiger:device-id="7" tiger:denom-id="25000"
     tiger:primary-win="1" tiger:credits-to-wager-cashable="2"
     tiger:win-to-handpay="true" tiger:handpay-action="REMOTE"
     tiger:remote-key-off-timeout="600000"/>
```

tiger:Human.removeID

The `tiger:Human.removeID` verb simulates the removal of an ID from an ID reader device.

See also: `Human.insertID`

Attribute

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first ID Reader device. • A value of "-1" is legal, but should not be used because it will result in a failure.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet removes the current ID, using the first ID reader device.

```
    <tiger:Human.removeID tiger:device-id="-2"/>
```

tiger:Human.removeIDToDatabase

The `tiger:Human.removeIDToDatabase` verb simulates the removal of an ID from an ID Reader. The ID is returned to the ID database from which it came. The ID number that is removed is unlocked if it was locked when it was inserted.

IMPORTANT NOTE!
If you use your own ID database file, you must place the file in the **/user-data/id-<name>.xml** location.

For RST, the user-data directory **will** be deleted when the tool is uninstalled.
For RLT, the user-data directory **will not** be deleted when the tool is uninstalled.

See also `tiger:Human.insertIDFromDatabase`.

Attribute

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first ID reader device. • A value of "-1" is legal, but should not be used because it will result in a failure.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

Using the first ID Reader device remove the current ID and release it back to the external ID database from which it came.

```
<tiger:Human.removeIDToDatabase tiger:device-id="-2" />
```

tiger:Human.smartCardTransfer

The `tiger:Human.smartCardTransfer` verb simulates a player transfering funds to or from a smart card. Note that if no transfer amount is specified, all funds are transfered.

Attribute

Attribute	Restrictions	Description
tiger:amount-to-transfer-cashable	type: integer use: optional default: "0"	Amount of cashable funds to transfer (in millicents).
tiger:amount-to-transfer-non-cashable	type: integer use: optional default: "0"	Amount of non-cashable funds to transfer (in millicents).
tiger:amount-to-transfer-promo	type: integer use: optional default: "0"	Amount of promo funds to transfer (in millicents).
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first ID reader device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:direction	type: tiger:SmartCardDirection use: optional default: "FROM_EGM"	Direction of smart card transfer.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet transfers all the money from the player's credit meter to the smart card.

```
<tiger:Human.smartCardTransfer tiger:direction="FROM_EGM"/>
```

tiger:Human.watToEGM

The `tiger:Human.watToEGM` verb simulates a WAT transfer from the host to the EGM.

See also: `tiger:Human.watToHost`

Attributes

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first WAT device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:id-reader-device-id	type: integer use: optional default: "-2"	Identifier of the ID reader device. • A value of "-2" means the first ID reader device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:account-id	type: string use: optional	Player's WAT Account ID from which to transfer the funds.
tiger:amount-to-transfer cashable	type: integer use: optional default: "0"	Amount of cashable funds to transfer (in millicents).
tiger:amount-to-transfer-non-cashable	type: integer use: optional default: "0"	Amount of non-cashable funds to transfer (in millicents).
tiger:amount-to-transfer-promo	type: integer use: optional default: "0"	Amount of promotional funds to transfer (in millicents).

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then , tiger:tiger and tiger:try.

Example

This snippet transfers $10.00 from WAT account **12345678** to the cashable credit meter on the EGM, using the first WAT device.

```
<tiger:Human.watToEGM tiger:account-id="12345678" tiger:amount-to-transfer-
cashable="1000000"/>
```

tiger:Human.watToHost

The `tiger:Human.watToHost` verb simulates a WAT transfer from the EGM to the host.

See also: `tiger:Human.watToEGM`

Attribute	Restrictions	Description
tiger-id	type: integer use: optional default: "-2"	Device Identifier. A value of "-2" means the first WAT device. A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:id-reader-device-id	type: integer use: optional default: "-2"	Identifier of the ID reader device. • A value of "-2" means the first ID reader device. • A value of "-1" is legal, but should not be used because it will result in a failure.
tiger:account-id	type: string use: optional	Account ID of the player.
tiger:amount-to-transfer cashable	type: integer use: optional default: "0"	Amount of cashable funds to transfer (in millicents).
tiger:amount-to-transfer-non-cashable	type: integer use: optional default: "0"	Amount of non-cashable funds to transfer (in millicents).
tiger:amount-to-transfer-promo	type: integer use: optional default: "0"	Amount of promotional funds to transfer (in millicents).

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet transfers $10.00 cashable from the EGM to WAT account number **12345678**, using the first WAT device.

