Chapter 21

Look Inside

voucher Class

21.1 Introduction

The `voucher` class is used to manage the process of issuing and redeeming payment *vouchers* (sometimes referred to as tickets or coupons) at an EGM. Vouchers can be `cashable`, `promotional`, or `nonCashable`. The `voucher` class includes commands and events to issue and redeem vouchers, and to set validation identifiers. The class also includes commands to retrieve the `voucher` history log maintained by the EGM.

When a player cashes out at an EGM and the EGM is configured for printing payment vouchers, the EGM may issue vouchers in lieu of coin or other payment. A voucher may only contain one type of credit. Thus, if multiple types of credits (`cashable`, `promotional`, `nonCashable`) are on the EGM when the player cashes out, the EGM may have to produce multiple vouchers. Vouchers issued by one EGM may be redeemed at another EGM. If the credits are `cashable` or `promotional`, vouchers can be redeemed at a cashier's station or at a self-service kiosk.

21.1.1 Device Class Information: Single-Device

Extension in v3.0: g2s3

The `voucher` class is a single-device class. The EGM MUST only expose one active `voucher` device. Class-level meters and logs MUST include activity related to both active and inactive devices. Inactive devices may be exposed through the `commConfig` class. Transaction logs associated with the `voucher` class MUST be maintained at the class level.

21.1.2 allowVoucherIssue and allowVoucherRedeem Attributes

Extension in v2.0.0: g2s1

The `voucher` class supports both issuing and redeeming vouchers. The `allowVoucherIssue` profile attribute controls functionality related to validation data for a `voucher` device; it does not (directly) control voucher issuance. However, without validation data, a `voucher` device will be unable to issue cash-out vouchers and other types of generic vouchers.If `allowVoucherIssue` is set to `false` for a device:

- The EGM MUST NOT place any `getValidationData` commands for the `voucher` device in the outbound queue.
- The EGM MUST NOT generate the `G2S_VCE101 Validation ID Data Expired` event for the `voucher` device.
- The EGM MUST NOT disable the `voucher` device due to not having validation ID data.

The `allowVoucherRedeem` profile attribute controls whether a `voucher` device MAY be used for voucher redemption. If `allowVoucherRedeem` is set to `false`, the EGM MUST NOT generate the `redeemVoucher` command for the `voucher` device.

21.2 Transaction Identifiers

Within the G2S protocol, voucher redemption is a four-step process:

1. **Request**: The EGM requests permission from the host to redeem a voucher.
2. **Authorize**: The host authorizes the EGM to redeem a voucher.
3. **Commit**: The EGM notifies the host that it has committed a voucher value to the credit meter or rejected the voucher.
4. **Acknowledge**: The host acknowledges that it received a commitment command.

The commands used to redeem vouchers include a `transactionId` attribute. The `transactionId` attribute is used to link all four parts together. When an EGM initiates a voucher redemption, the EGM includes the `transactionId` in the initial command sent to the host. Both the host and the EGM include the `transactionId` in all subsequent commands related to the redemption process. For consistency, the `transactionId` is also included in the commands used to issue vouchers.

The EGM MUST generate transaction identifiers as a sequence that strictly increases by 1 (one). The EGM MUST maintain the counters used to generate transaction identifiers in persistent storage. A single counter MUST be used for all protocol-related transactions within the EGM. See Chapter 1 for more details on transaction identifiers.

21.3 Seeds and Validation IDs

Within the G2S protocol, the `voucher` class has been designed to support a wide range of operating requirements. Many of these requirements relate to the ability of an EGM to produce vouchers after it has lost communications with the validation system — that is, when the validation system is offline (see Section 21.3.2, Validation System Offline, for more details). Within the G2S protocol, configuration parameters support many different scenarios for offline behavior. The scenarios revolve around three key concepts:

- The number of vouchers an EGM may produce without host interaction,
- The amount of time an EGM can produce vouchers without host interaction, and
- Whether vouchers can be produced at all while offline.

To support issuing vouchers, the EGM is provided a set of validation identifiers and a seed value for each identifier. The seed values are comprised of from 0 (zero) to 20, UTF-8 encoded characters in the range of U+0020 to U+007E (ASCII printable characters). Seed values outside this range will cause validation errors. Validation seeds are used only for manual authentication (see Section 21.4). Validation identifiers are 18-digit numbers printed on the voucher in both human-readable and bar-coded form. This information MUST be stored in persistent memory.

NOTE:
While validation identifiers are required to be 18-digit numbers when printed on vouchers, in some elements, the validation identifiers MUST be masked for security reasons. In such cases, the leftmost 14 (fourteen) digits MUST be masked and replaced with non-numeric values. For example, in the `voucherLog` element, the validation identifier "`12345678901234567 8`" MUST be masked and a value such as "`xxxxxxxxxxxxxx5678`" MUST be reported.

The EGM can only produce vouchers so long as it has unused validation identifiers available. To prevent an EGM from running out of validation identifiers, the host may specify a threshold at which the EGM MUST request additional validation identifiers. The validation identifiers should not be a continuous sequence. The programmers of host systems are free to use their ingenuity to reduce the predictability of the validation identifiers and, thus, enhance security.

It is anticipated that the `validationId` list will be managed in a manner which will not waste identifiers, and which will not inadvertently place the EGM in a state where it does not have any identifiers available. To facilitate this, the normal process is for the host to add to the validation identifiers currently remaining on the EGM. In this way the EGM will always have the data necessary to print a cash-out voucher.

If an EGM is permitted to print offline vouchers, it may do so until either it has used up the available validation identifiers or the `valIdListLife` timer expires. Note that, even if an EGM is not permitted to print offline vouchers, it is always possible that an EGM will print a voucher and then be unable to report that voucher because the host just went offline. To enable the EGM to make full use of all available validation identifiers, the host should ensure the maximum number of log entries is greater than the maximum number of validation identifiers given to the EGM. A large number of validation identifiers will enable the EGM to print vouchers offline for an extended period of time. However, if the seeds and voucher identifiers on an EGM are ever compromised, offline vouchers will no longer be secure. The `valIdListRefresh` attribute can be used to limit this exposure. The host may then decide whether to allow the EGM to keep the existing validation data or replace it with new data.

The host should carefully select the values for the `valIdListLife` and `valIdListRefresh` attributes. Values that are too small may cause frequent and unnecessary requests for new validation identifiers. This may make the issuance of vouchers at the EGM difficult and may cause excessive traffic at the host. As noted above, values that are too large may compromise the security of the validation identifiers.

Even in normal online scenarios, an EGM that is not getting much play may have the same set of validation identifiers and seeds for an extended period of time. To improve the security of the validation seeds, the host may want to periodically change the validation identifiers and seeds. The host may use the `deleteCurrent` attribute to send the EGM a completely new set of validation identifiers and seeds. To prevent the possibility of a `validationId` being used twice, the new list should never include any validation identifiers that are potentially currently on the EGM.

With the `deleteCurrent` attribute set to `true`, the EGM will delete its current list and replace it with the new list in a single operation. In this way, the EGM will never be in a position where it will not have validation identifiers available for a requested cash-out. The EGM MUST NOT issue event `G2S_VCE102 Validation ID Data Updated`, until all outstanding `issueVoucher` commands have been acknowledged. It is recommended that a host not arbitrarily "burn" validation identifiers. After the host has received the `G2S_VCE102 Validation ID Data Updated` event and verified which of the previous validation identifiers were actually used, the host may return the unused identifiers to its pool of available identifiers.

In regards to vouchers, the proper start up sequence for the EGM is:

1. Retry unacknowledged voucher transactions.
2. Generate a `getValidationData` request to inform the host of the current status of the validation identifier list.

Under no circumstances should an EGM request validation identifiers from the host unless the host has acknowledged all `issueVoucher` commands. This can be determined from the EGM's `voucher` transaction log.

21.3.1 allowVoucherIssue Attribute

Extension in v2.0.0: g2s1

When the `allowVoucherIssue` profile attribute is set to `false` for a `voucher` device:

- The EGM MUST NOT place any `getValidationData` commands for the `voucher` device in the outbound queue.
- The EGM MUST NOT generate the `G2S_VCE101 Validation ID Data Expired` event for the `voucher` device.
- The EGM MUST NOT disable the `voucher` device due to not having validation ID data.

21.3.2 Validation System Offline

Extension in v3.0: g2sVSO

In the G2S protocol, host offline is typically indicated by the `transportState` attribute of the `commsStatus` command being set to something other than `G2S_transportUp` for the host's `communications` device. This may occur when the no-response timer expires or due to an issue with the underlying transport. See Section 2.4.8, Transport-Related Events, for more details.

However, the validation database, which contains issued vouchers, may be on a different physical server from the server acting as the `voucher` host. Regulatory requirements dealing with offline voucher operation are not concerned with whether the EGM is communicating with the `voucher` host; they are concerned with whether the EGM is communicating with the validation system — that is, whether data for a voucher issued by the EGM can be stored in the validation database in a timely manner.

Clearly, if the EGM has lost communications with the `voucher` host, it has also lost communications with the validation system, whether that is the same server or a different server. However, if the validation database is on a separate server, it is possible for the validation system to be down even though the `voucher` host is still communicating with the EGM.

To support EGM compliance with regulations that prohibit the EGM from printing more than one voucher when the validation system is down, it is essential that the `voucher` host not generate an `issueVoucherAck` command until the validation data from the `issueVoucher` command is stored in the validation database.

With this extension, when the EGM issues a voucher, the EGM starts a timer using the value from the `noAckTimer` attribute of the `voucherProfile` command. If the timer expires before the EGM receives the `issueVoucherAck` command for the voucher or the `transportState` attribute of the voucher host's `communication` device is set to anything other than `G2S_transportUp`, the EGM sets the `systemOnLine` attribute of the `voucherStatus` command to `false` and generates event `G2S_VCE112 Validation System Offline`. Once the `transportState` attribute is set to `G2S_transportUp` and all issued vouchers have been acknowledged, the `systemOnLine` attribute is set to `true` and the event `G2S_VCE113 Validation System Not Offline`. See Section 21.11.1.1, systemOnLine Attribute, for more details.

The `printOffLine`, `maxOffLinePayOut`, and `printNonCashOffLine` attributes of the `voucherProfile` command require specific behavior when the validation system is offline. While the `systemOnLine` attribute is set to `false`, the EGM MUST consider the validation system offline and comply with the behavior specified for those attributes.

NOTE:
Even if an EGM does not support this extension, it SHOULD still assume that a delay in receiving the `issueVoucherAck` command (15 seconds recommended) means the validation system is offline. Otherwise, the EGM may not be compliant with regulations that prohibit offline vouchers.

21.3.2.1 Offline Handpay Vouchers

Extension in v3.0: g2sVSO1

G2S supports a secure offline voucher process, which allows vouchers to be printed and delivered directly to the player even when the validation system is offline. However, not all jurisdictions allow an EGM to print a voucher in direct response to a player cashout request when the validation system is offline. This extension is specifically for use in jurisdictions that only allow an EGM to print an offline voucher after the EGM has locked up in a handpay request and an attendant has turned the reset key. Offline vouchers generated in this way are referred to as Offline Handpay Vouchers.

To satisfy these requirements, if the `enableHandpayVoucher` attribute of the `voucherProfile` command is set to `true` and the validation system is offline (that is, the `systemOnLine` attribute of the `voucherStatus` command is `false`) and there is no other cashout method available, the EGM MUST lock up and generate a cancel credit handpay request (`G2S_cancelCredit`) in response to a player-initiated cashout request.

In such cases, if the `printOffLine` attribute of the `voucherProfile` command is set to `true`, the EGM MAY print an Offline Handpay Voucher when the attendant keys off the handpay request. If the attendant keys off the handpay request to an Offline Handpay Voucher, the EGM MUST generate one or more Offline Handpay Vouchers, as necessary, to satisfy the cashout request and remove the credits from the EGM.

Offline Handpay Vouchers contain the same validation information as standard vouchers. However, the barcode MUST be offset on Offline Handpay Vouchers to prevent the Offline Handpay Vouchers from being redeemed at an EGM. In addition, the EGM MUST use the title specified in the `titleHandpayVoucher` attribute of the `voucherProfile` command rather than the title that would be printed on a standard offline voucher in normal circumstances. The validation system does not need to treat the Offline Handpay Vouchers differently than standard vouchers.

When configured to generate Offline Handpay Vouchers, the EGM MUST ignore the `valIdListLife` timer regardless of whether the validation system is online or offline. Even if the `valIdListLife` timer has expired for the current set of validation IDs, the EGM MUST still use the validation IDs to generate Offline Handpay

Vouchers. In such cases, event `G2S_VCE101 Validation ID Data Expired` MUST NOT be generated. Requirements related to the `maxOffLinePayOut` and `printNonCashOffLine` attributes of the `voucherProfile` command MUST still be enforced.

In addition, to assure that there are always validation IDs available for generating Offline Handpay Vouchers, when the `enableHandpayVoucher` attribute of the `voucherProfile` command is set to `true`, the EGM MUST disable the voucher device (that is, set the `egmEnabled` attribute of the voucher device to `false`) if the number of available validation IDs is less than two. If the voucher device is required for play, this action will cause the EGM to be disabled.

Subsequently, after the EGM has disabled the voucher device, if additional credits are on the EGM, the EGM MUST ignore the `egmEnabled` attribute and continue to use the available validation IDs to produce Offline Handpay Vouchers for player-initiated cashouts until the supply of validation IDs is completely depleted. The `egmEnabled` attribute is set to false simply as a means of triggering the required-for-play behavior and disabling the EGM.

21.4 Manual Authentication

A key feature of the G2S protocol is the ability to manually authenticate offline vouchers produced by EGMs. This procedure is used in cases where an EGM produced a voucher but did not communicate the voucher information to the host; and thus, the host does not have a record of the voucher.

A 32-character manual authentication identifier MUST, if possible, be printed on every voucher for cashable and promotional credits produced by the EGM. The authentication identifier is derived from a 128-bit MD5 hash of the EGM identifier, validation identifier, voucher amount, and seed value. If an EGM cannot print a manual authentication identifier on vouchers, it MUST NOT print offline vouchers. Operational circumstances MAY prevent the EGM from printing manual authentication codes. For example, the operator might choose to use a printer that does not have voucher templates that support manual authentication codes, or the operator might configure the EGM to not print manual authentication codes. If an EGM is not configured to print manual authentication codes on vouchers, it MUST NOT allow the `printOffLine` attribute of the `voucherProfile` to be set to `true`.

