GSA GAT PROTOCOL
V3.50.1
Game Authentication Terminal

Gaming Standards Association
GAT Technical Committee

Released: 2011/02/13
Document ID: gsa-p0093.002.00

GAMINGSTANDARDS.COM

GSA GAT Protocol v3.50.1, Document ID gsa-p0093.002.00

Released 2011/02/13, by Gaming Standards Association (GSA).

Patents and Intellectual Property

NOTE: The user's attention is called to the possibility that compliance with this [standard/specification] may require use of an invention covered by patent rights. By publication of this [standard/specification], GSA takes no position with respect to the validity of any such patent rights or their impact on this [standard/specification]. Similarly, GSA takes no position with respect to the terms or conditions under which such rights may be made available from the holder of any such rights. Contact GSA for further information.

Trademarks and Copyright

GSA Contact Information

E-mail: sec@gamingstandards.com

WWW: http://www.gamingstandards.com

Table of Contents

I About This Document

The GSA GAT Protocol is a communication standard used by regulators and operators to identify and authenticate gaming software and firmware in the field.

I.I Acknowledgements

The Gaming Standards Association expresses its appreciation to all members of the GAT committee (past and present) as well as gaming regulators and others, for their significant contribution and dedication to the creation of this standard.

I.II Related Documents

SVC Serial Protocol v1.0

http://www.gamingstandards.com/pdfs/standards/SVC_r1.pdf

Game Authentication Terminal Program (GAT3) Requirements Document

http://www.gamingstandards.com/pdfs/standards/GSA_GAT3_r1.pdf

EIA/TIA-232 (RS-232)

http://www.tiaonline.org/standards/

I.III Document Conventions

I.III.I Indicating Requirements, Recommendations, and Options

Terms and phrases in this document that indicate requirements, recommendations, and options are used as defined in the IETF RFC 2119.

In summary:

Requirements:
To indicate requirements, this document uses "MUST", "MUST NOT", "REQUIRED".

Recommendations:

To indicate recommendations, this document uses "SHOULD", "SHOULD NOT", "RECOMMENDED".

Options:

To indicate **options**, this document uses "MAY" or "OPTIONAL".

I.III.II Corrections and Clarifications

A pale **yellow** banner identifies content that has been corrected or clarified since the last released version, along with text that identifies in what version the changes were made. The following example shows how this convention is used, and indicates that corrections were made in v3.50.1 to content.

Note that correction banners and the associated inserted and deleted text is highlighted only in the mark-up PDF of released versions, and are provided only for changes made between the last released version and the current released version. Correction indicators are not carried forward from version to version.

Corrections in v3.50.1

Lorem ipsum dolor sit amet, consectetur ~~consectetur~~ adipiscing elit. Aliquam consectetur justo vel odio consequat rutrum. Morbi magna neque, blandit a dictum nec, vestibulum ac velit. Donec ultrices imperdiet mi, eget pharetra enim porttitor quis. Nam vestibulum massa eget augue consectetur condimentum tempus enim pellentesque.

I.III.III Other Formatting Conventions

- Blue text indicates an internal link or external hyperlink to a URL.
- **Bold** (other than in headings) or underlined text is used for emphasis, unless specifically indicated otherwise.
- *Italicized* text (other than in headings) is used for terms being introduced and/or being defined.
- `Courier New` font is used to indicate code or pseudo code.

Chapter 1

Introduction

1.1 Overview

GAT defines a communications protocol used, between a master and an EGM, to authenticate software and firmware components within the EGM. Typically, a portable PC or a laptop is used for the role of the master. EGMs and other devices can be used for the role of the EGM.

The GAT communication protocol is simple in order to reduce complexity of design, implementation, testing and usage. Due to the simplicity of this protocol, a standard layered approach is not necessary. Only the physical layer and the application layer command set are specified.

The GAT protocol and associated calculations are to be run on a properly functioning EGM. Any attempt to use GAT while an EGM is in an error state, tilted, or otherwise malfunctioning is beyond the scope of this standard.

The GAT protocol and associated calculations are designed for the purposes of verifying software content on an EGM. Any attempt to use GAT for any other purpose, such as verifying jackpots, game history recall, and so forth, is beyond the scope of this standard.

Chapter 2

Physical Layer

2.1 Physical Layer Between EGM and Master

The physical layer between the EGM and the master is:

- point-to-point
- full duplex
- no handshaking
- 3-wire (Tx/Rx/Gnd) RS232C

The default communication:

- 9600 baud with eight data bits
- no parity
- one stop bit

The master is typically a laptop PC and is generally assumed to provide a standard DE9 (commonly known as a DB9) male connector (DE9M) configured as a DTE interface, as shown in Table 2.1.

Table 2.1 Pinout for DE9M Connector Configured as DTE

Pin	Function
Pin 2	RX. Receives data.
Pin 3	TX. Transmits data.
Pin 5	GND. Signal ground.

The EGM MUST provide a connector suitable for connection to this typical master DE9M. There are three options by which this may be accomplished:

1. The EGM MAY provide a standard DE9 female connector (DE9F) configured as a DCE, as shown in Table 2.2. The master may connect to the EGM using a standard RS-232 "straight-through" cable.

Table 2.2 Pinout for DE9F Connector Configured as DCE

Pin	Function
Pin 2	TX. Transmits data.
Pin 3	RX. Receives data.
Pin 5	GND. Signal ground.

2. The EGM MAY provide a standard DE9 male connector (DE9M) configured as a DTE, as shown in Table 2.1. The master may connect to the EGM using a standard RS-232 "null modem" cable.
3. The EGM MAY provide a non-standard connector. If a non-standard connector is provided, the EGM manufacturer MUST clearly document the pinout for this connector, and MUST make available a cable or adapter that mates to the EGM's GAT connector on one end and has a standard DE9 female connector (DE9F) configured as a DCE, as shown in Table 2.2, on the other end. This cable MUST NOT exceed 10 feet in length.

The EGM GAT connector MUST be located within a secure area of the EGM. It is recommended that the GAT connector be located in an easily accessible location within the interior of the EGM cabinet and labeled for easy identification.

NOTE:
This standard does not specify whether a dedicated physical port is (or is not) required for the EGM GAT connector. This leaves the option open to the manufacturer as to whether port sharing is an acceptable solution within the particular jurisdiction where it will be used. It is up to the manufacturer to determine whether the jurisdiction will allow port sharing.