```
<tiger:Human.watToHost tiger:account-id="12345678" tiger:amount-to-transfer-
cashable="1000000"/>
```

tiger:if-cashable-credit-meter

The `tiger:if-cashable-credit-meter` verb allows you to add a condition based on the current value of the cashable credit meter. If the condition is true, the verbs in the `then` block are executed. If the condition is *not* true, the `else` block are executed.

This verb may contain tiger:then and tiger:else statements.

Attributes

Attribute	Restrictions	Description
tiger:conditional	type: string use: required	Cashable credit meter value.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

If the cashable credit meter is less than $0.50 then break out of the enclosing loop.

```
<tiger:if-cashable-credit-meter tiger:conditional="<50000">
     <tiger:then>
       <core:break />
     </tiger:then>
</tiger:if-voucher-available>
```

tiger:if-smart-card-negotiable

The `tiger:if-smart-card-negotiable` verb allows you to have condition logic based on if the smart card is negotiable. If the smart card is negotiable, the verbs in the `then` block are executed. If the smart card is not negotiable, the verbs in the `else` block are executed.

Attribute

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first ID reader device. • A value of "-1" is legal, but should not be used because it will result in a failure.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet enters the smart card PIN if the smart card is not negotiable.

```
<tiger:if-smart-card-negotiable>
     <tiger:then>
     </tiger:else> <tiger:Human.enterPIN/>
</tiger:if-smart-card-negotiable>
```

tiger:if-vendor

The `tiger:if-vendor` verb allows you to have conditional logic based on the vendor of the current EGM. If the vendor of the current EGM matches the vendor specified in the`tiger:if-vendor`statement, the verbs in the `tiger:then` block are executed. If the vendor of the current EGM does not match the given vendor, the verbs in the `tiger:else` block are executed.

Note: All EGMs match the vendor value `GSA` regardless of manufacturer.

This verb may contain tiger:then and tiger:else statements.

Attribute

Attribute	Restrictions	Description
tiger:vendor	type: string use: required minLength: 3 maxLength: 3	3-character manufacturer ID used in the egmId.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Examples

In this snippet, if the EGM is from `XYZ` company, add an entry to the log.

```
<tiger:if-vendor tiger:vendor="XYZ">
     <tiger:then>
       <core:log>This is an XYZ EGM.</core:log>
     </tiger:then>
</tiger:if-vendor>
```

In this snippet, if the EGM is **not** from `ABC` company, the script exits.

```
<tiger:if-vendor tiger:vendor="ABC">
     <tiger:then>
     </tiger:then>
     <tiger:else>
       <core:exit />
     </tiger:else>
</tiger:if-vendor>
```

tiger:if-voucher-available

The `tiger:if-voucher-available` verb allows you to have conditional logic based on the availability of a voucher in the database. If there is a voucher available, the verbs in the `tiger:then` block are executed. If there are no vouchers available in the database, the `tiger:else` block are executed.

IMPORTANT NOTE!
If you use your own voucher database file, you must place the file in the **/user-data/voucher-<name>.xml** location.

For RST, the user-data directory **will** be deleted when the tool is uninstalled.
For RLT, the user-data directory **will not** be deleted when the tool is uninstalled.

This verb may contain tiger:then and tiger:else statements.

Attribute

Attribute	Restrictions	Description
tiger:database-name	type: string use: required	Specifies the database name that will be used to retrieve Vouchers from. The database maps to a `../conf/voucher-${database-name}.xml` in the local file system.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Examples

This snippet inserts a voucher from the specified "initial-load" database if there is a voucher available.