Each validation identifier that the host system sends to an EGM also includes an associated seed value. At the time of issuance, the authentication identifier and the voucher amount are printed on the voucher. To authenticate a voucher, the validation identifier and amount are entered into the host system. Because the host system knows the EGM identifier and seed, it can create its own authentication identifier. The authentication identifier recreated by the host can be compared to the authentication identifier printed on the voucher. If they match, the voucher can be presumed valid for the amount entered with an extremely high probability.

The following procedure is used to produce the authentication identifier:

1. Produce a string composed of, left to right, the
    - EGM identifier, 32 8 bit printable ASCII characters (U+0020 to U+007E), ASCII "0" (U+0030) padded right
    - validation identifier, 18 Numeric ("0" through "9") ASCII characters (U+0030 to U+0039)
    - seed value, 20, UTF-8 encoded characters in the range of U+0020 to U+007E, ASCII "0" (U+0030) padded left
    - the numerical value of the voucher amount as printed on the voucher, in cents, with no punctuation or currency sign, 20 characters ASCII "0" (U+0030) padded left. For example, a voucher with a printed value of $107.35 would result in a string of: 00000000000000010735
2. Convert all lower case characters a-z in the composed string to upper case ASCII characters.
3. Produce a 128-bit hash value with the MD5 algorithm using the 90-character string as input.
4. Produce the authentication identifier by casting the resulting hash value into a 32-character hex representation. Convert all alpha characters to their upper case form.

21.4.1 printNonCashOffLine Attribute

Extension in v2.0.0: g2s1

A 32-character manual authentication identifier is printed on every voucher for cashable and promotional credits produced by an EGM. If the `printNonCashOffLine` attribute is set to `true` in the `voucherProfile`, an authentication identifier is also printed on all vouchers for non-cashable credits.

21.4.2 Test Cases for Voucher Authentication Algorithm

The following tables list the test cases for the G2SVoucher Authentication ID algorithm. Note that the specification does not include the dashes (-) in the Authentication ID. They have been added to improve readability.

Table 21.1 Test Case 1 for Voucher Authentication Algorithm

	Value	String Component
EGM ID	EGM-1111	EGM-11110000000000000000000000000
Validation ID	800000000000000151	800000000000000151
Seed	13857577	0000000000013857577
Amount	5000	0000000000000005000
Authentication ID	F723-13F5-2751-E731-5B56-49B7-0895-7CCD	

Table 21.2 Test Case 2 for Voucher Authentication Algorithm

	Value	String Component
EGM ID	EGM-1111	EGM-11110000000000000000000000000
Validation ID	800000000000000248	800000000000000248
Seed	13857577	0000000000013857577
Amount	5000	0000000000000005000
Authentication ID	2E04-3435-E6B3-E319-FDB3-EEED-CF5D-6192	

Table 21.3 Test Case 3 for Voucher Authentication Algorithm

	Value	String Component
EGM ID	EGM-1234	EGM-12340000000000000000000000000
Validation ID	899999999999999906	899999999999999906
Seed	98765432ABC	0000000098765432ABC
Amount	100000	0000000000000100000
Authentication ID	74BD-304C-D77B-37EE-ECBC-4CA6-F994-C027	

Table 21.4 Test Case 4 for Voucher Authentication Algorithm (Sheet 1 of 2)

	Value	String Component
EGM ID	EGM-8989	EGM-89890000000000000000000000000
Validation ID	875693785693753584	875693785693753584
Seed	X93859785	00000000000X93859785

Table 21.4 Test Case 4 for Voucher Authentication Algorithm (Sheet 2 of 2)

	Value	String Component
Amount	255555	00000000000000255555
Authentication ID	6DE2-D548-C077-F77A-8A60-EC43-6743-3CFA	

Table 21.5 Test Case 5 for Voucher Authentication Algorithm

	Value	String Component
EGM ID	EGM-ID-9483285935	EGM-ID-948328593500000000000000
Validation ID	800008583986534043	800008583986534043
Seed	3466wst533	00000000003466WST533
Amount	99999999	00000000000099999999
Authentication ID	D41C-5D1A-583F-FA40-BC40-EAF6-A763-9B52	

Table 21.6 Test Case 6 for Voucher Authentication Algorithm

	Value	String Component
EGM ID	EGM-ID-8953435654	EGM-ID-895343565400000000000000
Validation ID	811111111111111141	811111111111111141
Seed	346653374	00000000000346653374
Amount	87654321	00000000000087654321
Authentication ID	CEAF-3411-7CFC-1CBD-84E9-EFE8-EC71-2DC7	

21.5 Transaction Logs

The G2S protocol requires that an EGM maintain logs of critical transactions in persistent storage. In case of power or communications outages, these logs are used to preserve and recover critical transactions. When an EGM is recovering from a power outage, communications outage, or other event that may have caused commands to be lost, the EGM MUST scan the logs of critical transactions and retry any transactions for which the EGM has not received and processed a response command from the host. Transaction logs are maintained on a class-by-class basis; all of the transactions related to a particular class are contained within class-level logs. The number of critical transactions that MUST be maintained in persistent storage by the EGM may be configured on a device-by-device basis. In general, a log containing the last 35 transactions is recommended.

When recovering from an outage, an EGM MUST retry all critical transactions for which the EGM has not received and processed a response command from the host. Due to timing differences, this process may cause the EGM to send duplicate commands to the host. The host MUST be aware of this possibility and be prepared to detect duplicate logical commands coming from an EGM.

Within the `voucher` class, the EGM MUST store critical data related to voucher issuance and redemption operations in persistent memory. This data MUST include all data necessary to construct accurate voucher commands when normal operations are resumed following an EGM failure. See Chapter 1 for more details on transaction logs.

21.6 Log Sequence Numbers

The EGM MUST assign a log sequence number, `logSequence`, to each entry in its `voucher` log. A single log is used for all devices within the `voucher` class. The EGM MUST generate the log sequence numbers as a sequence that strictly increases by 1 (one) starting at `1` (one). The EGM MUST maintain the counter used to generate the log sequence numbers in persistent memory. Within a single transaction log, the log sequence numbers MUST appear as an unbroken series that strictly increases by 1 (one). See Chapter 1 for more details on log sequence numbers.

Log sequence numbers identify the sequence of transactions within a single transaction log. Transaction identifiers uniquely identify an individual transaction. Both are generated as series that strictly increases by 1 (one); however, they serve two completely different purposes.

A `redeemVoucher` request may result in the voucher being returned to the player. Like any other voucher transaction, the EGM MUST record such a transaction in persistent memory until it is acknowledged. However, a player could repeatedly insert an invalid voucher, causing the contents of the `voucher` log to be filled with voucher rejections. All other transactions would be flushed. For this reason, if the most recent transaction in the `voucher` log is from a failed redemption request that has been committed and acknowledged, the EGM MUST overwrite that log entry with any subsequent `voucher` transactions. In this case, the `logSequence` MUST NOT be incremented. The same `logSequence` MUST be used for the subsequent transaction. However, a new `transactionId` MUST be assigned. If the most recent transaction in the `voucher` log is from a failed redemption request where the voucher was rejected, and the transaction has not been acknowledged then the EGM MUST initiate a new log entry, assigning a new `logSequence` and a new `transactionId`. The EGM MUST NOT initiate a redemption request using a `voucher` device when there are any unacknowledged transactions in the log for the same `voucher` device.

- Thus, the `voucher` log MAY contain more than one unacknowledged redemption request, but MUST NOT contain more than one unacknowledged redemption request for any particular `voucher` device.

21.7 Request-Response Pairs

The following tables organize the commands contained within the `voucher` class into request-response pairs:

- Unless an error code is being reported, the specified response MUST be generated as the response to the specified request.
- If the "Owner" column indicates "Yes", the specified response MUST NOT be generated by the EGM unless the specified request is sent from the owner of the device. See Section 1.10.3, Registered Hosts, for more details.
- If the "Guest" column indicates "Yes", the specified response MUST NOT be generated by the EGM unless the specified request is sent from the owner or a guest of the device. See Section 1.10.3, Registered Hosts, for more details.
- For commands originated by the host, if the "OK When Disabled" column indicates "Yes", the specified response MUST be generated by the EGM, as appropriate, when the device is disabled. See Section 1.10.6.3, Commands Not Allowed When Device Disabled, for more details.
- For commands originated by the EGM, if the "OK When Disabled" column indicates "Yes", the specified request or notification MUST be generated by the EGM, as appropriate, when the device is disabled. See Section 1.10.6.3, Commands Not Allowed When Device Disabled, for more details.

Table 21.7 Commands Originated by EGM

Request	Response	OK When Disabled
`getValidationData`	`validationData`	*
`issueVoucher`	`issueVoucherAck`	Yes
`redeemVoucher`	`authorizeVoucher`	No
`commitVoucher`	`commitVoucherAck`	Yes

* The EGM MUST NOT generate `getValidationData` commands when `hostEnabled` is set to `false`. The EGM MAY generate `getValidationData` commands normally when `hostEnabled` is set to `true`, regardless of the value of `egmEnabled`.

Table 21.8 Commands Originated by Host

Request	Response	Owner	Guest	OK When Disabled
`setVoucherState`	`voucherStatus`	Yes	No	Yes
`setVoucherLockOut`	`voucherStatus`	Yes	No	Yes*
`getVoucherStatus`	`voucherStatus`	Yes	Yes	Yes
`getVoucherProfile`	`voucherProfile`	Yes	Yes	Yes
`getVoucherLogStatus`	`voucherLogStatus`	Yes	Yes	Yes
`getVoucherLog`	`voucherLogList`	Yes	Yes	Yes

* EGMs MUST NOT process this request for a disabled device if the `lockOut` attribute for the request is set to `true`; when the `lockOut` attribute is set to `false`, the EGM MUST process the request as if the device were enabled. If the EGM does not process a request because the device is disabled then the EGM MUST generate a `G2S_APX016 Command Not Processed, Device Disabled` error response.

21.7.1 Owner-Controlled Parameters

The following table identifies the owner-controlled parameters for this class and the required behavior when those parameters are required to be reset.

Table 21.9 Owner-Controlled Parameters

Command	Required Behavior
`setVoucherState`	Unchanged.
`setVoucherLockOut`	If `hostLocked` is set to `true`, `hostLocked` MUST be set to `false` and event `G2S_VCE010 Device Not Locked by Host` MUST be generated; otherwise unchanged.
`validationData`	If the validation ID data is not expired, the validation ID data MUST be expired and event `G2S_VCE101 Validation ID Data Expired` MUST be generated; otherwise unchanged.
`authorizeVoucher`	If voucher redemptions are pending or in process, if possible, the voucher redemptions MUST be aborted and event `G2S_VCE109 Voucher Rejected` MUST be generated with `egmException` set to `99`; otherwise unchanged.

21.8 setVoucherState Command

21.8.1 Command Description

This command is used by a host to enable or disable the voucher functionality of an EGM. Only the owner of the device can execute this command. The `voucherStatus` command is generated in response to the `setVoucherState` command.

The `voucher` device MAY be disabled at any time by the host. If the `voucher` device is disabled while a voucher-related transaction is active, the EGM MUST, if possible, abort the transaction and report the result of the transaction to the host. If the transaction has already been committed and cannot be aborted, the EGM MUST continue processing the transaction and report the result of the transaction to the host. In general, while a device is disabled, the EGM MUST attempt to report each untried transaction for the device once. If such a transaction is not acknowledged by the host, the EGM SHOULD wait until the device is re-enabled before retrying that transaction. While the `voucher` device is disabled, the EGM MUST process any acknowledgements received from the host as normal. If the `idReader` device associated with the `voucher` device is disabled, the `voucher` device MUST act as if no ID is present. While the `voucher` device is disabled, the EGM MUST NOT initiate any new transactions related to the `voucher` device.

When the `voucher` device is disabled by the host — that is, `hostEnabled` is set to `false` — the `valIdListLife` timer in the `voucherProfile` MUST be expired to allow an orderly restart of the `voucher` device by the host. The `valIdListLife` timer MUST NOT be expired when `egmEnabled` is set to `false`.

The text message contained in the `disableText` attribute becomes eligible for display when the device becomes disabled — that is, following the `G2S_VCE003 Device Disabled by Host` event. The text message is no longer eligible for display once the device is re-enabled — that is, following the `G2S_VCE004 Device Not Disabled by Host` event. The text message is superseded by a subsequent `setVoucherState` command for the same device. If the text message is empty, the text message MUST NOT be displayed. See Section 3.7 for more details regarding the display of text messages.

21.8.2 Attribute and Element Detail

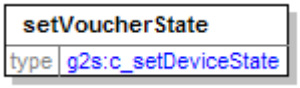


Table 21.10 setVoucherState Attributes

Attribute	Restrictions	Description

`enable`	`type: xs:boolean`
	`use: optional`
	`default: true`
		Indicates whether vouchers should be enabled.

`disableText`	`type: t_textMessage`
	`use: optional`
	`default: <empty>`
		Text message to display while the device is disabled.

21.9 setVoucherLockOut Command

21.9.1 Command Description

This command is used by a host to lock the EGM. See Section 3.4, Disable, Lockout, and Cabinet State, for more details. Only the owner of the device can execute this command. The `voucherStatus` command is generated in response to the `setVoucherLockOut` command.

The text message contained in the `lockText` attribute becomes eligible for display when the EGM is locked by the device — that is, following the `G2S_CBE211 Host Action Locked EGM` event. The text message is no longer eligible for display once the EGM is no longer locked by the device — that is, following the `G2S_VCE010 Device Not Locked by Host` event. The text message is superseded by a subsequent `setVoucherLockOut` command for the same device. If the text message is empty, the text message MUST NOT be displayed. See Section 3.7 for more details regarding the display of text messages.

21.9.2 Attribute and Element Detail

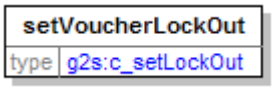


Table 21.11 setVoucherLockOut Attributes

Attribute	Restrictions	Description

`lockOut`	`type: xs:boolean`
	`use: optional`
	`default: false`
		Indicates whether the EGM should be locked.

`lockText`	`type: t_textMessage`
	`use: optional`
	`default: <empty>`
		Text message to display while the EGM is locked by the device.