Chapter 3

Application Command Layer

3.1 Overview

Corrections in 3.50.1

At the application layer, the master sends a *query* to the EGM and waits for the *response* before sending another command. The EGM always responds to a query with a response. As a consequence no more than one query / response may be pending at the master / EGM side at any given time.

The EGM MUST validate the length and CRC, and then it MUST validate the command byte. The EGM SHOULD NOT respond to ~~packet~~messages with invalid length, CRC, or command bytes.

The Master MUST validate the length and CRC, and then it MUST validate the command byte. The master SHOULD ignore ~~packet~~messages with invalid length, CRC, or command bytes.

The following time-outs will be in effect:

1. The EGM MUST respond within 200ms of receiving a complete ~~packet~~message from the master.
2. If the master does not receive a response to a request, the master SHOULD wait at least 225ms before sending another request.
3. The recommended inter-byte timeout value is 5ms.
4. If the EGM has determined that the previously received byte was the last byte of a valid ~~packet~~message, or 200ms have elapsed since the previously received byte, the EGM SHOULD treat the next byte received as belonging to a new ~~packet~~message.
5. The master MUST wait at least 10ms upon receipt of a response before transmitting again.

3.2 Application Layer Format

3.2.1 Byte Order

The GAT protocol uses Big Endian (most significant byte first) byte ordering for all cases where multi-byte, numeric information is conveyed by the GAT protocol unless another format is specifically stated (typically through the use of the Data Format byte).

3.2.2 Bit Order

For bit-field parameters, bit **0** always refers to the least significant bit. Bit **7** always refers to the most significant bit. The following table may be used to determine bit positions:

Table 3.1 Bit Positions (Sheet 1 of 2)

Bit	Bit Mask	Description
0	0x01	Least significant bit.
1	0x02	2nd bit position.
2	0x04	3rd bit position.
3	0x08	4th bit position.

Table 3.1 Bit Positions (Sheet 2 of 2)

Bit	Bit Mask	Description
4	0x10	5th bit position.
5	0x20	6th bit position.
6	0x40	7th bit position.
7	0x80	Most significant bit.

3.2.3 Transmission Order

The bytes of a message are transmitted from left to right—that is, command byte first and CRC bytes last. The order of the bits within a byte follows the RS-232 specification of LSB (bit **0**) first and MSB last. All bits of a byte are transmitted before the next byte is started.

3.2.4 Data Formats

Corrections in v3.50.1

The following data formats are supported by the GAT protocol:

Binary: Each byte represents a binary value between `0x00` through `0xFF` inclusive.

Packed BCD: Each byte represents a decimal value between 00 and 99 inclusive, represented as ~~hexadecimal~~binary `0x00` through ~~`0x99 inclusive`~~.

HEX-ASCII: A hexadecimal string representation of a ~~hexadecimal~~binary value. ~~Hexadecimal~~Binary values are converted to uppercase ASCII hexadecimal strings that ~~form an uppercase ASCII~~ representation~~ of~~ the ~~hexadecimal~~binary values. ~~For example: the value 0x0123456789abcdef (or 0x0123456789ABCDEF) is represented as the string `0123456789ABCDEF`~~ An even number of nibbles (hexadecimal digits) MUST be included. Only ASCII characters 0-9 (0x30 through 0x39) and A-F (0x41 through 0x46) MUST be used. For example: the binary value 0x0123456789abcdef (or 0x0123456789ABCDEF) is represented as the string 0123456789ABCDEF and is transmitted as the bytes 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46. See Section 5.4, Example SHA-1 HMAC Authentication, which documents a transmission that includes a 20-byte binary key value.

ASCII: An ASCII data string. May include control characters such as CR (`0x0D`) and LF (`0x0A`).

XML: A well-formed XML document conforming to XML version 1.0.

XML version 1.0 requires that XML processors MUST support UTF-8 and UTF-16 encodings of an XML document. Thus, implementations of the GAT protocol MUST support UTF-8 and UTF-16 encodings for the XML data type. However, since UTF-8 tends to create smaller document sizes than UTF-16, implementations of this protocol SHOULD use UTF-8 encodings for XML documents. The GAT protocol does not provide a mechanism for selecting the encoding of an XML document. The default encoding is UTF-8.

3.2.5 Application Layer Frame

Corrections in 3.50.1

Table 3.2 Frame Structure

Command	Length	Message Data	CRC
1 byte binary	1 byte binary	0 to 251 bytes (varies)	2 bytes binary

This frame consists of the following fields:

Table 3.3 Frame Field Descriptions

Field	Description
Command	This is a command byte that indicates the ~~packet~~message format and its purpose. Transmitted first.
Length	The total number of bytes in frame (including Command, Length, Message Data, and CRC bytes). Note: The maximum ~~packet~~message length is restricted to 255 bytes.
Message Data	This field contains any data relevant to the command. The data format depends on the specific command.
CRC	A CRC-16 checksum of the Command, Length, and Message Data fields. Each frame is protected with a 16-bit Cyclic Redundant Check sequence. The CRC uses the industry standard CRC-16 polynomial generator of x^16 + x^15 + x^2 + 1 starting with a seed of 0xFFFF. See Appendix A for further details on correct implementation of this CRC. Transmitted last.

3.3 Commands - Query / Response Pairs

Each query has one corresponding response. The appropriate matched response should be returned by the EGM when a query is received and processed. The command byte for a response is the same as that of the query, except the high bit is set (i.e. `0x02-0x82`).

3.3.1 Status Query (0x01 SQ)

[Master ⇨ EGM] Request the current status information from the EGM.

Table 3.4 0x01 SQ Structure

Cmd = 0x01 SQ	Length = 0x04	CRC
1 byte binary	1 byte binary	2 bytes binary

3.3.2 Status Response (0x81 SR)

[EGM ⇨ Master] Return the current status information.

Table 3.5 0x81 SR Structure

Cmd = 0x81 SR	Length = 0x08	Version ID	Status Data1	Data Format	CRC
1 byte binary	1 byte binary	2 bytes packed BCD	1 byte binary	1 byte binary	2 bytes binary

Table 3.6 0x81 SR Fields

Field	Description
Version ID	Indicates the version of the GAT protocol supported by the EGM. The version is a 4-digit number, where the first byte is 2-digit major revision number and the second byte is 2-digit minor revision number. The errata revision number is not included. For example, `0x03 0x50` indicates GAT version 3.50.0, 3.50.1, 3.50.2, and so on; `0x03 0x51` indicates GAT version 3.51.0, 3.51.1, 3.51.2, and so on; and, `0x04 0x01` indicates GAT version 4.1.0, 4.1.1, 4.1.2, etc.
Status Data1	General Status: Bit **0**: Calculation Status. `0` = Idle. `1` = Calculating. Bit **1**: Last Authentication Results. `0` = Not Available. `1` = Available. Bit **2** & **3**: See Table 3.7 for Current Calculation. Bit **4** to **7**: Reserved. Always set to `0`.
Data Format	Data formats supported: `0x00` = Reserved, do not use. `0x01` = Plain text format. `0x02` – XML format. `0x03` to `0xFF` – Reserved for future use.