```
<tiger:if-voucher-available tiger:database-name="initial-load">
     <tiger:then>
       <tiger:insertVoucherFromDabase database-name="initial-load" />
     </tiger:then>
</tiger:if-voucher-available>
```

This snippet inserts a voucher from the specified "initial-load" database if there is a voucher available. If there are no vouchers available in the initial-load database, the script exits.

```
<tiger:if-voucher-available tiger:database-name="initial-load">
     <tiger:then>
       <tiger:insertVoucherFromDabase database-name="initial-load" />
     </tiger:then>
     <tiger:else>
       <core:exit />
     </tiger:else>
</tiger:if-voucher-available>
```

tiger:message`

Attributes

Attribute	Restrictions	Description
There are no attributes for this verb.		

Content

Content	Restrictions	Description
text	type: string	

This verb may be included in: tiger:Transport.SendRawMessage

Example

tiger:Progressive.getHostInfo

The `tiger:Progressive.getHostInfo` verb sends a `progressive.getHostInfo` command to the owner of the specified progressive device.

Attribute

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default:"-2"	Device identifier. • A value of "-2" means the first Progressive device. • A value of "-1" is legal, but should not be used because it will result in a failure.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then , tiger:tiger and tiger:try.

Example

This snippet sends the `progressive.getHostInfo` command to the host, using the first progressive device.

```
<tiger:Progressive.getHostInfo/>
```

tiger:random

The `tiger:random` verb executes verbs randomly. Here is how it works:

1. All of the weight attributes from the `<tiger:action/>` child elements are put into a sequence, which generates a range of [0,*n*>, where *n* is the total of all of the weights.
2. When the `<tiger:random/>` element is executed, a random number is generated in the range [0,*n*>.
3. The `<tiger:action/>` element that corresponds in the range with the random value is selected.
4. All of the child elements in the selected `<tiger:action/>` element are executed, one at a time.

Attribute

Attribute	Restrictions	Description
There are no attributes for this verb.		

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then , tiger:tiger and tiger:try.

Element

Enumeration	Description
tiger:action	Verbs associated with a given random outcome.

Example

This snippet randomly selects one of the action blocks to execute.

```
<tiger:random>
     <tiger:action tiger:weight="50">
       <tiger:Human.insertNote tiger:device-id="1" tiger:currency-id="USD" tiger:denom-
       id="100000" tiger:note-action="DROP"/>
     </tiger:action>
     <tiger:action tiger:weight="5">
       <tiger:pause/>
     </tiger:action>
     <tiger:action>
       <tiger:pause/>
     </tiger:action>
     <tiger:action>
       <tiger:Human.playSimpleGame tiger:device-id="7" tiger:denom-id="25000"
       tiger:primary-win="1" tiger:credits-to-wager-cashable="2"/>
     </tiger:action>
</tiger:random>
```

tiger:tiger

The `tiger:tiger` verb is the root of the Tiger script.

This verb may contain the following: core:exit, core:fail, core:include, core:log, core:metadata, core:parameters, core:pause, tiger:assert, tiger:break, tiger:continue, tiger:DataModel.snapshot, tiger:DataModel.waitForDeviceState, tiger:DataModel.waitForEventQueueToDrain, tiger:DataModel.waitForHandpayKeyOff, tiger:Egm.resetEgm, tiger:Events.cabinetEvents, tiger:Events.coinAcceptorEvent, tiger:Events.noteAcceptorEvents, tiger:Events.printerEvents, tiger:Human.cancelCancelCreditHandpay, tiger:Human.cashOut, tiger:Human.changeDoorState, tiger:Human.createVoucher, tiger:Human.createVoucherToDatabase, tiger:Human.dispenseCoins, tiger:Human.dispenseNotes, tiger:Human.doCoinDrop, tiger:Human.doNoteDrop, tiger:Human.getWATAccounts, tiger:Human.getWATBalance, tiger:Human.insertCoins, tiger:Human.insertID, tiger:Human.insertIDFromDatabase, tiger:Human.insertNote, tiger:Human.insertVoucher, tiger:Human.insertVoucherFromDatabase, tiger:Human.keyOff, tiger:Human.playPaytableGame, tiger:Human.playSimpleCentralGame, tiger:Human.playSimpleGame, tiger:Human.removeID, tiger:Human.removeIDToDatabase, tiger:Human.watToEGM, tiger:Human.watToHost, tiger:if-vendor, tiger:if-voucher-available, tiger:Progressive.getHostInfo, tiger:random, tiger:repeat, tiger:Transport.sendMyCommand, tiger:Transport.sendRawMessage, tiger:try-catch and tiger:WAT.getKeyPair.

Examples

This snippet pauses the script for five seconds.