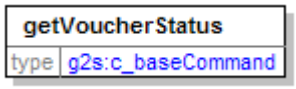
`lockTimeOut`	`type: t_lockTimeOut`
	`use: optional`
	`default: 1000`
		Maximum duration of the lock, in milliseconds. If a lock has not been removed within the specified time, the EGM should remove the lock.

21.10 getVoucherStatus Command

21.10.1 Command Description

This command is used by a host to request the current status of the voucher functionality from the EGM. The `voucherStatus` command is generated in response to `getVoucherStatus`.

21.10.2 Attribute and Element Detail



The `getVoucherStatus` command has no attributes or sub-elements.

21.11 voucherStatus Command

21.11.1 Command Description

This command is used by an EGM to report the current status of the `voucher` device. The `voucherStatus` command is generated in response to `getVoucherStatus`, `setVoucherState`, and `setVoucherLockOut` commands.

21.11.1.1 systemOnLine Attribute

Extension in v3.0: g2sVSO

The `systemOnLine` attribute indicates whether the validation system is online. The `systemOnLine` attribute MUST be set to `false` when either of the following occurs:

- The `transportState` attribute of the `commsStatus` command for the voucher host's communication device is set to anything other than `G2S_transportUp`.
- The EGM issues a voucher and does not receive an `issueVoucherAck` command for the voucher within the time interval specified by the `noAckTimer` attribute of the `voucherProfile`.

Whenever the state of the `systemOnLine` attribute changes from `true` to `false`, the EGM MUST generate event `G2S_VCE112 Validation System Offline`.

The `systemOnLine` attribute MUST be set to `true` when both of the following conditions are `true`:

- The `transportState` attribute is set to `G2S_transportUp`.
- The EGM has received `issueVoucherAck` commands for all issued vouchers.

Whenever the state of the `systemOnLine` attribute changes from `false` to `true`, the EGM MUST generate event `G2S_VCE113 Validation System Not Offline`.

21.11.2 Attribute and Element Detail

Table 21.12 voucherStatus Attributes (Sheet 1 of 2)

Attribute	Restrictions	Description

`configurationId`	`type: t_configurationId`
	`use: optional`
	`default: 0`
		Configuration identifier set by the `optionConfig` host.

`egmEnabled`	`type: xs:boolean`
	`use: optional`
	`default: true`
		Indicates whether the device has been enabled by the EGM.

`hostEnabled`	`type: xs:boolean`
	`use: optional`
	`default: true`
		Indicates whether the device has been enabled by the `voucher` host.

`hostLocked`	`type: xs:boolean`
	`use: optional`
	`default: false`
		Indicates whether the `voucher` device has been locked by the host.

Table 21.12 voucherStatus Attributes (Sheet 2 of 2)

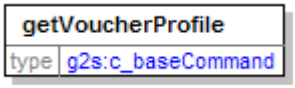
Attribute	Restrictions	Description
validationListId	type: t_validationListId use: required	Validation list identifier sent by the host in the most recent validationData command.
Extension in v3.0: g2sVSO		
systemOnLine	type: xs:boolean use: optional default: true	Indicates whether the validation system is online.

21.12 getVoucherProfile Command

21.12.1 Command Description

This command is used by a host to request the current voucher profile from an EGM. The voucher profile contains the protocol-related configuration option selections for the `voucher` device. A `voucherProfile` command is generated in response to the `getVoucherProfile` command.

21.12.2 Attribute and Element Detail



The `getVoucherProfile` command has no attributes or sub-elements.

21.13 voucherProfile Command

21.13.1 Command Description

This command is used by an EGM to report the current profile of a `voucher` device. The voucher profile contains the protocol-related configuration option selections for the `voucher` device. The configuration options can be set locally at the EGM or remotely via a configuration server using commands within the `optionConfig` class. The `voucherProfile` command is generated in response to a `getVoucherProfile` command.

Much of the static data to be printed on vouchers is provided as configuration options. The number of characters noted in the description of many of these attributes designates the maximum number of characters that can be printed on a standard G2S voucher in the area reserved for the attribute.

The `expireCashPromo` attribute specifies the number of days before a newly issued `cashable` or `promotional` voucher expires. The `expireNonCash` attribute specifies the default number of days before a newly issued `nonCashable` voucher expires. These values usually indicate when the voucher expires for use at an EGM. The system may have a separate, later expiration for redemption at the casino cage or cashier's booth. The `printExpCashPromo` attribute controls whether or not the EGM actually prints the expiration on `cashable` and `promotional` vouchers. The `printExpNonCash` attribute controls whether or not the EGM actually prints the expiration on `nonCashable` vouchers.

21.13.2 Attribute and Element Detail

Table 21.13 voucherProfile Attributes (Sheet 1 of 5)

Attribute		Restrictions	Description

`configurationId`	*	`type: t_configurationId`
		`use: optional`
		`default: 0`
			Last configuration identifier set by a G2S host; set to `0` (zero) when configuration changes are made by hosts other than G2S hosts.

`restartStatus`	*	`type: xs:boolean`
		`use: optional`
		`default: true`
			Extension in v2.1: g2sA

Table 21.13 voucherProfile Attributes (Sheet 2 of 5)

Attribute		Restrictions	Description
			Indicates the state of the `hostEnabled` attribute upon EGM restart. When `restartStatusMode` is set to `true`, a value of `true` indicates that `hostEnabled` must be set to its last known state; a value of `false` indicates that `hostEnabled` must be set to `false`. When `restartStatusMode` is set to `false`, a value of `true` indicates that `hostEnabled` must be set `true`; a value of `false` indicates that `hostEnabled` must be set to `false`.

`requiredForPlay`	*	`type: xs:boolean`
		`use: optional`
		`default: false`
			Indicates whether the EGM MUST be disabled if either `egmEnabled` or `hostEnabled` is set to `false`.

`minLogEntries`		`type: xs:int`
		`use: optional`
		`default: 35`
		`minIncl: 1`
			Indicates the minimum number of log entries that the EGM MUST maintain in persistent memory.

`timeToLive`	*	`type: t_timeToLive`
		`use: optional`
		`default: 30000`
			Time-to-live value for requests originated by the device.

`idReaderId`	*	`type: t_deviceId`
		`use: optional`
		`minIncl: 0`
		`default: 0`
			Device identifier of the `idReader` device associated with the `voucher` device; set to `0` (zero) if there is no associated `idReader` device. Wildcards not permitted.

`combineCashableOut`	*	`type: xs:boolean`
		`use: optional`
		`default: false`
			Indicates whether promotional credits MUST be converted to cashable credits when issuing vouchers.

`allowNonCashOut`	*	`type: xs:boolean`
		`use: optional`
		`default: true`
			Indicates whether the EGM is allowed to issue non-cashable vouchers for cash-outs when vouchers are enabled.

`maxValIds`	*	`type: xs:int`
		`use: optional`
		`default: 20`
		`minIncl: 1`
			Maximum number of validation identifiers the EGM may buffer.

`minLevelValIds`	*	`type: xs:int`
		`use: optional`
		`default: 15`
		`minIncl: 0`
			Minimum level (number) of validation identifiers that the EGM should have at all times. Whenever the current number of Validation identifiers reaches this number the EGM must request additional identifiers to reach the number defined by the `maxValIds` field.

Table 21.13 voucherProfile Attributes (Sheet 3 of 5)

Attribute		Restrictions	Description
valIdListRefresh	*	type: t_milliseconds use: optional default: 43200000	A timer is reset to this value each time EGM receives a validationData command. The default equates to 12 hours. If the timer expires, EGM MUST generate the getValidationData command.
valIdListLife	*	type: t_milliseconds use: optional default: 86400000	A timer is reset to this value each time the EGM receives a validationData command. The default equates to 24 hours. Validation identifiers MUST NOT be used while timer is expired.
voucherHoldTime	*	type: t_milliseconds use: optional default: 15000	Sets the maximum amount of time an EGM should wait for a response to a redeemVoucher command before rejecting a voucher.
printOffLine	*	type: xs:boolean use: optional default: true	Indicates whether vouchers can be issued while the validation system is offline (see Section 21.3.2, Validation System Offline, for more details); true indicates that vouchers can be issued; false indicates that vouchers cannot be issued.
expireCashPromo	*	type: xs:int use: optional default: 30 minIncl: 0	Number of days before cashable and promotional vouchers expire.
printExpCashPromo	*	type: xs:boolean use: optional default: true	Indicates whether expirations are printed on cashable and promotional vouchers.
expireNonCash	*	type: xs:int use: optional default: 30 minIncl: 0	Default number of days before non-cashable vouchers expire.
printExpNonCash	*	type: xs:boolean use: optional default: true	Indicates whether expirations are printed on non-cashable vouchers.
propName	*	type: t_voucherTitle40 use: required	Name of the property.
propLine1	*	type: t_voucherTitle40 use: required	First address line of the property.
propLine2	*	type: t_voucherTitle40 use: required	Second address line of the property.
titleCash	*	type: t_voucherTitle16 use: required	Title printed on vouchers for cashable credits.

Table 21.13 voucherProfile Attributes (Sheet 4 of 5)

Attribute		Restrictions	Description
titlePromo	*	type: t_voucherTitle16 use: optional default: <empty>	Title printed on vouchers for promotional credits. If not specified, use the value of the titleCash attribute.
titleNonCash	*	type: t_voucherTitle16 use: required	Title printed on vouchers for non-cashable credits.
titleLargeWin	*	type: t_voucherTitle16 use: required	Title printed on vouchers for wins greater than cabinetProfile.largeWinLimit.
titleBonusCash	*	type: t_voucherTitle16 use: required	Title printed on vouchers printed as a result of external bonus awards of cashable amounts.
titleBonusPromo	*	type: t_voucherTitle16 use: optional default: <empty>	Title printed on vouchers printed as a result of external bonus awards of promotional amounts. If not specified, use the value in the titleBonusCash attribute.
titleBonusNonCash	*	type: t_voucherTitle16 use: required	Title printed on vouchers printed as a result of external bonus awards of non-cashable amounts.
titleWatCash	*	type: t_voucherTitle16 use: required	Title printed on vouchers printed as a result of WAT transfers of cashable amounts.
titleWatPromo	*	type: t_voucherTitle16 use: optional default: <empty>	Title printed on vouchers printed as a result of WAT transfers of promotional amounts. If not specified, use the value in the titleWatCash attribute.
titleWatNonCash	*	type: t_voucherTitle16 use: required	Title printed on vouchers printed as a result of WAT transfers of non-cashable amounts.
Extension in v2.0.0: g2s1			
allowVoucherIssue	*	type: xs:boolean use: optional default: true	Indicates whether the voucher device supports validation data functionality; does not (directly) control voucher issuance.
allowVoucherRedeem	*	type: xs:boolean use: optional default: true	Indicates whether the voucher device supports voucher redemption functions.
maxOnLinePayOut	*	type: t_meterValue use: optional default: 0	Maximum value that can be paid using a voucher while communications to the host, which owns the voucher device, are active. The value 0 (zero) indicates that there is no limit.

Table 21.13 voucherProfile Attributes (Sheet 5 of 5)

Attribute	Restrictions	Description
maxOffLinePayOut *	type: t_meterValue use: optional default: 0	Maximum value that can be paid using a voucher while the validation system is offline (see Section 21.3.2, Validation System Offline, for more details). The value 0 (zero) indicates that there is no limit.
printNonCashOffLine *	type: xs:boolean use: optional default: true	Indicates whether non-cashable vouchers can be issued while the validation system is offline (see Section 21.3.2, Validation System Offline, for more details). To issue non-cashable vouchers while the validation system is offline, both printOffLine and allowNonCashOut MUST also be set to true. Note that manual authentication identifiers are only printed on non-cashable vouchers when printNonCashOffLine is set to true.
Extension in v3.0: g2s3		
usePlayerIdReader *	type: xs:boolean use: optional default: false	Indicates which idReader device to associate with the voucher device. When set to true, the EGM MUST use the idReader device associated with the currently active player session; otherwise, the EGM MUST use the idReader device specified in the idReaderId attribute of the voucherProfile command.
Extension in v3.0: g2sVSO		
noAckTimer *	type: t_milliseconds use: optional default: 15000	Indicates the maximum time between when a voucher is issued (when the voucher is printed) and when the issueVoucherAck command for the voucher is received before the validation system is declared offline.
Extension in v3.0: g2sVSO1		
enableHandpayVoucher *	type: xs:boolean use: optional default: false	Indicates whether Offline Handpay Vouchers are enabled.
titleHandpayVoucher *	type: t_voucherTitle16 use: optional default: "HANDPAY VOUCHER"	Title printed on Offline Handpay Vouchers.

* Standard configuration option that MUST be included in the standard option configuration group – G2S_voucherOptions – for devices within the voucher class.

21.14 getValidationData Command

21.14.1 Command Description

This command is used by an EGM to request new validation identifiers and seeds from a host. All issued vouchers MUST be acknowledged and `hostEnabled` MUST be set to `true` before the `getValidationData` command may be issued by the EGM. The `getValidationData` command MAY be issued regardless of the state of the `egmEnabled` attribute. For example, if the `egmEnabled` attribute has been set to `false` because the EGM has run out of validation IDs or the validation IDs have expired, provided that `hostEnabled` has not been set to `false`, the EGM MAY continue to request new validation data from the host. The EGM will include the most recent `validationListId` received from the host, if any. In this way, the host can verify whether the EGM currently has valid data. The `numValidationIds` attribute indicates how many validation identifiers the EGM needs to refill its buffer (to `maxValIds`). The `validationData` command is generated in response to the `getValidationData` command.

If all issued vouchers have been acknowledged and `hostEnabled` is set to `true`, the EGM MUST generate the `getValidationData` command under the following circumstances:

- When the EGM needs to replenish its validation identifier list based on `minLevelValIds`.
- When the `valIdListRefresh` timer or the `valIdListLife` timer is expired.

Once the EGM has determined that the validation data list needs to be refreshed, the EGM MUST continue to retry the `getValidationData` command at the frequency set in the `timeToLive` attribute of the `voucherProfile` command until acknowledged by a `validationData` command. See Section 1.22.4, Command Retry, for more details.