Table 3.7 0x81 Status Data1 Field: Bit 2 & 3, Current Calculation

Bit 3 Value	Bit 2 Value	Description
0	0	Requested.
1	0	Calculating.
0	1	Finished.
1	1	Error, cannot complete or failed.

3.3.3 Last Authentication Status Query (0x02 LASQ)

[Master ⇨ EGM] Request the status of the last authentication performed by the EGM. Only the status of the last completed authentication is returned.

Table 3.8 0x02 LASQ Structure

Cmd = 0x02 LASQ	Length = 0x04	CRC
1 byte binary	1 byte binary	2 bytes binary

3.3.4 Last Authentication Status Response (0x82 LASR)

[EGM ⇨ Master] Return the status of the last authentication result calculated by the EGM.

Table 3.9 0x82 LASR Structure

Cmd = 0x82 LASR	Length = 0x09	Authentication Level	Time	CRC
1 byte binary	1 byte binary	1 byte binary	4 bytes binary	2 bytes binary

Table 3.10 0x82 LASR Fields

Field	Description
Authentication Level	Indicates the level or type of authentication that was calculated. A value of `0x01` refers to Level 1 Authentication, `0x02` refers to Level 2 Authentication, and so on. A value of `0x00` indicates no authentication results are available. For this version of the GAT protocol, an EGM MUST support levels 0xBA and 0x00. Other levels MAY be defined in other versions of the GAT protocol and MAY be supported by the EGM.
Time	Time (in seconds) since last results were calculated. If no authentication results are available, then a value of `0x00000000` is returned.

3.3.5 Last Authentication Results Query (0x03 LARQ)

[Master ⇨ EGM] Request the previous/currently available Authentication results.

Table 3.11 0x03 LARQ Structure

Cmd = 0x03 LARQ	Length = 0x07	Data Format	Frame Number	CRC
1 byte binary	1 byte binary	1 byte binary	2 bytes binary	2 bytes binary

Table 3.12 0x03 LARQ Fields

Field	Description
Data Format	The format of the data: `0x00` = Reserved, do not use. `0x01` = Plain text format. `0x02` = XML format. `0x03` to `0xFF` = Reserved for future use.
Frame Number	This number, with the most significant byte first, is used to indicate the Data Frame that should be returned as data in the Last Authentication Results Response (0x83 LARR). The frame number data is indexed from `1`, so a value of `0` is illegal. The range is large enough to handle a file containing up to 65535 frames.

NOTE:
It is important to note that this mechanism of accessing the authentication results is linear, not random access. The rule exists in order to reduce any possible load or restrictions on the implementation within the EGM. The implications of this are that for each result, the first frame requested can only be frame 1. After that the master can only request either the *first* frame, frame *n*, or frame *n+1*, where *n* was the previous frame requested. This results in a linear request process, with the ability to reset back to the first frame, or request a retransmit of the current frame, or request that the next frame be transmitted.

3.3.6 Last Authentication Results Response (0x83 LARR)

[EGM ⇨ Master] Return a data frame of the previous or currently available Authentication results.

Table 3.13 0x83 LARR Structure

Cmd = 0x83 LARR	Length = 0x07 to 0xFF	Status Data	Frame Number	Data	CRC
1 byte binary	1 byte binary	1 byte binary	2 bytes binary	0 to 248 bytes (varies)	2 bytes binary

Table 3.14 0x83 LARR Fields

Field	Description
Status Data	General Status: Bit **0**:Error Status. `0` = No error. `1` = Error. (Note: Error would usually indicate either no data available, or an invalid frame.) Bit **1**: Frame Status. `0` = Not Last Frame. `1` = Last Frame.
Frame Number	Used to indicate the frame, with the most significant byte first, that is being returned in the Data field. MAY be set to frame 0 (`0x00 0x00`) when an error is being reported (Bit **0** of the Status Data set to `1`).
Data	Contains requested Authentication information (formatted as requested). This response is the mechanism used by the EGM to communicate the result of any special function. See Chapter 4 and Chapter 5 for further discussion of the format for authentication and special function responses.

NOTE:
Authentication Results are not available while an Authentication Calculation is in progress. If a 0x03 LARQ request is received while an Authentication Calculation is in progress, the EGM MUST return an error to the master in the 0x83 LARR response, setting Bit **0** and Bit **1** of the Status Data to `1`.

3.3.7 Initiate Authentication Calculation Query (0x04 IACQ)

[Master ⇨ EGM] Request that the EGM start authentication calculation.

Table 3.15 0x04 IACQ Structure

Cmd = 0x04 IACQ	Length = 0x05 to 0xFF	Authentication Level	Authentication Parameter	CRC
1 byte binary	1 byte binary	1 byte binary	0 to 250 bytes HEX-ASCII	2 bytes binary

Table 3.16 0x04 IACQ Fields

Field	Description
Authentication Level	Indicates the level or type of authentication calculation that should be returned. A value of `0x01` refers to Level 1 Authentication, `0x02` refers to Level 2 Authentication, and so on. A value of `0x00` is illegal. For this version of the GAT protocol, an EGM MUST support level 0xBA. The EGM MUST return error code 0x04 if level 0x00 is requested. Other levels MAY be defined in other versions of the GAT protocol and MAY be supported by the EGM. The special authentication level `0xBA` is used by the master to signal that the Authentication Parameter field contains a special function command. In this case, the Authentication Parameter field MUST have the first byte set to `0x00`. See Chapter 4 and Chapter 5 for further discussion of special functions.
Authentication Parameter	The Authentication Parameter value is used for some Authentication Levels. The same value is used for all modules verified by an Authentication Level. If the value is longer than required by an Authentication Level, it is truncated, the high order bytes discarded. The Authentication Parameter is represented in HEX-ASCII format. If the Authentication Level is set to the special value `0xBA`, the first byte of the Authentication Parameter field MUST be set to `0x00` while the remainder of the field contains the special function. See Chapter 4 for details. The data format is specified with each command.