```
<tiger:tiger>
     <core:pause tiger:duration="5000"/>
</tiger:tiger>
```

tiger:Transport.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.

Note: You must have an RST Tester Toolkit license

HTTP Type: POST

Call Type: Synchronous

Attributes

Attribute	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: agamid 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.

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>
```

tiger:Transport.sendMyCommand

The `Transport.sendMyCommand` verb sends any valid G2S command to the owner of the specified device. The advantage of this verb over `Transport.sendRawMessage` is that the tool will wrap the message in a valid G2S message wrapper, supplying all needed (and correct) message and class information.

Attributes

Attribute	Restrictions	Description
tiger:device-class	type: device-class	
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first device of the specified class. • A value of "-1" is legal, but should not be used because it will result in a failure.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet sends the specified command to the owner of the first cabinet device.

```
<tiger:Transport.sendMyCommand tiger:device-class="G2S_cabinet" tiger:device-id="-2">
    <g2s:setCabinetState xmlns:g2s="http://www.gamingstandards.com/g2s/schemas/v1.0.3"
    g2s:enable="false" g2s:disableText="The Reason" />
</tiger:Transport.sendMyCommand>
```

tiger:Transport.sendRawMessage

The `Transport.sendRawMessage` verb sends a raw message to a given host.

Attribute

Attribute	Restrictions	Description
tiger:host-id	type: integer use: optional default: "1"	Host ID to which you will send this command.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Element

Element	Restrictions	Description
tiger:message	type: string minLength: 1	Message text.

Example

This snippet causes the script to send a raw message to Host 1. The script will not wait for a response.

```
<tiger:Transport.sendRawMessage tiger:host-id="1">
     <tiger:message>
       <![CDATA[
       "Anything can go here - a G2S message, or the Gettysburg address."
       ]]>
     </tiger:message>
</tiger:Transport.sendRawMessage>
```

tiger:try-catch

The `tiger:try-catch` verb is a means for catching and handling verb errors. The verbs in the try block are executed. If any of those verbs reports an error (of any kind) the verbs in the `tiger:catch` block are executed. This allows you to keep your script running if you think it is acceptable to handle the error on your own.

Note:

If no error occurs in the `try` block, the `tiger:catch` block is never executed.

If an error occurs in the `try` block, the `tiger:catch` block is executed immediately. No additional commands in the `try` block are executed.

If an error occurs in the `tiger:catch` block, the normal error handling rules apply. It is legal to put a new `tiger:try-catch` verb in the `tiger:catch` block. This element may contain the following elements: `tiger:catch`, `tiger:try`

Elements

Elements	Description
tiger:try	The `try` block.
tiger:catch	The `catch` block.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet attempts to insert a note. If there is a failure, it is noted and the script continues.

```
<tiger:try-catch>
     <tiger:try>
       <tiger:Human.insertNote tiger:denom-id="100000" />
     </tiger:try>
     <tiger:catch>
       <core:log core:level="warn">
          The note insertion failed. The script will continue.
       </core:log>
     </tiger:catch>
</tiger:try-catch>
```

tiger:catch

The `tiger:catch` element contains the verbs to execute when there is a failure in the `tiger:try` element of a `tiger:try-catch` verb.