If the `voucher` device is enabled, the `valIdListLife` timer is not expired, and the EGM still has unused validation identifiers, the EGM may continue to issue vouchers while waiting for the `validationData` command. If the number of validation identifiers currently available on the EGM is higher than `minLevelValIds` then the `numValidationIds` attribute should be set to `0` (zero).This tells the host that the EGM does not currently need additional validation identifiers. The host may optionally send zero new identifiers to refresh the timers.

If the EGM does not currently have any validation identifiers or the `valIdListLife` timer has expired, the `egmEnabled` attribute MUST be set to `false`. Similarly, if the `printOffLine` attribute of the `voucherProfile` is set to `false` and the validation system is offline (see Section 21.3.2, Validation System Offline, for more details), the `egmEnabled` attribute MUST be set to `false`.

If the `valIdListLife` timer is expired at the time this command is issued, the `valIdListExpired` attribute MUST be set to `true`. This most likely indicates the EGM has been offline for an extended period of time or the `voucher` device has been disabled. The host may completely replace any validation data remaining in the EGM's buffer by sending the `validationData` command with the `deleteCurrent` attribute set to `true` and including validation identifiers and seeds up to the number specified in the `maxValIds` profile attribute. Alternatively, the host may simply refresh the `valIdListLife` timer by sending the `validationData` command with `deleteCurrent` set to `false` and including 0 (zero) or more validation identifiers and seeds up to the number requested in the `numValidationIds` attribute. In this case, the EGM MUST continue to use any remaining previous validation IDs before using the new IDs.

21.14.1.1 allowVoucherIssue Attribute

Extension in v2.0.0: g2s1

When the `allowVoucherIssue` profile attribute is set to `false` for a `voucher` device:

- The EGM MUST NOT place any `getValidationData` commands for the `voucher` device in an outbound queue.
- The EGM MUST NOT disable the `voucher` device because the device does not currently have any validation identifiers or the `valIdListLife` timer has expired.

21.14.2 Attribute and Element Detail

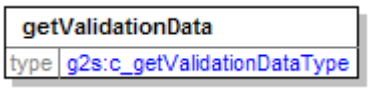


Table 21.14 getValidationData Attributes

Attribute	Restrictions	Description
`configurationId`	`type: t_configurationId` `use: required`	Configuration identifier set by the `optionConfig` host.
`validationListId`	`type: t_validationListId` `use: required`	The `validationListId` that was received in the most recent `validationData` command, or set to `0` (zero) if EGM has never received any validation identifiers.

`numValidationIds`	`type: xs:int`
	`use: optional`
	`default: 0`
	`minIncl: 0`
		Number of validation identifiers that the host should send to the EGM.

`valIdListExpired`	`type: xs:boolean`
	`use: optional`
	`default: false`
		Indicates whether the `valIdListLife` timer has expired.

21.15 validationData Command

21.15.1 Command Description

This command is sent from a host to provide new validation identifier and seed pairs to an EGM. The `validationData` command is generated in response to the `getValidationData` command.

The `validationListId` attribute is an arbitrary value provided by the host to track validation identifiers. The host can subsequently use this value to determine that an EGM has the validation identifiers that the host expects it to have.

The host may use the `deleteCurrent` attribute to instruct the EGM to delete any remaining validation identifiers and seeds before adding the new identifiers to its buffer. A host may want to use this if, for example, an EGM comes online with a `validationListId` the host does not recognize. If `deleteCurrent` is set to `false`, the EGM MUST add the new identifiers to its buffer, following any existing identifiers. Validation identifiers MUST be used in the order provided.

After new validation identifiers, if any, have been placed in the buffer and all outstanding `issueVoucher` commands have been acknowledged, the EGM MUST generate the `G2S_VCE102 Validation ID Data Updated` event.

If any of the new validation identifiers or seeds cannot be used, for example if a validation ID includes non-numeric data or the command includes more validation identifiers than the EGM can buffer, the EGM MUST NOT use any of the new identifiers. The EGM MUST NOT delete any existing validation identifiers, refresh its `valIdListLife` timer, or generate the `G2S_VCE102 Validation ID Data Updated` event. Instead, the EGM SHOULD generate the `G2S_APE001 At Least One Syntax/Semantic Command Error` event and include a text string describing the nature of the error.

21.15.2 Attribute and Element Detail

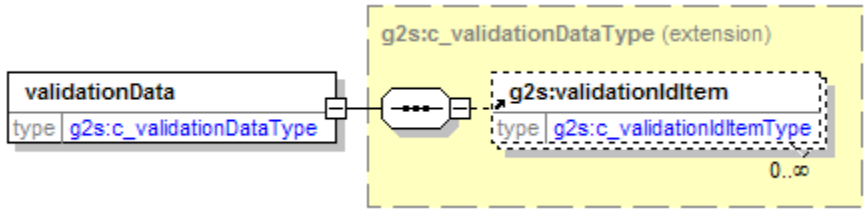


Table 21.15 validationData Attributes

Attribute	Restrictions	Description
`validationListId`	`type: t_validationListId` `use: required`	Identifier for the list of validation identifiers. Arbitrarily assigned by the host.

`deleteCurrent`	`type: xs:boolean`
	`use: optional`
	`default: false`
		Indicates whether EGM should delete all current validation identifiers before adding new identifiers.

Table 21.16 validationData Elements

Element	Restrictions	Description
validationIdItem	minOcc: 0 maxOcc: ∞	Contains a validation identifier. See attributes in Table 21.17.

Table 21.17 validationIdItem Attributes

Attribute	Restrictions	Description
validationId	type: t_validationId use: required	Validation identifier. This number is printed on the voucher and used to create the manual authentication identifier.
validationSeed	type: t_validationSeed use: required	Corresponding seed value used to create a manual authentication identifier.

21.16 issueVoucher Command

21.16.1 Command Description

This command is used by an EGM to notify a host that a voucher has been issued. The `issueVoucherAck` command is generated in response to the `issueVoucher` command.

The `issueVoucher` command and related events SHOULD be generated as soon as the EGM is irreversibly committed to the voucher issuance operation and the associated credits have been removed from the credit meter. The EGM SHOULD NOT wait until the final results of the print operation are known. Waiting for the final results of the print operation could cause significant delays in reporting that the voucher issuance operation had taken place. Presentation errors MAY be reported by setting the `egmException` attribute of the `issueVoucher` command to `1` (one). However, reporting any such errors SHOULD NOT delay the reporting of the voucher issuance operation.

EGM-originated print operations, including voucher issuance operations, are recorded in the `printer` class log. Host systems can monitor events within the `printer` class and inspect the `printer` class log to determine whether presentation errors are occurring.

The EGM MUST continue to retry the `issueVoucher` command until a valid `issueVoucherAck` command is received.

Before cashing out non-cashable credits to a voucher, the following rules MUST be applied:

- If the `allowNonCashOut` attribute of the `voucherProfile` is set to `true` and there is no expiration associated with the non-cashable credits, the EGM MAY produce a voucher for the credits. The default `expireNonCash` expiration MUST be used for the voucher. Note, this expiration applies only to the voucher, not to the non-cashable credits themselves.
- If the `allowNonCashOut` attribute of the `voucherProfile` is set to `true`, there is an expiration associated with the non-cashable credits, and the current date and time is the same as or prior to that expiration, the EGM MAY produce a voucher for the credits.
- If the `allowNonCashOut` attribute of the `voucherProfile` is set to `true`, there is an expiration associated with the non-cashable credits, and the current date and time is after that expiration, the EGM MUST NOT produce a voucher for the credits.
- If the `allowNonCashOut` attribute of the `voucherProfile` is set to `false`, the EGM MUST NOT produce a voucher for non-cashable credits.

It is important to note that `promotional` credits are fully cashable by the player. The only difference is in the accounting. While it is advantageous to the casino to maintain the promotional status of cashable `promotional` credits when cashing out to a voucher, it is usually undesirable for the player to receive two cashable vouchers for one cash-out. When cash-out vouchers are generated, the `combineCashableOut` configuration controls whether the EGM MUST convert promotional credits to cashable when printing a voucher. If `promotional` credits are converted to `cashable` the promotional status of the `promotional` credits is lost. If the EGM has `nonCashable` credits, the EGM MUST always generate a separate voucher for this credit type. If `combineCashableOut` is set to `false`, the EGM may therefore be required to generate up to three vouchers to fully cash out all credits on an EGM.

With the G2S protocol, vouchers may be issued in response to player cash-out requests or due to machine limitations such as a win that does not fit in the credit meter. Under certain circumstances, vouchers may also be issued for large wins that cannot be paid directly by the EGM because of regulatory requirements, such as W-2Gs in the USA (i.e. wins greater than `cabinetProfile.largeWinLimit`). If a voucher is issued as a result of a large win handpay being keyed off to a voucher, the `largeWin` attribute MUST be set to `true` by the EGM

when the voucher is issued. Subsequently, when the player attempts to redeem the voucher, based on jurisdictional requirements, the host will determine whether the voucher can be redeemed at an EGM. Note that "large win" is not related to whether the win initially resulted in a handpay. It is based solely on whether the win resulted in a handpay because of the `cabinetProfile.largeWinLimit`.

For vouchers other than cash-outs from the credit meter, references to the direct source transactions MUST be included in the `voucherSourceRef` sub-elements of the `issueVoucher` element. For example, this includes vouchers produced as a result of handpay, game play, progressive, bonus, and WAT transactions, as well as change vouchers from `redeemVoucher` transactions. Note that, for example, if a win results in a handpay which is subsequently keyed off to a voucher, only the handpay is considered a source transaction, not the game play. The handpay log entry will provide the linkage between the game play and the voucher.

If the EGM has access to player identification information, it MUST be sent to the host to identify the active player at the time the voucher was issued.

The host must make a best-effort to acknowledge `issueVoucher` commands. Class-specific application-level error codes MUST NOT be used. Until acknowledged, the EGM MUST continue to retry the commands at the frequency set in the `timeToLive` attribute of the `voucherProfile` command. See Section 1.22.4, Command Retry, for more details. Failure to acknowledge the commands may cause the log to fill up, the `voucher` device to be disabled, and a loss of functionality. See Section 1.18.4, Committed Transactions, for more details.

21.16.1.1 Duplicate Commands

The host MUST consider an `issueVoucher` command logically equivalent to a previous `issueVoucher` command if the host detects that the `transactionId` associated with the voucher issuance was reported in a previous `issueVoucher` command for the same EGM since the last time that non-volatile storage was cleared on the EGM. In such cases, the host MUST generate a logically equivalent `issueVoucherAck` command in response to the `issueVoucher` command.

21.16.2 Attribute and Element Detail

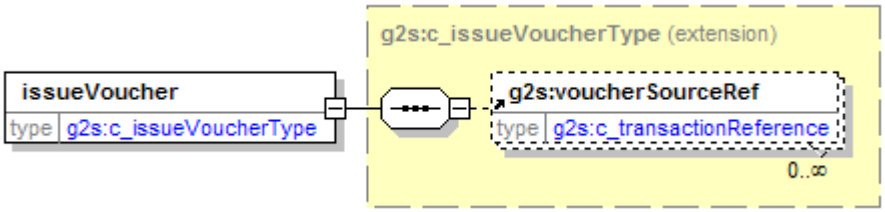


Table 21.18 issueVoucher Attributes (Sheet 1 of 3)

Attribute	Restrictions	Description

`transactionId`	`type: t_transactionId`
	`use: required`
	`minIncl: 1`
		Voucher transaction identifier.

`idReaderType`	`type: t_idReaderTypes`
	`use: optional`
	`default: G2S_none`
		The `idReaderType` of the `idReader` device associated with the `voucher` device. If no `idReader` device is associated or the device is disabled then set to `G2S_none`.

Table 21.18 issueVoucher Attributes (Sheet 2 of 3)

Attribute	Restrictions	Description
idNumber	type: t_idNumber use: optional default: <empty>	The idNumber present in the idReader device associated with the voucher device. If no idReader device is associated or the device is disabled then set to <empty>.
playerId	type: t_playerId use: optional default: <empty>	The playerId present in the idReader device associated with the voucher device. If no idReader device is associated or the device is disabled then set to <empty>.
validationId	type: t_validationId use: required	Voucher identifier.
voucherAmt	type: t_meterValue use: required	Voucher amount.
creditType	type: t_creditTypes use: required	Credit type: G2S_cashable, G2S_promo, or G2S_nonCash.
voucherSource	type: t_voucherSources use: optional default: G2S_egmIssued	Indicates voucher was issued by the EGM.
largeWin	type: xs:boolean use: optional default: false	Indicates whether the voucher was issued as the result of a large win.
voucherSequence	type: xs:int use: optional default: 0 minIncl: 0	Internal EGM voucher issuance sequence number printed on the voucher. Number must start at 1 (one) and roll over to 1 (one) when it reaches 9999.
expireCredits	type: xs:boolean use: optional default: false	Indicates whether voucher was issued using date/time expiration provided when non-cashable credits were transferred to the EGM (true), or default [n] days expiration (false).
expireDateTime	type: t_g2sDateTime use: optional default: 2000-01-01T00:00:00.000-00:00	Expiration of voucher in date/time (only valid if expireCredits is set to true).
transferAmt	type: t_meterValue use: optional default: 0	Actual amount transferred. This MUST be equal to the voucherAmt attribute.
transferDateTime	type: t_g2sDateTime use: required	Date/time that the voucher was issued; SHOULD be the same as the date/time printed on the voucher and the same as the date/time displayed by the EGM in its operator logs.
expireDays	type: xs:int use: optional default: -1 minIncl: -1	Expiration of voucher in number of days (only valid if expireCredits is set to false). -1 indicates no expiration is provided.

Table 21.18 issueVoucher Attributes (Sheet 3 of 3)

Attribute	Restrictions	Description
egmAction	type: t_egmVoucherActions use: required	Type of action taken: G2S_issued.
egmException	type: t_egmVoucherExceptions use: optional default: 0	Indicates voucher issued or partial voucher issued (presentation error).

Table 21.19 issueVoucher Elements

Element	Restrictions	Description
voucherSourceRef	minOcc: 0 maxOcc: ∞	Contains information about any associated transaction. See attributes in Table 21.20.