NOTE:
If an Authentication Calculation is in progress when this command is received by the EGM, the EGM MUST abort the calculation and start the new Authentication Calculation. Issuing a new Authentication Calculation while the EGM is calculating is not recommended. The master can determine the state of the EGM using the 0x01 SQ command.

3.3.8 Initiate Authentication Calculation Response (0x84 IACR)

Corrections in v3.50.1

[EGM ⇨ Master] Indicate that the EGM has received a 0x04 IACQ command. The EGM SHOULD maintain the last 0x04 IACQ result for the master to retrieve for as long as that result is valid, even while the master is disconnected. Whenever a new 0x04 IACQ request is received by the EGM, the EGM MUST overwrite any previous results with the new authentication results. If an error occurred such that the IACQ request did not result in new authentication results, the 0x84 IACR response MUST report the error and the EGM MAY overwrite or otherwise discard the previous authentication results. In addition, the EGM SHOULD discard the last 0x04 IACQ result whenever the EGM is reset or the set of supported calculations changes—for example, due to a change to the set of components on the EGM. If the operator has placed the EGM in a special GAT authentication mode in order to calculate authentication results, the EGM MAY also discard the last result when the operator causes the EGM to exit its GAT authentication mode.

Table 3.17 0x84 IACR Structure

Cmd = 0x84 IACR	Length = 0x045	Status	CRC
1 byte binary	1 byte binary	1 byte binary	2 bytes binary

Table 3.18 0x84 IACR Fields

Field	Description
Status	Bit **0**: ACK/NACK. `0` = Cannot Acknowledge. `1` = Acknowledged. Bit **1**: Calculation Started. `0` = Not started. `1` = Started. Bit **2**: Level Compliance Error. `0` = Valid Level. `1` = Invalid Level requested.

Chapter 4

Special Functions

4.1 Overview

Corrections in v3.50.1

The master may request the EGM to execute a number of special functions. This is accomplished by setting the Authentication Level to `0xBA` and providing the appropriately formatted command in the Authentication Parameter field of an Initiate Authentication Calculation Query (0x04 IACQ). Results from the execution of a special function are sent to the master from the EGM in the Last Authentication Results Response (0x83 LARR).

When formatting special function commands, the following rules MUST be observed:

1. Individual data elements within the command MUST be separated by the tab character (`0x09`). A tab character MUST NOT precede the first data element. A tab character MUST NOT follow the last data element.
2. The name of the special function MUST be the first data element in the command. The name of the special function is contained in the `Feature` element of the response to the "Get Special Functions" command.
3. Unless specified otherwise in the description of the special function, parameters of the special function, if any, MUST follow the first data element in the same order as they are reported in the response to the "Get Special Functions" command.
4. When a parameter of the special function specifies a wildcard, the master may replace the wildcard with an appropriate corresponding value—for example, the matching value contained in the SEEDS.INI configuration file used by the GAT3.exe program.
5. Wildcards MUST be constructed from a leading `%%` sentinel (two percent signs), a wildcard name, and a trailing `%%` sentinel (two percent signs)—for example, `%%SHA1_HMAC%%`. The wildcard name MUST be constructed using one or more valid ASCII characters in the range `0x20` to `0x7E`, excluding `0x25` (the percent sign).
6. The master MUST provide an actual value for the wildcard. If there is no corresponding value for the wildcard, the wildcard MUST be replaced by "(none)" (0x28 0x6E 0x6F 0x6E 0x65 0x29). If the wildcard represents a seed, hash, offset, or HMAC key, the text string "(none)" MUST be interpreted to mean "no seed, hash, offset, or key provided" and MUST NOT be used as a seed, hash, offset, or key.
7. Special Functions that call for an offset parameter, a salt parameter, a key parameter, or an authentication hash parameter MUST provide those values in a HEX-ASCII data format (see Section 3.2.4, Data Formats, for more details). If the values are not in HEX-ASCII data format, the EGM SHOULD respond to a 0x04 IACQ command containing such values with a 0x84 IACR command containing status 0x00 and not execute the special function.

When the master issues the 0x04 IACQ command, the EGM responds with the Initiate Authentication Calculation Response (0x84 IACR) command. The EGM MUST use the Status field of the 0x84 IACR to indicate the state of the request. One of the following states in Table 4.1 MUST be reported by the EGM:

Table 4.1 0x84 IACR States

0x84 Response Status Field	State
0x00	Request not acknowledged—invalid Authentication Parameters detected. Special function will not be executed.
0x01	Request acknowledged and special function will be executed.
0x03	Request acknowledged and special function started.
0x04	Request not acknowledged—invalid Authentication Level detected. Special function will not be executed.

The master MUST be prepared to receive other states from the EGM. Any such states simply indicate that the request could not be acknowledged (Bit **0** set to 0 or Bit **2** set to 1). The master MUST interpret other states as if state 0x04 was reported (when Bit **2** is set to 1) or as if state 0x00 was reported (when Bit **2** is set to 0).

After the master issues an 0x04 IACQ containing a special function request, the master may use the Status Query (0x01 SQ) command to determine if the results of the special function are ready. The EGM should use the Status field of the Status Response (0x81 SR) to determine the state of the request. One of the following states MUST be reported by the EGM:

Table 4.2 0x81 SR States

0x81 Response Status Data1 Field	States	Description
0x00	Idle, Not Available, and Requested	The special function request has been received but has not yet been executed.
0x04	Idle, Not Available, and Finished	No special function results are available from the EGM. This is the initial state of the EGM before any special function requests have been executed.
0x06	Idle, Available, and Finished	The special function has been completed and the results are available.
0x09	Calculating, Not Available, and Calculating	The special function is executing.
0x0C	Idle, Not Available, and Error	The special function failed in some way. No further information is available.
0x0E	Idle, Available, and Error	The special function failed in some way. Information regarding the error is available.

The master MUST be prepared to receive other states from the EGM. Any such states are contradictory and/or ambiguous. The master MUST interpret other states as if state 0x0C (Idle, Not Available, and Error) was reported.

Once the EGM has indicated results are ready, the results may be obtained by the master through the use of the Last Authentication Results Query (0x03 LARQ). The EGM should then respond with the 0x83 LARR command and set the Data field to appropriate value.