See also `tiger:try-catch`

This element may contain the following elements: core:exit, core:fail, core:include, core:log, core:metadata, core:parameters, core:pause, tiger:assert, tiger:DataModel.waitForDeviceState, tiger:DataModel.waitForEventQueueToDrain, tiger:DataModel.waitForHandpayKeyOff, tiger:Events.coinAcceptorEvents, tiger:Events.noteAcceptorEvents, tiger:Events.printerEvents, tiger:Human.cancelCancelCreditHandpay, tiger:Human.cashOut, tiger:Human.changeDoorState, tiger:Human.createVoucher, tiger:Human.dispenseCoins, tiger:Human.dispenseNotes, tiger:Human.doCoinDrop, tiger:Human.doNoteDrop, tiger:Human.getWATAccounts, tiger:Human.getWATBalance, tiger:Human.insertCoins, tiger:Human.insertID, tiger:Human.insertNote, tiger:Human.insertVoucher, tiger:Human.keyOff, tiger:Human.playSimpleGame, tiger:Human.removeID, tiger:Human.watToEGM, tiger:Human.watToHost, tiger:if-vendor, tiger:Progressive.getHostInfo, tiger:random, tiger:repeat, tiger:try-catch and tiger:WAT.getKeyPair.

This element may be included in the following elements: `tiger:try-catch`.

Example

This snippet attempts to insert a note. If there is a failure, it is noted and the script continues.

```
<tiger:try-catch>
     <tiger:try>
            <tiger:Human.insertNote tiger:denom-id="100000" />
       </tiger:try>
       <tiger:catch>
            <core:log core:level="warn">
                  The note insertion failed. The script will continue.
            </core:log>
     </tiger:catch>
</tiger:try-catch>
```

tiger:try

The `tiger:try` element contains the verbs to attempt to execute as part of a `tiger:try-catch` verb.

See also `tiger:try-catch`

This element may contain the following elements: core:exit, core:fail, core:include, core:log, core:metadata, core:parameters, core:pause, tiger:assert, tiger:DataModel.waitForDeviceState, tiger:DataModel.waitForEventQueueToDrain, tiger:DataModel.waitForHandpayKeyOff, tiger:Events.coinAcceptorEvents, tiger:Events.noteAcceptorEvents, tiger:Events.printerEvents, tiger:Human.cancelCancelCreditHandpay, tiger:Human.cashOut, tiger:Human.changeDoorState, tiger:Human.createVoucher, tiger:Human.dispenseCoins, tiger:Human.dispenseNotes, tiger:Human.doCoinDrop, tiger:Human.doNoteDrop, tiger:Human.getWATAccounts, tiger:Human.getWATBalance, tiger:Human.insertCoins, tiger:Human.insertID, tiger:Human.insertNote, tiger:Human.insertVoucher, tiger:Human.keyOff, tiger:Human.playSimpleGame, tiger:Human.removeID, tiger:Human.watToEGM, tiger:Human.watToHost, tiger:if-vendor, tiger:Progressive.getHostInfo, tiger:random, tiger:repeat, tiger:try-catch and tiger:WAT.getKeyPair.

This element may be included in the following elements: tiger:try-catch.

Example

This snippet attempts to insert a note. If there is a failure, it is noted and the script continues.

```
<tiger:try-catch>
     <tiger:try>
       <tiger:Human.insertNote tiger:denom-id="100000" />
     </tiger:try>
     <tiger:catch>
       <core:log core:level="warn">
            The note insertion failed. The script will continue.
       </core:log>
     </tiger:catch>
</tiger:try-catch>
```

tiger:WAT.getKeyPair

The `WAT.GetKeyPair` verb sends a `WAT.getKeyPair` command. The *keyPairId* is set to `WAT.Status.keyPairId` plus one (1). The *hashType* is set to `WAT.Profile.hashType`.

Attribute

Attribute	Restrictions	Description
tiger:device-id	type: integer use: optional default: "-2"	Device identifier. • A value of "-2" means the first WAT device. • A value of "-1" is legal, but should not be used because it will result in a failure.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then, tiger:tiger and tiger:try.