Table 21.20 voucherSourceRef Attributes

Attribute	Restrictions	Description
deviceClass	type: t_deviceClass use: required	Device class of the associated transaction. Wildcards not permitted.
deviceId	type: t_deviceId use: required	Device identifier of the associated transaction. Wildcards not permitted.
transactionId	type: t_transactionId use: required minIncl: 1	Transaction identifier of the associated transaction.
logSequence	type: t_logSequence use: required	Log sequence of the associated transaction.
cashableAmt	type: t_meterValue use: optional default: 0	Cashable amount of the associated transaction.
promoAmt	type: t_meterValue use: optional default: 0	Promotional amount of the associated transaction.
nonCashAmt	type: t_meterValue use: optional default: 0	Non-cashable amount of the associated transaction.

21.17 issueVoucherAck Command

21.17.1 Command Description

This command is used by a host to acknowledge the receipt of an `issueVoucher` command from an EGM (see Section 21.3.2, Validation System Offline, for more details).

21.17.1.1 Duplicate Commands

The EGM MUST consider an `issueVoucherAck` command logically equivalent to a previous `issueVoucherAck` command if the EGM detects from its `voucher` class log that the voucher issuance associated with the `transactionId` has already been acknowledged — that is, the state of the voucher is no longer `G2S_issueSent`. In such cases, the EGM MUST NOT generate any additional `G2S_VCE105 Voucher Issue Command Acknowledged` events.

21.17.2 Attribute and Element Detail

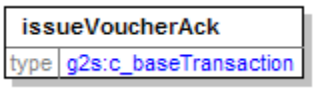


Table 21.21 issueVoucherAck Attributes

Attribute	Restrictions	Description

`transactionId`	`type: t_transactionId`
	`use: required`
	`minIncl: 1`
		Voucher transaction identifier.

21.18 redeemVoucher Command

21.18.1 Command Description

This command is used by an EGM to send a voucher redemption request to a host. The `redeemVoucher` command identifies a voucher that has been inserted in the EGM's note acceptor. The `authorizeVoucher` command is generated in response to the `redeemVoucher` command to indicate whether the voucher is valid.

If the redemption is not authorized within the time specified in the `voucherHoldTime` attribute, the EGM MUST reject the voucher and generate a `commitVoucher` command indicating that the voucher was rejected due to a timeout (`egmException = "5"`). If an `authorizeVoucher` command is received subsequently, or at any time a voucher is not being held in escrow, the EGM should ignore the command. While waiting for the `voucherHoldTime` to expire, the EGM MUST continue to retry the `redeemVoucher` command at the frequency set in the `timeToLive` attribute of the `voucherProfile` command. See Section 1.22.4, Command Retry, for more details.

After issuing a `redeemVoucher` command, after the voucher is stacked or rejected, the EGM MUST always generate a `commitVoucher` command to report the final disposition of the voucher redemption request. Even if the EGM does not receive an `authorizeVoucher` command or receives an error in response to the `redeemVoucher` command, the EGM MUST still generate a `commitVoucher` command for the host to confirm the outcome of the redemption request.

If the EGM has access to player identification information, it MUST send this information to the host to allow the host to apply any card restrictions to the voucher redemption.

21.18.1.1 Duplicate Commands

The host MUST consider a `redeemVoucher` command logically equivalent to a previous `redeemVoucher` command if the host detects that the `transactionId` associated with the voucher redemption request was reported in a previous `redeemVoucher` or `commitVoucher` command for the same EGM since the last time that non-volatile storage was cleared on the EGM. In such cases, the host MUST generate a logically equivalent `authorizeVoucher` command in response to the `redeemVoucher` command.

21.18.1.2 allowVoucherRedeem Attribute

Extension in v2.0.0: g2s1

When the `allowVoucherRedeem` profile attribute is set to `false` for a `voucher` device the EGM MUST NOT generate the `redeemVoucher` command for the `voucher` device.

21.18.2 Attribute and Element Detail

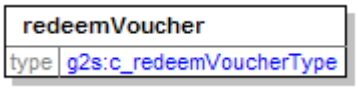


Table 21.22 redeemVoucher Attributes (Sheet 1 of 2)

Attribute	Restrictions	Description

`transactionId`	`type: t_transactionId`
	`use: required`
	`minIncl: 1`
		Voucher transaction identifier.

Table 21.22 redeemVoucher Attributes (Sheet 2 of 2)

Attribute	Restrictions	Description
idReaderType	type: t_idReaderTypes use: optional default: G2S_none	The idReaderType of the idReader device associated with the voucher device. If no idReader device is associated or the device is disabled then set to G2S_none.
idNumber	type: t_idNumber use: optional default: <empty>	The idNumber present in the idReader device associated with the voucher device. If no idReader device is associated or the device is disabled then set to <empty>.
playerId	type: t_playerId use: optional default: <empty>	The playerId present in the idReader device associated with the voucher device. If no idReader device is associated or the device is disabled then set to <empty>.
validationId	type: t_validationId use: required	Voucher identifier.

21.19 authorizeVoucher Command

21.19.1 Command Description

This command is used by the host to authorize or deny the redemption of a voucher. The `authorizeVoucher` command contains the information the EGM requires to perform the redemption. The `authorizeVoucher` command is generated in response to a `redeemVoucher` command.

To authorize the redemption of a voucher, the host MUST set the `voucherAmt` to a non-zero value and MUST set the `hostException` attribute to `0` (zero). The host can set the `voucherSource` attribute to `G2S_systemIssued` to indicate that the voucher was issued by the system as part of a promotion, or `G2S_egmIssued` to indicate the voucher was printed by another EGM, or a kiosk or similar process. This can be used in jurisdictions where the voucher expense for EGM-issued vouchers is taken at the time of redemption. System-issued vouchers have no corresponding deduction and therefore MUST be metered separately from EGM-issued vouchers. In jurisdictions where this is not an issue, all vouchers can be redeemed and metered as EGM-issued vouchers.

For non-cashable vouchers, the host may optionally specify an expiration to be assigned to those specific non-cashable credits by specifying a date/time in the `expireDateTime` attribute and setting the `expireCredits` attribute to `true`. If the `expireCredits` attribute is `false`, the credits do not have any specific expiration associated with them. See Section 3.18, cabinetProfile Command for more details regarding acceptance of non-cashable credits.

If the host determines that the voucher is not valid for redemption on the EGM, the `voucherAmt` MUST be set to `0` (zero) and the `hostException` attribute MUST be set to a non-zero value, indicating the reason for rejection. The `creditType`, `voucherSource`, `largeWin`, `voucherSequence`, `expireCredits`, and `expireDateTime` attributes are not used and may be set to any syntactically correct value.

The host may use the `hostAction` attribute to force an EGM to stack a voucher that is not valid, or force the EGM to reject the voucher following a valid redemption. If the host authorizes redemption and the EGM is unable to redeem the voucher for any reason, the EGM MUST reject the voucher regardless of the `hostAction` value. The host should use this attribute with extreme caution. The `hostAction` attribute may be set to one of three values:

- `G2S_egmAction` tells the EGM to perform its normal action of stacking or rejecting a voucher; for example, stacking a redeemed voucher and rejecting all others.
- `G2S_stack` tells the EGM to stack a voucher following successful completion of the `authorizeVoucher` command as directed by the `hostException` attribute. If the host authorizes redemption of the voucher, the EGM MUST NOT stack the voucher if it is unable to redeem the voucher for any reason, and it MUST NOT redeem the voucher if it is unable to stack the voucher for any reason. If the host does not authorize redemption (i.e. `hostException` is set to a non-zero value), the EGM MUST stack the voucher if possible.
- `G2S_reject` tells the EGM to reject the voucher regardless of whether it was successfully redeemed or not. If `hostAction` is set to `G2S_reject`, the EGM MUST NOT stack the voucher under any circumstances.

The `egmAction` attribute indicates the final disposition of the voucher — that is, whether the voucher was stacked or rejected. If the voucher was stacked by the EGM, the `egmAction` attribute MUST be set to `G2S_redeemed`. Otherwise, if the voucher was rejected (not stacked) by the EGM, the `egmAction` attribute MUST be set to `G2S_rejected`. The `egmAction` attribute MUST be set based on the actual action performed by the EGM, not the action requested by the host in the `hostAction` attribute.

When the EGM has transferred the credits to the credit meter or the voucher is rejected, the EGM MUST generate a `commitVoucher` command for the host. In all cases, following the receipt of an `authorizeVoucher` command, the EGM MUST generate a `commitVoucher` command to report the results of the transfer, even if no funds were transferred.

When an `authorizeVoucher` command is generated, the host MUST record that a redemption is pending for the voucher. Until a `commitVoucher` command is received or the status is manually reset, additional redemptions MUST NOT be permitted for that voucher.

21.19.1.1 Duplicate Commands

The EGM MUST consider an `authorizeVoucher` command logically equivalent to a previous `authorizeVoucher` command if the EGM detects from its `voucher` class log that the redemption of the voucher associated with the `transactionId` has already been authorized or denied — that is, the state of the voucher redemption request is no longer `G2S_redeemSent`. In such cases, the EGM MUST NOT generate any additional `G2S_VCE107 Voucher Authorized` events.

21.19.2 Attribute and Element Detail

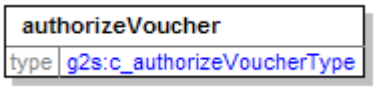


Table 21.23 authorizeVoucher Attributes (Sheet 1 of 2)

Attribute	Restrictions	Description

`transactionId`	`type: t_transactionId`
	`use: required`
	`minIncl: 1`
		Voucher transaction identifier.

`validationId`	`type: t_validationId` `use: required`	Voucher identifier.
`voucherAmt`	`type: t_meterValue` `use: required`	Amount of voucher.
`creditType`	`type: t_creditTypes` `use: required`	Credit type: `G2S_cashable`, `G2S_promo`, or `G2S_nonCash`.

`voucherSource`	`type: t_voucherSources`
	`use: optional`
	`default: G2S_egmIssued`
		Indicates whether the voucher was issued by the system or an EGM.

`largeWin`	`type: xs:boolean`
	`use: optional`
	`default: false`
		Indicates whether the voucher was originally issued as the result of a large win.

`voucherSequence`	`type: xs:int`
	`use: optional`
	`default: 0`
	`minIncl: 0`
		Voucher sequence number provided by the host: MAY be the same value reported when the voucher was issued; MAY be another value. The EGM does not validate this number, but only records the value in its log.

`expireCredits`	`type: xs:boolean`
	`use: optional`
	`default: false`
		Indicates whether non-cashable credits have a date/time expiration assigned to them.

Table 21.23 authorizeVoucher Attributes (Sheet 2 of 2)

Attribute	Restrictions	Description
expireDateTime	type: t_g2sDateTime use: optional default: 2000-01-01T00:00:00.000-00:00	Expiration assigned to non-cashable credits (only valid if expireCredits is set to true).
hostAction	type: t_hostVoucherActions use: optional default: G2S_egmAction	Host designated stacker action: G2S_egmAction, G2S_stack, or G2S_reject.
hostException	type: t_hostVoucherExceptions use: optional default: 0	Host transfer exception code.

21.20 commitVoucher Command

21.20.1 Command Description

This command is used by an EGM to report the results of a voucher redemption previously initiated with a `redeemVoucher` command.

The `commitVoucher` command is generated regardless of whether or not the redemption was successful. If unsuccessful, the `transferAmt` attribute MUST be set to `0` (zero). If successful, the `transferAmt` attribute MUST be set to the actual amount transferred, which MUST be the total amount of the voucher. If the voucher is rejected, the `egmException` attribute MUST indicate the reason for rejection. If the voucher is rejected, the host should reset the status of the voucher so that it can be redeemed elsewhere.

The EGM MUST continue to retry the `commitVoucher` command until a valid `commitVoucherAck` command is received.

The host must make a best-effort to acknowledge `commitVoucher` commands. Class-specific application-level error codes MUST NOT be used. Until acknowledged, the EGM MUST continue to retry the commands at the frequency set in the `timeToLive` attribute of the `voucherProfile` command. See Section 1.22.4, Command Retry, for more details. Failure to acknowledge the commands may cause the log to fill up, the `voucher` device to be disabled, and a loss of functionality. See Section 1.18.4, Committed Transactions, for more details.

21.20.1.1 Duplicate Commands

The host MUST consider a `commitVoucher` command logically equivalent to a previous `commitVoucher` command if the host detects that the `transactionId` associated with the redemption or rejection of the voucher was reported in a previous `commitVoucher` command for the same EGM since the last time that non-volatile storage was cleared on the EGM. In such cases, the host MUST generate a logically equivalent `commitVoucherAck` command in response to the `commitVoucher` command.

21.20.2 Attribute and Element Detail

Table 21.24 commitVoucher Attributes (Sheet 1 of 2)

Attribute	Restrictions	Description

`transactionId`	`type: t_transactionId`
	`use: required`
	`minIncl: 1`
		Voucher transaction identifier.

`validationId`	`type: t_validationId` `use: required`	Voucher identifier.
`voucherAmt`	`type: t_meterValue` `use: required`	Voucher amount.
`creditType`	`type: t_creditTypes` `use: required`	Credit type: `G2S_cashable`, `G2S_promo`, or `G2S_nonCash`.

Table 21.24 commitVoucher Attributes (Sheet 2 of 2)

Attribute	Restrictions	Description
voucherSource	type: t_voucherSources use: optional default: G2S_egmIssued	Indicates whether the voucher was issued by the system or an EGM.
largeWin	type: xs:boolean use: optional default: false	Indicates whether the voucher was originally issued as the result of a large win.
voucherSequence	type: xs:int use: optional default: 0 minIncl: 0	Voucher sequence number that was provided by the host in the authorizeVoucher command.
expireCredits	type: xs:boolean use: optional default: false	Indicates whether non-cashable credits have a date/time expiration assigned to them.
expireDateTime	type: t_g2sDateTime use: optional default: 2000-01-01T00:00:00.000-00:00	Expiration assigned to non-cashable credits (only valid if expireCredits is set to true).
transferAmt	type: t_meterValue use: optional default: 0	Actual amount transferred.
transferDateTime	type: t_g2sDateTime use: required	Date and time that the voucher was redeemed or rejected.
egmAction	type: t_egmVoucherActions use: required	Type of action taken: G2S_redeemed or G2S_rejected.
egmException	type: t_egmVoucherExceptions use: optional default: 0	Indicates voucher redeemed or reason for voucher rejection.

21.21 commitVoucherAck Command

21.21.1 Command Description

This command is used by a host to acknowledge the receipt of a `commitVoucher` command from an EGM.