The Data Format for the special function responses that are defined in this section is always XML. Thus, after a special function that is defined in this section has been successfully executed by the EGM, the Data Format of the 0x81 SR from the EGM MUST specify XML format (`0x02`). Likewise, when the master requests the results of a special function that is defined in this section, the Data Format of the 0x03 LARQ from the master MUST specify XML format (`0x02`). Other formats may be used for other types of functions and for reporting errors.

As described in Section 3.3.8, Initiate Authentication Calculation Response (0x84 IACR), the EGM SHOULD maintain the last 0x04 IACQ result for the master to retrieve for as long as that result is valid, even while the master is disconnected. Requesting a new special function MUST overwrite the previous results with the new authentication results. If an error occurred such that the IACQ request did not result in new authentication results, an error MUST be reported in the 0x84 IACR response and the EGM MAY overwrite or otherwise discard the previous authentication results. The EGM SHOULD discard the last 0x04 IACQ result whenever the EGM is reset or the set of supported special functions changes.

In the following sections, "<00>" is used to indicate an ASCII null character (byte value of `0x00`) and "<09>" is used to indicate an ASCII tab character (byte value of `0x09`).

4.2 Defined Special Functions

The GAT process is primarily intended to facilitate compliance with jurisdictional requirements. For example, Nevada requires an EGM to provide a method to authenticate all EGM control programs and data on demand via an approved communication port and protocol. It is up to each manufacturer to determine which components are included in these requirements. It is also up to each manufacturer to determine to what granularity components may be authenticated. It is strongly recommended that the master be able to authenticate components to the same level of granularity that they are submitted to the jurisdiction for approval.

4.2.1 Special Function: Get Special Functions

Corrections in v3.50.1

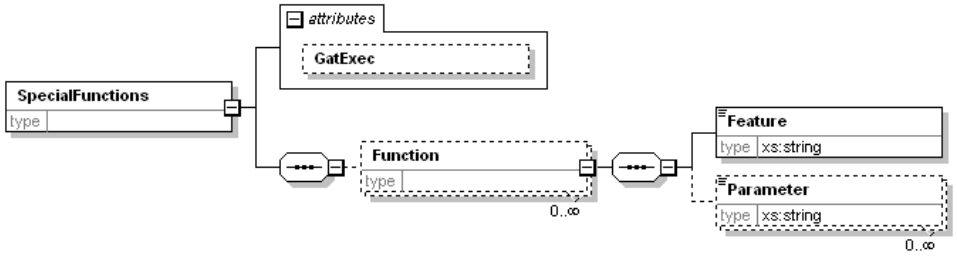
All EGMs MUST support the "Get Special Functions" special function. To discover which special functions an EGM supports, the master may send the following 0x04 IACQ:

Table 4.3 0x04 IACQ Structure for Get Special Functions

Cmd	Length	Authentication Level	Authentication Parameter (Data)	CRC
0x04	0x1B	0xBA	<00>Get Special Functions	0x2B54

Upon receipt of this special function, the EGM MUST acknowledge it with a correctly formatted 0x84 IACR. Once the EGM indicates it is finished by returning a Status of `0x06` in a 0x81 SR, the master may then retrieve the listing by sending a 0x03 LARQ command. The EGM should respond with a 0x83 LARR command containing the supported special functions.

The response MUST be XML formatted and conform to the following definition (See Appendix B for more details):



The `GatExec` attribute MUST be set to `default` for ~~GAT v3.50.0~~ compatibility with this version of the GAT protocol. The original GAT3 protocol intended that this attribute could be set to the path of an executable program on the master; and, the master would save the response in a file by the filename specified in the first parameter, and then execute the program specified by `GatExec`. This capability is NOT supported by this version of the GAT protocol~~GAT v3.50.0~~.

The EGM MUST return a list of all special functions that it supports, other than the "Get Special Functions" special function. The "Get Special Functions" special function MUST NOT be included in the response. Each special function MUST have a feature name and MAY have zero or more parameters as appropriate to each special function.

4.2.2 Special Function: Get File filename.xml

The "Get File" is a generic special function which allows the master to obtain an XML response as identified by the included filename.

The first parameter (for example, `filename.xml`) MUST be included, and identifies the nature of the data that will be returned by the EGM when the master sends this special function.

Optional parameters may be included as appropriate to the special function.

Upon receipt of this special function, the EGM MUST acknowledge it with a correctly formatted 0x84 IACR. Once the EGM is finished, the master may then retrieve the listing by sending a 0x03 LARQ command. The EGM should respond with a 0x83 LARR command containing the requested data.

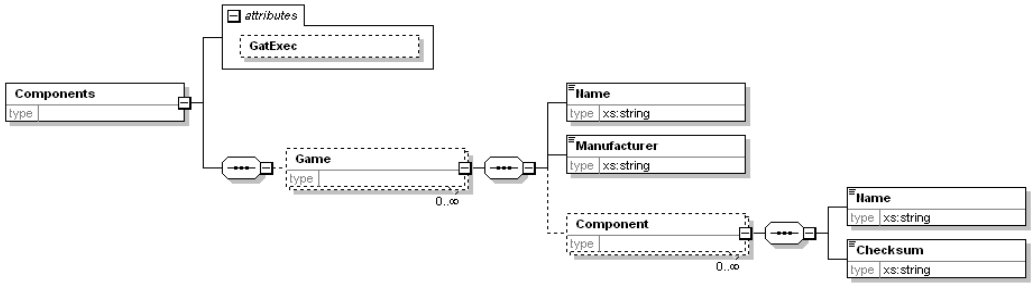
4.2.2.1 Get File AuthenticationResponse.xml %%SHA1_HMAC%%

Corrections in v3.50.1

All EGMs MUST include the "Get File" feature with the parameters `AuthenticationResponse.xml` and `%%SHA1_HMAC%%` in the response to the Get Special Functions command. This specific form of the "Get File" special function is used by the master to obtain the authentication results for all EGM components using a single request. Only components that can be authenticated using the SHA-1 and SHA-1 HMAC algorithms may be included in the response.

The specific wildcard parameter `%%SHA1_HMAC%%` MUST be included so the master may optionally provide a SHA-1 HMAC key. If the master does not provide a key (i.e wildcard replaced with "(none)"), the EGM MUST authenticate each component using the SHA-1 algorithm, not SHA-1 HMAC with a NULL or zero key. If the master does provide a key, the EGM MUST authenticate each component using the SHA-1 HMAC algorithm.