Example

This snippet causes the script to send a `getKeyPair` command to the host.

```
<tiger:WAT.getKeyPair/>
```

Chapter 5: Adding Tester Toolkit Verbs

About Tester Toolkit Verbs

The Tester Toolkit is an *optional* module for RST. This module allows you to extend normal testing by customizing messages sent by the RST. You can:

- create a list of modifications that RST makes to specified outgoing messages, using message modification.
- clear all message modifications.
- block or change the message response to the host, using message disruption definitions.
- clear all message disruption definitions.

tiger: MessageDisruptionDefinition.Add

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

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

Attributes

Attributes	Restrictions	Description
action	type: tiger:MessageDisruptionAction use: optional default: "SEND_MSX003"ju8	Action to take for the definition.
attributeName	type: string use: optional default: " "	Name of the attribute for adding and searching.
attributeNamespaceURI	type: string use: optional default: "<ANY-NAMESPACE>"	Namespace of the attribute, for both adding and searching. When you use `<ANY-NAMESPACE>`, all attributes are searched regardless of namespace.
attributeValue	type: string use: optional default: " "	If the `action` element is SET, this is the new value of the attribute, whether it is added or replaced.
count	type: integer use: optional default: "1"	Controls how many times this definition is applied. When the count reaches zero, this modification record is deleted.
elementName	type: string use: optional default: "setKeepAlive"	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 use: required	Modification identifier. This value does not *have to* be unique, but it *should be* to more easily manage active modification records.

Example

This snippet adds a Message Disruption Definition for the current EGM. For the next `setKeepAlive` command received, the SmartEGM will send back an `G2S_MSX003 (Communications Not Online)` error in the `G2SACK` command.

```
<tiger:MessageDisruptionDefinition.add tiger:id="ID-1" tiger:action="SEND_MSX003"
tiger:count="1" tiger:elementName="setKeepAlive" tiger:attributeNamespaceURI="<ANY-
NAMESPACE>" />
```

tiger:MessageDisruptionDefinition.clearAll

The `tiger:MessageDisruptionDefinition.clearAll` verb lets you clear all previously added message disruption definitions. All message disruption definitions are cleared once the method returns.

Attributes

There are no attributes for this verb.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then , tiger:tiger and tiger:try.

Example

This request clears all of the current message disruption definitions for this EGM.

```
<tiger:MessageDisruptionDefinition.clearAll />
```

tiger:MessageModifications.add

The `modificationsAddList` verb 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.

Attributes

Attributes	Restrictions	Description
action	type: tiger:ModificationAction use: optional default: "REMOVE"	Action to take for the definition.
attributeName	type: string use: optional default: "hostId"	Name of the attribute for adding and searching.
attributeNamespaceURI	type: string use: optional default: "<ANY-NAMESPACE>"	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 use: optional default: " "	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: integer use: optional default: "1"	Controls how many times this definition is applied. When the count reaches zero, this modification record is deleted.
elementName	type: string use: optional default: "setKeepAlive"	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 use: required	Modification identifier. This value does not *have to* be unique, but it *should be* to more easily manage active modification records.

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>
```

tiger:MessageModifications.clearAll

The `tiger:MessageModifications.clearAll` verb lets you clear all previously added message modification definition. Modifications are cleared once the method returns.

Attributes

There are no attributes for this verb.

This verb may be included in: tiger:action, tiger:catch, tiger:else, tiger:repeat, tiger:then , tiger:tiger and tiger:try.

Example

This request clears all of the current message modifications for this EGM.