21.21.1.1 Duplicate Commands

The EGM MUST consider a `commitVoucherAck` command logically equivalent to a previous `commitVoucherAck` command if the EGM detects from its `voucher` class log that the redemption or rejection of the voucher associated with the `transactionId` has already been acknowledged — that is, the state of the voucher redemption request is no longer `G2S_commitSent`. In such cases, the EGM MUST NOT generate any additional `G2S_VCE111 Voucher Commit Command Acknowledged` events.

21.21.2 Attribute and Element Detail

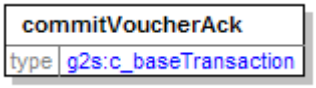


Table 21.25 commitVoucherAck Attributes

Attribute	Restrictions	Description

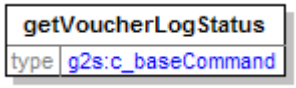
`transactionId`	`type: t_transactionId`
	`use: required`
	`minIncl: 1`
		Voucher transaction identifier.

21.22 getVoucherLogStatus Command

21.22.1 Command Description

This command is used by the host to request the current status of the voucher transaction log from an EGM. The response includes the sequence number of the last transaction and the total number of transactions in the log. A `voucherLogStatus` command is generated in response to a `getVoucherLogStatus` command.

21.22.2 Attribute and Element Detail



The `getVoucherLogStatus` command contains no attributes or sub-elements.

21.23 voucherLogStatus Command

21.23.1 Command Description

This command is used by the EGM to send the current status of the `voucher` transaction log to a host. The `voucherLogStatus` command is generated in response to the `getVoucherLogStatus` command.

21.23.2 Attribute and Element Detail

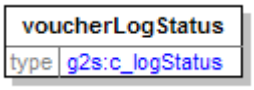


Table 21.26 voucherLogStatus Attributes

Attribute	Restrictions	Description

`lastSequence`	`type: t_logSequence`
	`use: optional`
	`default: 0`
	`minIncl: 0`
		The sequence number of the last transaction within the log.

`totalEntries`	`type: xs:int`
	`use: optional`
	`default: 0`
	`minIncl: 0`
		The total number of transactions within the log.

21.24 getVoucherLog Command

21.24.1 Command Description

This command is used by the host to request the `voucher` transaction log from an EGM. Additional information regarding the use of the `lastSequence` and `totalEntries` attributes can be found in Chapter 1. The `voucherLogList` command is generated in response to the `getVoucherLog` command.

21.24.2 Attribute and Element Detail

Table 21.27 getVoucherLog Attributes

Attribute	Restrictions	Description

`lastSequence`	`type: t_logSequence`
	`use: optional`
	`default: 0`
	`minIncl: 0`
		The sequence number of the transaction that should be the first entry in the list; if set to `0` (zero) then default to the last transaction.

`totalEntries`	`type: xs:int`
	`use: optional`
	`default: 0`
	`minIncl: 0`
		The total number of transactions that should be included in the list; if set to `0` (zero) then default to all transactions.

21.25 voucherLogList Command

21.25.1 Command Description

This command is used by the EGM to send the contents of the transaction log to a host. The `voucherLogList` command is generated in response to a `getVoucherLog` command.

For security reasons, because guest hosts are able to interrogate the `voucher` transaction log, only the rightmost 4 (four) digits of the validation ID are reported — the leftmost 14 (fourteen) digits MUST be masked and replaced with non-numeric values. For example, the validation ID "`12345678901234 5678`" MUST be masked and a value such as "`xxxxxxxxxxxxxx5678`" MUST be reported. The entire `validationId` should be logged for usage during command retry.

21.25.2 Attribute and Element Detail

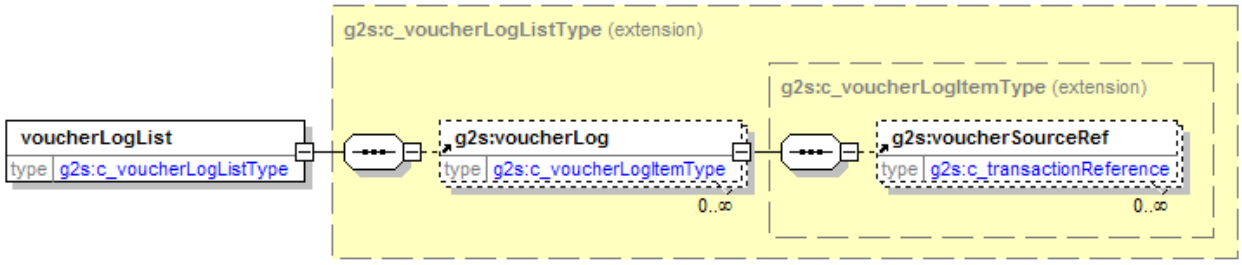


Table 21.28 voucherLogList Elements

Element	Restrictions	Description
`voucherLog`	`minOcc: 0` `maxOcc: ∞`	Contains information about a specific voucher transaction.

Table 21.29 voucherLog Attributes (Sheet 1 of 3)

Attribute	Restrictions	Description
Record Identification Attributes		
`logSequence`	`type: t_logSequence` `use: required`	Unique log sequence number assigned by the EGM; a series that strictly increases by 1 (one) starting at 1 (one).
`deviceId`	`type: t_deviceId` `use: required`	Device identifier of the device that generated the transaction. Wildcards not permitted.

`transactionId`	`type: t_transactionId`
	`use: required`
	`minIncl: 1`
		Voucher transaction identifier.

Voucher Transaction Status		
`voucherState`	`type: t_voucherStates` `use: required`	Current state of the voucher transaction.

Table 21.29 voucherLog Attributes (Sheet 2 of 3)

Attribute	Restrictions	Description
voucherAction	type: t_voucherActions use: required	Type of log entry: G2S_issue or G2S_redeem.
Player Identification Attributes		
idReaderType	type: t_idReaderTypes use: optional default: G2S_none	The idReaderType of the idReader device associated with the voucher device. If no idReader device is associated or the device is disabled then set to G2S_none.
idNumber	type: t_idNumber use: optional default: <empty>	The idNumber present in the idReader device associated with the voucher device. If no idReader device is associated or the device is disabled then set to <empty>.
playerId	type: t_playerId use: optional default: <empty>	The playerId present in the idReader device associated with the voucher device. If no idReader device is associated or the device is disabled then set to <empty>.
Voucher Attributes		
validationId	type: t_validationId use: required	Voucher identifier (only last four digits are reported to the host).
voucherAmt	type: t_meterValue use: required	Voucher amount.
creditType	type: t_creditTypes use: required	Credit type (G2S_cashable, G2S_promo, or G2S_nonCash).
voucherSource	type: t_voucherSources use: optional default: G2S_egmIssued	Indicates whether the voucher was issued by the system or an EGM.
largeWin	type: xs:boolean use: optional default: false	Indicates whether the voucher was issued as the result of a large win.
voucherSequence	type: xs:int use: optional default: 0 minIncl: 0	When the voucherAction is G2S_issue, the internal EGM voucher issuance sequence number printed on the voucher. When the voucherAction is G2S_redeem, the voucher sequence number that was provided by the host in the authorizeVoucher command.
expireCredits	type: xs:boolean use: optional default: false	Indicates whether non-cashable credits have a date/time expiration assigned to them.
expireDateTime	type: t_g2sDateTime use: optional default: 2000-01-01T00:00:00.000-00:00	Expiration assigned to non-cashable credits (only valid if expireCredits is set to true).

Table 21.29 voucherLog Attributes (Sheet 3 of 3)

Attribute	Restrictions	Description
hostAction	type: t_hostVoucherActions use: optional default: G2S_egmAction	May be used by host to force a voucher to be stacked or rejected regardless of amount paid to the player.
hostException	type: t_hostVoucherExceptions use: optional default: 0	Host transfer exception code.
Transfer Attributes		
transferAmt	type: t_meterValue use: optional default: 0	Actual amount transferred to the EGM.
transferDateTime	type: t_g2sDateTime use: required	Date and time that the voucher was issued, redeemed, or rejected.
expireDays	type: xs:int use: optional default: -1 minIncl: -1	Expiration of voucher in number of days (only valid if expireCredits is set to false). -1 indicates no expiration is provided.
egmAction	type: t_egmVoucherActions use: required	Type of action taken.
egmException	type: t_egmVoucherExceptions use: optional default: 0	EGM transfer exception code.

Table 21.30 voucherLog Elements

Element	Restrictions	Description
voucherSourceRef	minOcc: 0 maxOcc: ∞	Contains information about an associated issueVoucher transaction. See attributes in Table 21.31.

Table 21.31 voucherSourceRef Attributes (Sheet 1 of 2)

Attribute	Restrictions	Description
deviceClass	type: t_deviceClass use: required	Device class of the associated transaction. Wildcards not permitted.
deviceId	type: t_deviceId use: required	Device identifier of the associated transaction. Wildcards not permitted.
transactionId	type: t_transactionId use: required minIncl: 1	Transaction identifier of the associated transaction.
logSequence	type: t_logSequence use: required	Log sequence of the associated transaction.

Table 21.31 voucherSourceRef Attributes (Sheet 2 of 2)

Attribute	Restrictions	Description
cashableAmt	type: t_meterValue use: optional default: 0	Cashable amount of the associated transaction.
promoAmt	type: t_meterValue use: optional default: 0	Promotional amount of the associated transaction.
nonCashAmt	type: t_meterValue use: optional default: 0	Non-cashable amount of the associated transaction.

21.26 Data Types

The following table lists the data types specific to the `voucher` class:

Table 21.32 voucher Class Data Types (Sheet 1 of 2)

Data Type	Restrictions	Description
t_voucherActions	type: xs:string enumerations: G2S_issue G2S_redeem	Type of log entry. See Section 21.26.1 for description of enumerations.
t_hostVoucherActions	type: xs:string enumerations: G2S_egmAction G2S_stack G2S_reject	Host designated stacker action. See Section 21.26.2 for description of enumerations.
t_egmVoucherActions	type: xs:string enumerations: G2S_issued G2S_pending G2S_redeemed G2S_rejected	Type of action taken by the EGM. See Section 21.26.3 for description of enumerations.
t_hostVoucherExceptions	type: t_exceptionCode enumerations: See Section 21.26.4.	Host transfer exception code.
t_egmVoucherExceptions	type: t_exceptionCode enumerations: See Section 21.26.5.	EGM transfer exception code.
t_validationListId	type: xs:long minIncl: 0	Host-assigned validation list identifier.
t_voucherSources	type: xs:string enumerations: G2S_egmIssued G2S_systemIssued	Source of voucher. See Section 21.26.6 for description of enumerations.
t_voucherStates	type: xs:string enumerations: G2S_issueSent G2S_issueAcked G2S_redeemSent G2S_redeemAuth G2S_commitSent G2S_commitAcked	States of the voucher transaction. See Section 21.26.7 for description of enumerations.
t_validationSeed	type: xs:string maxLen: 20 pattern: [-~]{0,20}	Validation seed data type.
t_voucherTitle16	type: xs:string maxLen: 16	16-character data type for voucher titles.
t_voucherTitle40	type: xs:string maxLen: 40	40-character data type for voucher titles.

Table 21.32 voucher Class Data Types (Sheet 2 of 2)

Data Type	Restrictions	Description
t_validationId	type: xs:string minLen: 18 maxLen: 18	Voucher identifier.

21.26.1 t_voucherActions Enumerations: Voucher Action Codes

The following table contains the enumerations for the t_voucherActions data type:

Table 21.33 Enumerations for t_voucherActions Data Type

Action	Description
G2S_issue	Voucher issuance log.
G2S_redeem	Voucher redemption log.

21.26.2 t_hostVoucherActions Enumerations: Host Action Codes

The following table contains the enumerations for the t_hostVoucherActions data type:

Table 21.34 Enumerations for t_hostVoucherActions Data Type

Action	Description
G2S_egmAction	Stack or reject to be determined by the EGM.
G2S_stack	Force voucher to be stacked.
G2S_reject	Force voucher to be rejected.

21.26.3 t_egmVoucherActions Enumerations: EGM Action Codes

The following table contains the enumerations for the t_egmVoucherActions data type:

Table 21.35 Enumerations for t_egmVoucherActions Data Type

Action	Description
G2S_issued	Voucher issued.
G2S_pending	Redemption requested.
G2S_redeemed	Voucher stacked.
G2S_rejected	Voucher returned to player (not stacked).

21.26.4 t_hostVoucherExceptions Enumerations: Host Transfer Exception Codes

The following table contains the enumerations for the `t_hostVoucherExceptions` data type:

Table 21.36 Exception Codes for t_hostVoucherExceptions Data Type

Exception Code	Description
0	Redemption authorized.
1	Redemption in process at another location.
2	Voucher already redeemed.
3	Voucher expired.
4	Voucher not found.
5	Voucher cannot be redeemed at this location.
6	Incorrect player for voucher.
99	Redemption denied – no reason given.

21.26.5 t_egmVoucherExceptions Enumerations: EGM Transfer Exception Codes

The following table contains the enumerations for the `t_egmVoucherExceptions` data type:

Table 21.37 Exception Codes for t_egmVoucherExceptions Data Type

Exception Code	Description
0	Transfer successful.
1	Printer presentation error – partial voucher issued.
2	Redemption error from host – voucher rejected.
3	Redemption exception from host – voucher rejected.
4	Redemption exception from host – voucher stacked.
5	Redemption timed out by EGM – voucher rejected.
6	Voucher exceeds credit limit – voucher rejected.
7	Game state changed – voucher rejected.
8	Another transfer in process – voucher rejected.
9	Cannot mix non-cashable expiration – voucher rejected.
10	Cannot mix non-cashable credits – voucher rejected.
99	Voucher rejected – reason unknown

21.26.6 t_voucherSources Enumerations: Voucher Source Codes

The following table contains the enumerations for the `t_voucherSources` data type:

Table 21.38 Enumerations for t_voucherSources Data Type

Action	Description
`G2S_egmIssued`	Voucher issued by EGM or kiosk.
`G2S_systemIssued`	Voucher issued by system.