Once the EGM indicates that the results are available, the master may then retrieve the listing by sending a 0x03 LARQ command. The EGM's response MUST be XML formatted and conform to the following definition (See Appendix B for more details):



The EGM's response is a list of all components that were authenticated, along with the SHA-1 or SHA-1 HMAC authentication result for each component. This list MUST include, at a minimum, all EGM control programs and data as required by the jurisdiction. The EGM MAY include additional components, but all returned components MUST be authenticated using the SHA-1 or SHA-1 HMAC algorithm.

Authentication MUST be performed at the component level, not as a single result for all control programs and data, unless all control programs and data on an EGM are approved by the jurisdiction as a single unit, in which case, all control programs and data MAY be identified as a single component of the EGM.

The `GatExec` attribute MUST be set to `default` for ~~GAT v3.50.0~~ compatibility with this version of the GAT protocol.

The `Name` element for each component SHOULD be consistent with the naming convention used when submitting the component to a regulator or testing agency for approval.

4.2.3 Special Function: Component name %%SHA1_HMAC%%

The "Component" special function identifies an individual component on an EGM that may be authenticated using the SHA-1 and SHA-1 HMAC algorithms.

The "name" parameter MUST be included, and is used to identify the specific component that will be authenticated by the EGM when the master sends this special function. The "name" MUST be consistent with the naming convention used when submitting this component to a regulator or testing agency for approval.

The specific wildcard parameter "`%%SHA1_HMAC%%`" MUST be included so the master may optionally provide a SHA-1 HMAC key. If the master does not provide a key (i.e wildcard replaced with "(none)"), the EGM MUST use the SHA-1 algorithm, not SHA-1 HMAC with a NULL or zero key. If the master does provide a key, the EGM MUST use the SHA-1 HMAC algorithm.

Once the EGM indicates that the results are available, the master may then retrieve the listing by sending a 0x03 LARQ command. The EGM's response MUST be XML formatted and conform to the Components definition specified in Section 4.2.2.1. It MUST include only one `Component` element, providing the SHA-1 or SHA-1 HMAC authentication result as appropriate for the component named in the command.

The EGM MUST support a "Component" special function for each individual component that can be authenticated using the "Get File AuthenticationResponse.xml" special function. The EGM MAY include additional "Component" special functions, for example to authenticate sub-components or special groups of components. Only components capable of being authenticated using the SHA-1 and SHA-1 HMAC algorithms may be exposed using the "Component" special function.

Chapter 5

Operational Scenarios

5.1 Sample Get Special Functions request

Here is a sample communication session where the master makes a request of the supported special functions:

Table 5.1 0x04 – IACQ

Field	Hex Value	Description
Command	04	Initiate Authentication Calculation Query.
Length	1B	27 bytes.
Authentication Level	BA	Special function designator.
Authentication Parameter	00	Special function designator.
	47 65 74 20 53 70 65 63 69 61 6C 20 46 75 6E 63 74 69 6F 6E 73	"Get Special Functions" special function.
CRC	2B 54	16-bit CRC.

Table 5.2 0x84 – IACR

Field	Hex Value	Description
Command	84	Initiate Authentication Calculation Response.
Length	05	5 bytes.
Status	03	Request acknowledged and special function started.
CRC	B8 72	16-bit CRC.

Here is a sample communication session where an EGM reports its list of supported special functions:

Table 5.3 0x03 – LARQ

Field	Hex Value	Description
Command	03	Last Authentication Results Query.
Length	07	7 bytes.
Data Format	02	XML format requested.
Frame Number	00 01	Request the 1st frame of data.
CRC	74 01	16-bit CRC.

Table 5.4 0x83 LARR

Field	Hex Value	Description
Command	83	Last Authentication Results Response.
Length	Up to FF	Total length of command.
Status Data	00	No error, this is not the last frame.
Frame Number	00 01	Frame number 1.
Data		First frame of XML special functions list (up to 248 bytes). See Section 5.1.1, Example Get Special Functions Response.
CRC	00 00 – FF FF	16-bit CRC.

5.1.1 Example Get Special Functions Response

```
<?xml version="1.0"?>
<SpecialFunctions GatExec="default">
    <Function>
        <Feature>Get File</Feature>
        <Parameter>AuthenticationResponse.xml</Parameter>
        <Parameter>%%SHA1_HMAC%%</Parameter>
    </Function>
    <Function>
           <Feature>Component</Feature>
           <Parameter>ABC_boot_123</Parameter>
           <Parameter>%%SHA1_HMAC%%</Parameter>
    </Function>
    <Function>
           <Feature>Component</Feature>
           <Parameter>ABC_os_345.pkg</Parameter>
           <Parameter>%%SHA1_HMAC%%</Parameter>
    </Function>
    <Function>
           <Feature>Component</Feature>
           <Parameter>ABC_game_789_012.pkg</Parameter>
           <Parameter>%%SHA1_HMAC%%</Parameter>
    </Function>
</SpecialFunctions>
```

5.2 Example All Components Authentication Request

Here is a sample communication session where the master makes a request for the EGM to authenticate all components:

Table 5.5 0x04 IACQ

Field	Hex Value	Description
Command	04	Initiate Authentication Calculation Query.
Length	32	50 bytes.
Authentication Level	BA	Special function designator.
Authentication Parameter	00	Special function designator.
	47 65 74 20 46 69 6C 65 09 41 75 74 68 65 6E 74 69 63 61 74 69 6F 6E 52 65 73 70 6F 6E 73 65 2E 78 6D 6C 09 31 32 33 34 41 42 43 44	"Get File<09>AuthenticationResponse.xml<09>1234ABCD" special function.
CRC	F3 4B	16-bit CRC.

Table 5.6 0x84 IACR

Field	Hex Value	Description
Command	84	Initiate Authentication Calculation Response
Length	05	5 bytes
Status	03	Request acknowledged and special function started.
CRC	B8 72	16-bit CRC

Here is a sample communication session where an EGM reports the authentication results for all components:

Table 5.7 0x03 LARQ

Field	Hex Value	Description
Command	03	Last Authentication Results Query.
Length	07	7 bytes.
Data Format	02	XML format requested.
Frame Number	00 01	Request the 1st frame of data.
CRC	74 01	16-bit CRC.

Table 5.8 0x83 LARR

Field	Hex Value	Description
Command	83	Last Authentication Results Response.
Length	Up to FF	Total length of command.
Status Data	00	No error, this is not the last frame.
Frame Number	00 01	Frame number 1.
Data		First frame of XML authentication results (up to 248 bytes). See Section 5.2.1, Example All Components Authentication Response.
CRC	00 00 - FF FF	16-bit CRC.