```
<tiger:MessageModifications.clearAll />
```

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

Attributes

There are no attributes for this verb.

core:gsa-protocol-name

Enumeration	Description
G2S	Game To System protocol.
S2S	System To System protocol.

core:LogLevelType

Enumeration	Description
ERROR	Messages related to program errors.
INFO	Messages that do not impact the system, but may be useful to know. INFO messages appear in black type.
WARN	Messages that indicate potentially harmful situations.

tiger:cashoutMethod

Enumeration	Description
HANDPAY	Cashout to a handpay.

tiger:coin-action

Enumeration	Description
DROP	Coin goes to drop.
HOPPER	Coin goes to hopper.

tiger:credit-type

Enumeration	Description
CASHABLE	Cashable voucher.
NON_CASHABLE	Non-cashable, promotional voucher.
PROMO	Cashable, promotional voucher.

tiger:device-class

Enumeration	Description
G2S_communications	Device for `communications` class.
G2S_cabinet	Device for `cabinet` class.
G2S_eventHandler	Device for `eventHandler` class.
G2S_meters	Device for `meters` class.
G2S_gamePlay	Device for `gamePlay` class.
G2S_deviceConfig	Device for `deviceConfig` class.
G2S_commConfig	Device for `commConfig` class.
G2S_optionConfig	Device for `optionConfig` class.
G2S_download	Device for `download` class.
G2S_handpay	Device for `handpay` class.
G2S_coinAcceptor	Device for `coinAcceptor` class.
G2S_noteAcceptor	Device for `noteAcceptor` class.
G2S_hopper	Device for `hopper` class.
G2S_noteDispenser	Device for `noteDispenser` class.
G2S_printer	Device for `printer` class.
G2S_progressive	Device for `progressive` class.
G2S_idReader	Device for `idReader` class.
G2S_bonus	Device for `bonus` class.
G2S_player	Device for `player` class.
G2S_voucher	Device for `voucher` class.
G2S_wat	Device for `wat` class.
G2S_gat	Device for `gat` class.
G2S_central	Device for `central` class.

tiger:doorAction

Enumeration	Description
OPEN	Door open.
CLOSE	Door closed.

tiger:door-name

Enumeration	Description
CABINET	EGM cabinet door.
LOGICAL	Logical EGM door.
AUXILIARY	Auxiliary fill compartment door.

tiger:keyOffAction

Enumeration	Description
HANDPAY	Key off the game to a local handpay.
CREDIT	key off the game to the credit meter.
VOUCHER	Key off the game to a voucher.
WAT	Key off the game to a WAT account
REMOTE	Key off the game to a remote server. The verb waits for the remote-key-off-timeout period. If the remote key off is not received in that period, the verb fails.

tiger:MessageDisruptionAction

Enumeration	Description
NORMAL_BEHAVIOR	No change in message response.
SEND_MSX001	Send MSX001 in response.
SEND_MSX002	Send MSX002 in response.
SEND_MSX003	Send MSX003 in response.
SEND_MSX004	Send MSX004 in response.
SEND_MSX005	Send MSX005 in response.
SEND_MSX006	Send MSX006 in response.
SEND_MSX007	Send MSX007 in response.
SEND_MSX008	Send MSX008 in response.
SEND_MSX099	Send MSX099 in response.
SEND_G2SACK	Send g2sAck in response.
SUPPRESS_G2SACK	Suppress g2sAck message.

tiger:note-action

Enumeration	Description
DISPENSER	Note goes to the dispenser.
DROP	Note goes to the stacker.
REJECT	Note is rejected.
RETURN	Note is returned to the player.

tiger:SetCabinetEvent

Enumeration	Description
AUXILIARY_DOOR_CLOSED	Auxiliary cabinet door closed.
AUXILIARY_DOOR_OPENED	Auxiliary cabinet door opened.
BACKUP_BATTERY_LOW	Backup battery is low.
CABINET_DOOR_CLOSED	Cabinet door closed.