21.26.7 t_voucherStates Enumerations: Voucher Transaction Status Codes

The following table contains the enumerations for the `t_voucherStates` data type:

Table 21.39 Enumerations for t_voucherStates Data Type

Status	Description
`G2S_issueSent`	Voucher issued, waiting for acknowledgement.
`G2S_issueAcked`	Voucher issued and acknowledged.
`G2S_redeemSent`	Redemption requested; waiting for authorization.
`G2S_redeemAuth`	Redemption authorized; transfer in process.
`G2S_commitSent`	Transfer action complete/aborted; waiting for acknowledgement.
`G2S_commitAcked`	Transfer action complete/aborted and acknowledged.

21.27 Error Codes

The following table lists the error codes specific to the `voucher` class:

Table 21.40 voucher class Error Codes

Error Code	Suggested Error Text
G2S_VCX001	Invalid Voucher ID
G2S_VCX002	Invalid Transaction ID
G2S_VCX003	Duplicate Transaction ID

21.28 Event Codes

The following table provides general information for events that may be generated by devices within the voucher class. The definitive information regarding the details of each event, including affected data, is contained in the individual event descriptions that follow this table.

NOTE:
The requirements in the individual event descriptions assume that devices are owned by a G2S host. Requirements may be different for EGM-owned devices. See Section 1.20.1, EGM-Owned Devices, as well as the appropriate guidelines for implementing EGM-owned devices, for more details.

Table 21.41 voucher Class Event Codes (Sheet 1 of 2)
(CB = cabinet, VC = voucher, BN = bonus, GP = gamePlay, JP = handpay, WT = wat, HP = hopper, ND = noteDispenser, PG = progressive, DF = dft)

Event Code and Text	Affected Device Logs, Meters, Status*									
	CB	VC	BN	GP	JP	WT	HP	ND	PG	DF
G2S_VCE001 Device Disabled by EGM	I	S								
G2S_VCE002 Device Not Disabled by EGM	I	S								
G2S_VCE003 Device Disabled by Host	I	S								
G2S_VCE004 Device Not Disabled by Host	I	S								
G2S_VCE005 Device Configuration Changed by Host		S								
G2S_VCE006 Device Configuration Changed by Operator		S								
G2S_VCE009 Device Locked by Host	I	S								
G2S_VCE010 Device Not Locked by Host	I	S								
G2S_VCE101 Validation ID Data Expired		S								
G2S_VCE102 Validation ID Data Updated		S								
G2S_VCE103 Voucher Issued	M	LM S†	MF	MF	MF	MF	M	M	M	MF‡
G2S_VCE105 Voucher Issue Command Acknowledged		L								
G2S_VCE106 Voucher Redemption Requested		L								
G2S_VCE107 Voucher Authorized		L								
G2S_VCE108 Voucher Redeemed	M	LM								
G2S_VCE109 Voucher Rejected		L								
G2S_VCE111 Voucher Commit Command Acknowledged		L								
Extension in v3.0: g2sVSO										
G2S_VCE112 Validation System Offline		S								

Table 21.41 voucher Class Event Codes (Sheet 2 of 2)
(CB = cabinet, VC = voucher, BN = bonus, GP = gamePlay, JP = handpay, WT = wat, HP = hopper, ND = noteDispenser, PG = progressive, DF = dft)

Event Code and Text	Affected Device Logs, Meters, Status*									
	CB	VC	BN	GP	JP	WT	HP	ND	PG	DF
G2S_VCE113 Validation System Not Offline		S								

* **S** = The event may cause the status of the device to be directly affected.
I = The event may cause the status of the device to be indirectly affected (the status record is not considered affected data for this event).
L = The event may cause a transaction record for the device to be directly affected.
M = Meters within the indicated class may be updated as a result of the event and may be included as affected data with the event.
F = The event may be caused by (or associated with) a transaction in another class and, thus, that transaction may become a source reference for the transaction within this class (that transaction record is not considered affected data for this event).

† Extension in v2.1: g2s2

‡ Extension in v2.1: g2sDF

21.28.1 Affected Data

The following table lists the device status elements and transaction log elements that are considered affected data for events within the class. The table also identifies the classes that contain meters which are considered affected data for events within this class.

See Section 1.20, Event Subscriptions, for complete details on which data must be reported with events.

Table 21.42 Elements Included With Events

Affected Data	Element/Class
Device Status	`voucherStatus`
Transaction Log	`voucherLog`
Meters	`cabinet`, `gamePlay`, `handpay`, `hopper`, `noteDispenser`, `progressive`, `bonus`, `voucher`, `wat`
	Extension in v2.1: g2sDF
	`dft`

See Section 3.4.7, evaluate(state) for an explanation of the conventions "evaluate(*deviceStatus*`.egmEnabled`)" and "evaluate(`cabinetStatus.egmState`)" as used in the following Device State Change descriptions.

21.28.2 G2S_VCE001 Device Disabled by EGM

The EGM MUST generate this event when the `egmEnabled` attribute of the `voucherStatus` command for the device is changed from `true` to `false`. The EGM MUST generate this event regardless of whether the device was actually disabled as a result of the change to the `egmEnabled` attribute (the device may have been previously disabled due to some other factor).

Table 21.43 G2S_VCE001 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	`voucherStatus.egmEnabled = "false"`. evaluate(`cabinetStatus.egmState`).
Meter State Changes	None.
Log State Changes	None.
Related Info	None.

21.28.3 G2S_VCE002 Device Not Disabled by EGM

The EGM MUST generate this event when the `egmEnabled` attribute of the `voucherStatus` command for the device is changed from `false` to `true`. The EGM MUST generate this event regardless of whether the device was actually enabled as a result of the change to the `egmEnabled` attribute (other factors may keep the device in a disabled state).

Table 21.44 G2S_VCE002 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	`voucherStatus.egmEnabled = "true"`. evaluate(`cabinetStatus.egmState`).
Meter State Changes	None.
Log State Changes	None.
Related Info	None.

21.28.4 G2S_VCE003 Device Disabled by Host

The EGM MUST generate this event when the `hostEnabled` attribute of the `voucherStatus` command for the device is changed from `true` to `false`. The EGM MUST generate this event regardless of whether the device was actually disabled as a result of the change to the `hostEnabled` attribute (the device may have been previously disabled due to some other factor). The EGM MUST generate this event regardless of whether the change to the `hostEnabled` attribute occurred because of a `setVoucherState` command from the owner host with the `enable` attribute set to `false`, a restart, or other means.

When the `hostEnabled` attribute of the `voucherStatus` command is set to `false` for a `voucher` device—for example, if set to `false` by the owner — the `valIdListLife` timer in the `voucherProfile` command MUST be expired. This is intended to allow an orderly restart of the `voucher` device by the owner.

Table 21.45 G2S_VCE003 Device, Meter, Log Changes, and Related Info (Sheet 1 of 2)

	Details
Device State Changes	`voucherStatus.hostEnabled = "false"`. evaluate(`cabinetStatus.egmState`). Expire `voucherProfile.valIdListLife` timer.
Meter State Changes	None.

Table 21.45 G2S_VCE003 Device, Meter, Log Changes, and Related Info (Sheet 2 of 2)

	Details
Log State Changes	None.
Related Info	`setVoucherState`.

21.28.5 G2S_VCE004 Device Not Disabled by Host

The EGM MUST generate this event when the `hostEnabled` attribute of the `voucherStatus` command for the device is changed from `false` to `true`. The EGM MUST generate this event regardless of whether the device was actually enabled as a result of the change to the `hostEnabled` attribute (other factors may keep the device in a disabled state). The EGM MUST generate this event regardless of whether the change to the `hostEnabled` attribute occurred because of a `setVoucherState` command from the owner host with the `enable` attribute set to `true`, a restart, or other means.

Table 21.46 G2S_VCE004 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	`voucherStatus.hostEnabled = "true"`. evaluate(`cabinetStatus.egmState`).
Meter State Changes	None.
Log State Changes	None.
Related Info	`setVoucherState`.

21.28.6 G2S_VCE005 Device Configuration Changed by Host

This event MUST be generated by the EGM after the configuration for the device has been changed by the configurator of the device via commands within the `optionConfig` class. The event MUST be generated after the `optionChangeStatus` command, which reports that the changes have been applied, is generated by the `optionConfig` device.

Table 21.47 G2S_VCE005 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	`voucherStatus.configurationId` set by host.
Meter State Changes	None.
Log State Changes	None.
Related Info	`optionChangeStatus`.

21.28.7 G2S_VCE006 Device Configuration Changed by Operator

This event MUST be generated by the EGM after the configuration for the device has been changed by an entity other than the configurator of the device — for example, when an operator makes changes locally at the EGM via an operator menu or other similar mechanism. The event MUST NOT be generated until the changes have been applied — for example, after the operator commits the changes or exits the operator menu.

This event MUST also be generated by the EGM after the configuration for the device has been changed due to a reset, restart, or similar action.

Table 21.48 G2S_VCE006 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	`voucherStatus.configurationId = "0"`.
Meter State Changes	None.
Log State Changes	None.
Related Info	None.

21.28.8 G2S_VCE009 Device Locked by Host

This event MUST be generated by the EGM after the `hostLocked` attribute of the `voucherStatus` command for the `voucher` device has been changed from `false` to `true`. The EGM MUST generate this event regardless of whether the change occurred because of a `setVoucherLockOut` command issued by a host with the `lockOut` attribute set to `true`, a restart, or other means. See Section 3.4, Disable, Lockout, and Cabinet State for more details.

Table 21.49 G2S_VCE009 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	`voucherStatus.hostLocked = "true"`. evaluate(`cabinetStatus.egmState`).
Meter State Changes	None.
Log State Changes	None.
Related Info	`setVoucherLockOut`.

21.28.9 G2S_VCE010 Device Not Locked by Host

This event MUST be generated by the EGM after the `hostLocked` attribute of the `voucherStatus` command for the `voucher` device has been changed from `true` to `false`. The EGM MUST generate this event regardless of whether the change occurred because of a `setVoucherLockOut` command issued by a host with the `lockOut` attribute set to `false`, a restart, or other means. See Section 3.4, Disable, Lockout, and Cabinet State for more details.

Table 21.50 G2S_VCE010 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	`voucherStatus.hostLocked = "false"`. evaluate(`cabinetStatus.egmState`).
Meter State Changes	None.
Log State Changes	None.
Related Info	`setVoucherLockOut`.

21.28.10 G2S_VCE101 Validation ID Data Expired

21.28.10.1 Event Description

This event MUST be generated by the EGM when the `valIdListLife` timer expires. See Section 21.14, getValidationData Command, for more details about refreshing the list of validation identifiers when the current set of validation identifiers has expired.

21.28.10.1.1 allowVoucherIssue and allowVoucherRedeem Attributes

Extension in v2.0.0: g2s1

When the `allowVoucherIssue` attribute of the `voucherProfile` command is set to `false` for a `voucher` device the EGM MUST NOT generate the `G2S_VCE101 Validation ID Data Expired` event for the `voucher` device.

21.28.10.2 Device, Meter, and Log Changes, and Related Info

Table 21.51 G2S_VCE101 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	`voucherStatus.egmEnabled = "false"`.
Meter State Changes	None.
Log State Changes	None.
Related Info	`getValidationData`. If the value of the `voucherStatus.egmEnabled` attribute is changed to `false`, event `G2S_VCE001 Device Disabled by EGM` MUST be generated.

21.28.11 G2S_VCE102 Validation ID Data Updated

This event MUST be generated by the EGM after the validation ID data has been updated by a `validationData` command.

Table 21.52 G2S_VCE102 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	evaluate(`voucherStatus.egmEnabled`).
Meter State Changes	None.
Log State Changes	None.
Related Info	`validationData`. If the value of the `voucherStatus.egmEnabled` attribute is changed to `true`, event `G2S_VCE002 Device Not Disabled by EGM` MUST be generated.

21.28.12 G2S_VCE103 Voucher Issued

This event MUST be generated by the EGM after a voucher has been issued and all associated meters have been updated.

Table 21.53 G2S_VCE103 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	None.
Meter State Changes	• Meter updates associated with cash-outs to vouchers are documented with this event. • Meter updates associated with game results, which may include the issuance of vouchers, are documented with event `G2S_GPE112 Game Ended` in the `gamePlay` class. • Meter updates associated with cancelled credits, which may include the issuance of vouchers, are documented with event `G2S_JPE104 Handpay Keyed Off` in the `handpay` class. • Meter updates associated with bonus awards, which may include the issuance of vouchers, are documented with event `G2S_BNE104 Bonus Award Paid` in the `bonus` class. • Meter updates associated with WAT transfers to vouchers are documented with event `G2S_WTE107 WAT Transfer Completed` in the `wat` class. • Regardless of the event with which the meter updates are documented, when the issuance of vouchers takes place, the meter updates MUST be treated as affected data for this event. • See Section 5.14, Meter Consistency in Complex Transactions for more details. See Table 21.54.
	Extension in v2.1: g2sDF • Meter updates associated with DFT transfers, which may include the issuance of vouchers, are documented with event `G2S_DFE105 Transfer Successfully Completed` in the `dft` class.
Log State Changes	Create `voucherLog`. See Table 21.55. Create `voucherSourceRef` for source transactions, as appropriate. See Table 21.56, Table 21.57, Table 21.58, and Table 21.59.
	Extension in v2.1: g2sDF Create `voucherSourceRef` for `dft` transactions, as appropriate. See Table 21.60.
Related Info	`issueVoucher`.

NOTES:

1. Transactions within the `bonus`, `gamePlay`, `handpay`, and `wat` classes may be source reference entries for a transaction within the `voucher` class. These source references are recorded with the Voucher Issued event.

2. When combining cashable and promotional credits on one voucher, the promotional credits are converted to cashable. While total in and total out will still balance, the individual credit types will not balance.

The following table lists the meter updates that MUST occur following the printing of a payment voucher for a player cash-out.

Table 21.54 G2S_VCE103 Meter State Changes

Bal	Device	Meter	Description
Y	cabinet	G2S_playerCashableAmt	Decrements by the voucher amount paid from the cashable credit meter.
Y	cabinet	G2S_playerPromoAmt	Decrements by the voucher amount paid from the promotional credit meter.
Y	cabinet	G2S_playerNonCashAmt	Decrements by the voucher amount paid from the non-cashable credit meter.
Y	voucher	G2S_cashableOutAmt	Increments by the voucher amount paid as cashable credits.
	voucher	G2S_cashableOutCnt	Increments by 1 (one) if cashable voucher issued.
Y	voucher	G2S_promoOutAmt	Increments by the voucher amount paid as promotional credits.
	voucher	G2S_promoOutCnt	Increments by 1 (one) if promotional voucher issued.
Y	voucher	G2S_nonCashOutAmt	Increments by the voucher amount paid as non-cashable credits.
	voucher	G2S_nonCashOutCnt	Increments by 1 (one) if non-cashable voucher issued.