5.2.1 Example All Components Authentication Response

```
<?xml version="1.0"?>
<Components GatExec="default">
    <Game>
        <Name>ABC</Name>
        <Manufacturer>A Better Company</Manufacturer>
        <Component>
            <Name>ABC_boot_123</Name>
            <Checksum>0833B58888612D2A37829F44B58A63FF32933FFF</Checksum>
        </Component>
        <Component>
            <Name>ABC_os_345.pkg</Name>
            <Checksum>AEC231D3EDF4D338F1F81DBAA98742A4D6278ECB</Checksum>
        </Component>
        <Component>
            <Name>ABC_game456_789_012.pkg</Name>
            <Checksum>377938A82F5DEA976D86119C1CD5B65EE9CE2413</Checksum>
        </Component>
    </Game>
</Components>
```

5.3 Example SHA-1 Authentication

Here is a sample communication session where the master makes a request for the EGM to perform a SHA-1 authentication for a "SHA1-Example" component:

The authentication calculation is based on a NIST example, where the "SHA1-Example" component consists of the 3 ASCII bytes:

"abc" or

616263

CSRC Home > Groups > ST > Cryptographic Toolkit

EXAMPLE ALGORITHMS

http://csrc.nist.gov/groups/ST/toolkit/examples.html

http://csrc.nist.gov/groups/ST/toolkit/documents/Examples/SHA1.pdf

Table 5.9 0x04 – IACQ

Field	Hex Value	Description
Command	04	Initiate Authentication Calculation Query.
Length	23	35 bytes.
Authentication Level	BA	Special function designator.
Authentication Parameter	00	Special function designator.
	43 6F 6D 70 6F 6E 65 6E 74 09 53 48 41 31 2D 45 78 61 6D 70 6C 65 09 28 6E 6F 6E 65 29	"Component<09>SHA1-Example<09>(none)" special function.
CRC	B8 BC	16-bit CRC.

Table 5.10 0x84 IACR

Field	Hex Value	Description
Command	84	Initiate Authentication Calculation Response.
Length	05	5 bytes.
Status	03	Request acknowledged and special function started.
CRC	B8 72	16-bit CRC.

Here is a sample communication session where an EGM reports the authentication result for the component:

Table 5.11 0x03 LARQ

Field	Hex Value	Description
Command	03	Last Authentication Results Query.
Length	07	7 bytes.
Data Format	02	XML format requested.
Frame Number	00 01	Request the 1st frame of data.
CRC	74 01	16-bit CRC.

Table 5.12 0x83 LARR

Field	Hex Value	Description
Command	83	Last Authentication Results Response.
Length	Up to FF	Total length of command.
Status Data	00	No error, this is not the last frame.
Frame Number	00 01	Frame number 1.
Data		First frame of XML authentication results (up to 248 bytes). See Section 5.3.1, Example SHA-1 Response.
CRC	0000 – FFFF	16-bit CRC.

5.3.1 Example SHA-1 Response

```
<?xml version="1.0"?>
<Components GatExec="default">
    <Game>
        <Name>ABC</Name>
        <Manufacturer>A Better Company</Manufacturer>
        <Component>
            <Name>SHA1-Example</Name>
            <Checksum>A9993E364706816ABA3E25717850C26C9CD0D89D</Checksum>
        </Component>
    </Game>
</Components>
```

5.4 Example SHA-1 HMAC Authentication

Here is a sample communication session where the master makes a request for the EGM to perform a SHA-1 HMAC authentication for a "SHA1-HMAC-Example" component:

The authentication calculation is based on a NIST example, where the "SHA1-HMAC-Example" component consists of the 34 ASCII bytes:

"Sample message for keylen<blocklen" or

5361 6D706C65 206D6573

73616765 20666F72 206B6579 6C656E3C 626C6F63 6B6C656E

And where the following 20-byte key is used:

00010203 04050607 08090A0B 0C0D0E0F 10111213

CSRC Home > Groups > ST > Cryptographic Toolkit

EXAMPLE ALGORITHMS

http://csrc.nist.gov/groups/ST/toolkit/examples.html

http://csrc.nist.gov/groups/ST/toolkit/documents/Examples/HMAC_SHA1.pdf

Table 5.13 0x04 – IACQ

Field	Hex Value	Description
Command	04	Initiate Authentication Calculation Query.
Length	48	72 bytes.
Authentication Level	BA	Special function designator.
Authentication Parameter	00	Special function designator.
	43 6F 6D 70 6F 6E 65 6E 74 09 53 48 41 31 2D 48 4D 41 43 2D 45 78 61 6D 70 6C 65 09 30 30 30 31 30 32 30 33 30 34 30 35 30 36 30 37 30 38 30 39 30 41 30 42 30 43 30 44 30 45 30 46 31 30 31 31 31 32 31 33	"Component<09>SHA1-HMAC-Example<09>000102030405060708090A0B0C0D0E0F10111213" special function.
CRC	13 3A	16-bit CRC.

Table 5.14 0x84 IACR

Field	Hex Value	Description
Command	84	Initiate Authentication Calculation Response.
Length	05	5 bytes.
Status	03	Request acknowledged and special function started.
CRC	B8 72	16-bit CRC.

Here is a sample communication session where an EGM reports the authentication result for the component:

Table 5.15 0x03 LARQ

Field	Hex Value	Description
Command	03	Last Authentication Results Query.
Length	07	7 bytes.
Data Format	02	XML format requested.
Frame Number	00 01	Request the 1st frame of data.
CRC	74 01	16-bit CRC.

Table 5.16 0x83 LARR

Field	Hex Value	Description
Command	83	Last Authentication Results Response.
Length	Up to FF	Total length of command.
Status Data	00	No error, this is not the last frame.
Frame Number	00 01	Frame number 1.
Data		First frame of XML authentication results (up to 248 bytes). See Section 5.4.1, Example SHA-1 HMAC Response.
CRC	0000 – FFFF	16-bit CRC.