CABINET_DOOR_OPENED	Cabinet door opened.
CABINET_FAULTS_CLEARED	Cabinet door faults cleared.
CASH_OUT	Cash out.
EGM_POWER_LOST	EGM power lost.

Enumeration	Description
EGM_POWER_UP	EGM power 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	Exit 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	Cabinet logic door is closed.
LOGIC_DOOR_OPENED	Cabinet logic door is opened.
NVM_FAILURE	Non-volatile memory failure.
OPERATOR_RESET_CABINET	Operator reset cabinet.
OPERATOR_CHANGED_CABINET_CONFIG	Operator has changed the cabinet configuration.
POWER_OFF_AUXILIARY_DOOR_OPEN	Power off auxiliary door open.
POWER_OFF_CABINET_DOOR_OPEN	Power off auxiliary cabinet door open.
POWER_OFF_LOGIC_DOOR_OPEN	Power off logic door open.
SERVICE_LAMP_OFF	Service lamp off.
SERVICE_LAMP_ON	Service lamp on.
VIDEO_DISPLAY_ERROR	Video display error.

tiger:SetCoinAcceptorEvent

Enumeration	Description
ACCEPTOR_FAULT	Acceptor fault.
ACCEPTOR_JAMMED	Acceptor jammed.
CLEAR_ALL_FAULTS	Clear all faults.
COIN_ACCEPTOR_CONNECTED	Coin acceptor connected.
COIN_DROP_DOOR_OPEN	Coin drop door open.
COMPONENT_FAULT	Component fault.
DIVERTER_FAULT	Diverter fault.
FIRMWARE_FAULT	Firmware fault.
ILLEGAL_ACTIVITY_DETECTED	Illegal activity detected.
LOCKOUT_MALFUNCTION	Lookout malfunction.
MECHANICAL_FAULT	Mechanical fault.
NON_VOLATILE_MEMORY_FAULT	Non-volatile memory fault.
OPTICAL_FAULT	Optical fault.
COIN_ACCEPTOR_DISCONNECTED	Coin acceptor disconnected.
COIN_DROP_DOOR_CLOSED	Coin drop door closed.
OPERATOR_CHANGED_COIN_ACCEPTOR_CONFIG	Operator changed coin acceptor configuration.

tiger:SetNoteAcceptorEvent

Enumeration	Description
ACCEPTOR_FAULT	Acceptor fault.
ACCEPTOR_JAMMED	Acceptor jammed.
CLEAR_ALL_FAULTS	Clear all faults.
COMPONENT_FAULT	Component fault.
FIRMWARE_FAULT	Firmware fault.
ILLEGAL_ACTIVITY_DETECTED	Illegal activity detected.
MECHANICAL_FAULT	Mechanical fault.
NON_VOLATILE_MEMORY_FAULT	Non-volatile memory fault.
NOTE_ACCEPTOR_CONNECTED	Note acceptor connected.
OPTICAL_FAULT	Optical fault.
STACKER_DOOR_OPEN	Stacker door open.
STACKER_FAULT	Stacker fault.

Enumeration	Description
STACKER_FULL	Stacker full.
STACKER_JAMMED	Stacker jammed.
STACKER_NEARLY_FULL	Stacker nearly full.
STACKER_REMOVED	Stacker removed.
VALIDATOR_TOTALS_RESET	Validator totals reset.
NOTE_ACCEPTOR_DISCONNECTED	Note acceptor disconnected.
OPERATOR_CHANGED_NOTE_ACCEPTOR_CONFIG	Operator changed note acceptor configuration.
STACKER_DOOR_CLOSED	Stacker door closed.
STACKER_INSERTED	Stacker inserted.

tiger:SetPrinterEvent

Enumeration	Description
CLEAR_ALL_FAULTS	Clear all faults.
COMPONENT_FAULT	Component faults.
FIRMWARE_FAULT	Firmware faults.
MECHANICAL_FAULT	Mechanical fault.
NON_VOLATILE_MEMORY_FAULT	Non-volatile memory fault.
OPTICAL_FAULT	Optical fault.
PAPER_EMPTY	Paper empty.
PAPER_JAM	Paper jam.
PAPER_LOW	Paper low.
PRINT_HEAD_OPEN	Print head open.
PRINTER_CHASSIS_OPEN	Printer chassis open.
PRINTER_CONNECTED	Printer connected.
OPERATOR_CHANGED_PRINTER_CONFIG	Operator changed printer configuration.
PRINT_HEAD_CLOSED	Print head closed.
PRINTER_CHASSIS_CLOSED	Printer chassis closed.
PRINTER_DISCONNECTED	Printer disconnected.

tiger:SmartCardDirection

Enumeration	Description
FROM_EGM	Transfer from the EGM to the smart card.
TO_EGM	Transfer to the EGM from the smart card.

tiger:voucher-action

Enumeration	Description
DROP	Voucher goes to the stacker.
REJECT	Voucher is rejected.
RETURN	Voucher is returned to the player.