5.4.1 Example SHA-1 HMAC Response

```
<?xml version="1.0"?>
<Components GatExec="default">
    <Game>
        <Name>ABC</Name>
        <Manufacturer>A Better Company</Manufacturer>
        <Component>
            <Name>SHA1-HMAC-Example</Name>
            <Checksum>4C99FF0CB1B31BD33F8431DBAF4D17FCD356A807</Checksum>
        </Component>
    </Game>
</Components>
```

Appendix A

CRC Calculation

A.1 CRC Calculation in Java

Corrections in v3.50.1

Here is an implementation of the CRC calculation in Java:

```
/**************************************************************************
* Reference:  http://www.gelato.unsw.edu.au/lxr/source/lib/crc16.c
*
* Uses irreducible polynomial:  1 + x^2 + x^15 + x^16
*
**************************************************************************/


public class CRC16
{
    private static int[] table = {
       0x0000, 0xC0C1, 0xC181, 0x0140, 0xC301, 0x03C0, 0x0280, 0xC241,
       0xC601, 0x06C0, 0x0780, 0xC741, 0x0500, 0xC5C1, 0xC481, 0x0440,
       0xCC01, 0x0CC0, 0x0D80, 0xCD41, 0x0F00, 0xCFC1, 0xCE81, 0x0E40,
       0x0A00, 0xCAC1, 0xCB81, 0x0B40, 0xC901, 0x09C0, 0x0880, 0xC841,
       0xD801, 0x18C0, 0x1980, 0xD941, 0x1B00, 0xDBC1, 0xDA81, 0x1A40,
       0x1E00, 0xDEC1, 0xDF81, 0x1F40, 0xDD01, 0x1DC0, 0x1C80, 0xDC41,
       0x1400, 0xD4C1, 0xD581, 0x1540, 0xD701, 0x17C0, 0x1680, 0xD641,
       0xD201, 0x12C0, 0x1380, 0xD341, 0x1100, 0xD1C1, 0xD081, 0x1040,
       0xF001, 0x30C0, 0x3180, 0xF141, 0x3300, 0xF3C1, 0xF281, 0x3240,
       0x3600, 0xF6C1, 0xF781, 0x3740, 0xF501, 0x35C0, 0x3480, 0xF441,
       0x3C00, 0xFCC1, 0xFD81, 0x3D40, 0xFF01, 0x3FC0, 0x3E80, 0xFE41,
       0xFA01, 0x3AC0, 0x3B80, 0xFB41, 0x3900, 0xF9C1, 0xF881, 0x3840,
       0x2800, 0xE8C1, 0xE981, 0x2940, 0xEB01, 0x2BC0, 0x2A80, 0xEA41,
       0xEE01, 0x2EC0, 0x2F80, 0xEF41, 0x2D00, 0xEDC1, 0xEC81, 0x2C40,
       0xE401, 0x24C0, 0x2580, 0xE541, 0x2700, 0xE7C1, 0xE681, 0x2640,
       0x2200, 0xE2C1, 0xE381, 0x2340, 0xE101, 0x21C0, 0x2080, 0xE041,
       0xA001, 0x60C0, 0x6180, 0xA141, 0x6300, 0xA3C1, 0xA281, 0x6240,
       0x6600, 0xA6C1, 0xA781, 0x6740, 0xA501, 0x65C0, 0x6480, 0xA441,
       0x6C00, 0xACC1, 0xAD81, 0x6D40, 0xAF01, 0x6FC0, 0x6E80, 0xAE41,
       0xAA01, 0x6AC0, 0x6B80, 0xAB41, 0x6900, 0xA9C1, 0xA881, 0x6840,
       0x7800, 0xB8C1, 0xB981, 0x7940, 0xBB01, 0x7BC0, 0x7A80, 0xBA41,
       0xBE01, 0x7EC0, 0x7F80, 0xBF41, 0x7D00, 0xBDC1, 0xBC81, 0x7C40,
       0xB401, 0x74C0, 0x7580, 0xB541, 0x7700, 0xB7C1, 0xB681, 0x7640,
       0x7200, 0xB2C1, 0xB381, 0x7340, 0xB101, 0x71C0, 0x7080, 0xB041,
       0x5000, 0x90C1, 0x9181, 0x5140, 0x9301, 0x53C0, 0x5280, 0x9241,
       0x9601, 0x56C0, 0x5780, 0x9741, 0x5500, 0x95C1, 0x9481, 0x5440,
       0x9C01, 0x5CC0, 0x5D80, 0x9D41, 0x5F00, 0x9FC1, 0x9E81, 0x5E40,
       0x5A00, 0x9AC1, 0x9B81, 0x5B40, 0x9901, 0x59C0, 0x5880, 0x9841,
       0x8801, 0x48C0, 0x4980, 0x8941, 0x4B00, 0x8BC1, 0x8A81, 0x4A40,
       0x4E00, 0x8EC1, 0x8F81, 0x4F40, 0x8D01, 0x4DC0, 0x4C80, 0x8C41,
       0x4400, 0x84C1, 0x8581, 0x4540, 0x8701, 0x47C0, 0x4680, 0x8641,
       0x8201, 0x42C0, 0x4380, 0x8341, 0x4100, 0x81C1, 0x8081, 0x4040,
    };


    public static int hash(byte[] bytes)
    {
```

```
        int crc = 0xFFFF;  // See Section 3.1 of SVC protocol spec

        for (byte b : bytes) {
            crc = (crc >>> 8) ^ table[(crc ^ b) & 0xff];
        }

        return crc;
    }

}
```

Appendix B

XSD for SpecialFunctions and Components

B.1 XSD

The following XML Schema Definition (XSD) identifies the proper syntax for the SpecialFunctions and Components XML data structures.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
    attributeFormDefault="unqualified">
    <!--GAT3 XML Structures.-->
    <xs:element name="SpecialFunctions">
        <xs:annotation>
            <xs:documentation>List of special functions.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Function" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Feature" type="xs:string"/>
                            <xs:element name="Parameter" type="xs:string" minOccurs="0"
                                maxOccurs="unbounded"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="GatExec" type="xs:string" use="optional" default="default"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Components">
        <xs:annotation>
            <xs:documentation>List of components and signatures.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Game" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Name" type="xs:string"/>
                            <xs:element name="Manufacturer" type="xs:string"/>
                            <xs:element name="Component" minOccurs="0" maxOccurs="unbounded">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="Name" type="xs:string"/>
                                        <xs:element name="Checksum" type="xs:string"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="GatExec" type="xs:string" use="optional" default="default"/>
        </xs:complexType>
    </xs:element>
    <!--End of Schema.-->
</xs:schema>
```

END OF DOCUMENT

Document ID: gsa-p0093.002.00

Released: 2011/02/13