Table 21.55 G2S_VCE103 Transaction Changes – Create voucherLog (Sheet 1 of 2)

Attribute	Set to Value …
logSequence	Next value in the series.
deviceId	voucher.deviceId.
transactionId	Next value in the series.
voucherState	Set to G2S_issueSent.
voucherAction	Set to G2S_issue.
idReaderType	idReaderProfile.idReaderType of the idReader device associated with the voucher device. If no idReader device is associated or the device is disabled then set to G2S_none.
idNumber	idReaderStatus.idNumber present in the idReader device associated with the voucher device. If no idReader device is associated or the device is disabled then set to <empty>.

Table 21.55 G2S_VCE103 Transaction Changes – Create voucherLog (Sheet 2 of 2)

Attribute	Set to Value …
playerId	idReaderStatus.playerId present in the idReader device associated with the voucher device. If no idReader device is associated or the device is disabled then set to <empty>.
validationId	The validation identifier of the voucher.
voucherAmt	Amount that the voucher was issued for.
creditType	Type of credits for which the voucher was issued.
voucherSource	Set to G2S_egmIssued.
largeWin	Set to true if voucher printed due to key-off of large win to voucher; otherwise, set to false.
voucherSequence	Sequence number printed on voucher.
expireCredits	Set to true if date/time expiration assigned to voucher.
expireDateTime	Date/time of voucher expiration (if expireCredits is set to true).
expireDays	Number of days before expiration of the voucher (if expireCredits is set to false).
hostAction	Set to G2S_egmAction.
hostException	Set to 0 (zero).
transferAmt	Amount that the voucher was issued for.
transferDateTime	Date/time voucher was issued.
egmAction	Set to G2S_issued.
egmException	Set to 0 (zero) or 1 (one) as appropriate.

Table 21.56 G2S_VCE103 voucherSourceRef: bonus Source References (Sheet 1 of 2)

Attribute	Source	Target
deviceClass	G2S_bonus	
deviceId	bonusLog.deviceId	
transactionID	bonusLog.transactionId	
logSequence	bonusLog.logSequence	
cashableAmt	If the bonusLog.creditType attribute is set to G2S_cashable, set to the cashable amount of the source transaction, typically bonusLog.bonusAwardAmt; otherwise set to 0 (zero). See Section 1.18 for more details.	If cashableAmt is greater than 0 (zero), set the voucherLog.voucherAmt attribute to the cashableAmt value, and set the voucherLog.creditType attribute to G2S_cashable.

Table 21.56 G2S_VCE103 voucherSourceRef: bonus Source References (Sheet 2 of 2)

Attribute	Source	Target
promoAmt	If the bonusLog.creditType attribute is set to G2S_promo, set to the promotional amount of the source transaction, typically bonusLog.bonusAwardAmt; otherwise set to 0 (zero). See Section 1.18 for more details.	If promoAmt is greater than 0 (zero), set the voucherLog.voucherAmt attribute to the promoAmt value, and set the voucherLog.creditType attribute to G2S_promo.
nonCashAmt	If bonusLog.creditType is set to G2S_nonCash, set to the non-cashable amount of the source transaction, typically bonusLog.bonusAwardAmt; otherwise set to 0 (zero). See Section 1.18 for more details.	If nonCashAmt is greater than 0 (zero), set the voucherLog.voucherAmt attribute to the nonCashAmt value, and set the voucherLog.creditType attribute to G2S_nonCash.

NOTE:
When a bonus is awarded across multiple game cycles, the source reference element MUST only reflect the amount awarded in the current game cycle.

Table 21.57 G2S_VCE103 voucherSourceRef: gamePlay Source References

Attribute	Source	Target
deviceClass	G2S_gamePlay	
deviceId	recallLog.deviceId	
transactionID	recallLog.transactionId	
logSequence	recallLog.logSequence	
cashableAmt	recallLog.finalWin	Set the voucherLog.voucherAmt attribute to the cashableAmt value, and set the voucherLog.creditType attribute to G2S_cashable.
promoAmt	0 (zero).	
nonCashAmt	0 (zero).	

Table 21.58 G2S_VCE103 voucherSourceRef: handpay Source References* (Sheet 1 of 2)

Attribute	Source	Target
deviceClass	G2S_handpay	
deviceId	handpayLog.deviceId	
transactionID	handpayLog.transactionId	
logSequence	handpayLog.logSequence	

Table 21.58 G2S_VCE103 voucherSourceRef: handpay Source References* (Sheet 2 of 2)

Attribute	Source	Target
cashableAmt	Set to the cashable amount of source transaction, typically handpayLog.requestCashableAmt, unless the requested amount was reduced by the host. See Section 1.18 for more details.	If cashableAmt is greater than 0 (zero), set the voucherLog.voucherAmt attribute to the cashableAmt value, and set the voucherLog.creditType attribute to G2S_cashable.
promoAmt	Set to the promotional amount of source transaction, typically handpayLog.requestPromoAmt, unless the requested amount was reduced by the host. See Section 1.18 for more details.	If promoAmt is greater than 0 (zero), set the voucherLog.voucherAmt attribute to the promoAmt value, and set the voucherLog.creditType attribute to G2S_promo.
nonCashAmt	Set to the non-cashable amount of source transaction, typically handpayLog.requestNonCashAmt, unless the requested amount was reduced by the host. See Section 1.18 for more details.	If nonCashAmt is greater than 0 (zero), set the voucherLog.voucherAmt attribute to the nonCashAmt value, and set the voucherLog.creditType attribute to G2S_nonCash.

* Note: A single handpay transaction may trigger multiple voucher transactions — one for each creditType requiring payment. Only the amount related to the applicable creditType MUST be recorded in the source reference.

Table 21.59 G2S_VCE103 voucherSourceRef: wat Source References*

Attribute	Source	Target
deviceClass	G2S_wat	
deviceId	watLog.deviceId	
transactionID	watLog.transactionId	
logSequence	watLog.logSequence	
cashableAmt	Set to the cashable amount of source transaction, typically watLog.authCashableAmt. See Section 1.18 for more details.	If cashableAmt is greater than 0 (zero), set the voucherLog.voucherAmt attribute to the cashableAmt value, and set the voucherLog.creditType attribute to G2S_cashable.
promoAmt	Set to the promotional amount of source transaction, typically watLog.authPromoAmt. See Section 1.18 for more details.	If promoAmt is greater than 0 (zero), set the voucherLog.voucherAmt attribute to the promoAmt value, and set the voucherLog.creditType attribute to G2S_promo.
nonCashAmt	Set to the non-cashable amount of source transaction, typically watLog.authNonCashAmt. See Section 1.18 for more details.	If nonCashAmt is greater than 0 (zero), set the voucherLog.voucherAmt attribute to the nonCashAmt value, and set the voucherLog.creditType attribute to G2S_nonCash.

* Note: A single WAT transaction may trigger multiple voucher transactions — one for each `creditType` requiring payment. Only the amount related to the applicable `creditType` MUST be recorded in the source reference.

21.28.12.1 G2S_VCE103 voucherSourceRef: dft Source References

Extension in v2.1: g2sDF

Table 21.60 G2S_VCE103 voucherSourceRef: dft Source References

Attribute	Source	Target
`deviceClass`	`G2S_dft`	
`deviceId`	`dftLog.devcieId`	
`transactionId`	`dftLog.transactionId`	
`logSequence`	`dftLog.logSequence`	
`cashableAmt`	`dftLog.authCashableAmt`	If `cashableAmt` is greater than `0` (zero), set the `voucherLog.voucherAmt` attribute to `cashableAmt` and set `voucherLog.creditType` to `G2S_cashable`.
`promoAmt`	`dftLog.authPromoAmt`	If `promoAmt` is greater than `0` (zero), set the `voucherLog.voucherAmt` attribute to `promoAmt` and set `voucherLog.creditType` to `G2S_promo`.
`nonCashAmt`	`dftLog.authNonCashAmt`	If `cashableAmt` is greater than `0` (zero), set the `voucherLog.voucherAmt` attribute to `nonCashAmt` and set `voucherLog.creditType` to `G2S_nonCash`.

21.28.13 G2S_VCE105 Voucher Issue Command Acknowledged

This event MUST be generated by the EGM after an `issueVoucherAck` response has been received from the host. See Section 21.17, issueVoucherAck Command, for more details.

Table 21.61 G2S_VCE105 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	None.
Meter State Changes	None.
Log State Changes	Update `voucherLog`. See Table 21.62.
Related Info	`issueVoucherAck`.

Table 21.62 G2S_VCE105 Transaction Changes – voucherLog

Attribute	Set to Value …
logSequence	Unchanged.
deviceId	Unchanged.
transactionId	Unchanged.
voucherState	Set to G2S_issueAcked.
voucherAction	Unchanged.
idReaderType	Unchanged.
idNumber	Unchanged.
playerId	Unchanged.
validationId	Unchanged.
voucherAmt	Unchanged.
creditType	Unchanged.
voucherSource	Unchanged.
largeWin	Unchanged.
voucherSequence	Unchanged.
expireCredits	Unchanged.
expireDateTime	Unchanged.
expireDays	Unchanged.
hostAction	Unchanged.
hostException	Unchanged.
transferAmt	Unchanged.
transferDateTime	Unchanged.
egmAction	Unchanged.
egmException	Unchanged.

21.28.14 G2S_VCE106 Voucher Redemption Requested

This event MUST be generated by the EGM after a voucher has been escrowed.

Table 21.63 G2S_VCE106 Device, Meter, Log Changes, and Related Info (Sheet 1 of 2)

	Details
Device State Changes	None.

Table 21.63 G2S_VCE106 Device, Meter, Log Changes, and Related Info (Sheet 2 of 2)

	Details
Meter State Changes	None.
Log State Changes	Create `voucherLog`. See Table 21.64.
Related Info	`redeemVoucher`.

Table 21.64 G2S_VCE106 Transaction Changes – voucherLog (Sheet 1 of 2)

Attribute	Set to Value …
`logSequence`	Next value in the series.
`deviceId`	`voucher.deviceId`.
`transactionId`	Next value in the series.
`voucherState`	Set to `G2S_redeemSent`.
`voucherAction`	Set to `G2S_redeem`.
`idReaderType`	`idReaderProfile.idReaderType` of the `idReader` device associated with the `voucher` device. If no `idReader` device is associated or the device is disabled then set to `G2S_none`.
`idNumber`	`idReaderStatus.idNumber` present in the `idReader` device associated with the `voucher` device. If no `idReader` device is associated or the device is disabled then set to <empty>.
`playerId`	`idReaderStatus.playerId` present in the `idReader` device associated with the `voucher` device. If no `idReader` device is associated or the device is disabled then set to <empty>.
`validationId`	Validation identifier of the voucher.
`voucherAmt`	Set to `0` (zero).
`creditType`	Set to `G2S_cashable`.
`voucherSource`	Set to `G2S_egmIssued`.
`largeWin`	Set to `false`.
`voucherSequence`	Set to `0` (zero).
`expireCredits`	Set to `false`.
`expireDateTime`	Set to `2000-01-01T00:00:00.000-00:00`.
`expireDays`	Set to `0` (zero).
`hostAction`	Set to `G2S_egmAction`.
`hostException`	Set to `0` (zero).
`transferAmt`	Set to `0` (zero).

Table 21.64 G2S_VCE106 Transaction Changes – voucherLog (Sheet 2 of 2)

Attribute	Set to Value …
transferDateTime	Current date/time.
egmAction	Set to G2S_pending.
egmException	Set to 0 (zero).

21.28.15 G2S_VCE107 Voucher Authorized

This event MUST be generated by the EGM after an authorizeVoucher response has been received from the host. See Section 21.19, authorizeVoucher Command, for more details.

Table 21.65 G2S_VCE107 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	None.
Meter State Changes	None.
Log State Changes	Update voucherLog. See Table 21.66.
Related Info	authorizeVoucher.

Table 21.66 G2S_VCE107 Transaction Changes – voucherLog (Sheet 1 of 2)

Attribute	Set to Value …
logSequence	Unchanged.
deviceId	Unchanged.
transactionId	Unchanged.
voucherState	Set to G2S_redeemAuth.
voucherAction	Unchanged.
idReaderType	Unchanged.
idNumber	Unchanged.
playerId	Unchanged.
validationId	Unchanged.
voucherAmt	authorizeVoucher.voucherAmt.
creditType	authorizeVoucher.creditType.
voucherSource	authorizeVoucher.voucherSource.
largeWin	authorizeVoucher.largeWin.
voucherSequence	authorizeVoucher.voucherSequence.

Table 21.66 G2S_VCE107 Transaction Changes – voucherLog (Sheet 2 of 2)

Attribute	Set to Value …
expireCredits	authorizeVoucher.expireCredits.
expireDateTime	authorizeVoucher.expireDateTime.
expireDays	Unchanged.
hostAction	authorizeVoucher.hostAction.
hostException	authorizeVoucher.hostException.
transferAmt	Unchanged.
transferDateTime	Current date/time.
egmAction	Unchanged.
egmException	Unchanged.

21.28.16 G2S_VCE108 Voucher Redeemed

This event MUST be generated by the EGM after a voucher redemption request has been authorized, the voucher has been redeemed, all associated meters have been updated, and the voucher has been stacked or rejected, as appropriate.

Table 21.67 G2S_VCE108 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	None.
Meter State Changes	See Table 21.68 and Table 21.69.
Log State Changes	Update voucherLog. See Table 21.70.
Related Info	commitVoucher.

The following table lists the meter updates that MUST occur following the redemption of an EGM-issued voucher.

Table 21.68 G2S_VCE108 Meter Updates – EGM-Issued Voucher Redeemed (Sheet 1 of 2)

Bal	Device	Meter	Description
Y	cabinet	G2S_playerCashableAmt	Increments by the voucher amount if paid to the cashable credit meter.
Y	cabinet	G2S_playerPromoAmt	Increments by the voucher amount if paid to the promotional credit meter.
Y	cabinet	G2S_playerNonCashAmt	Increments by the voucher amount if paid to the non-cashable credit meter.

Table 21.68 G2S_VCE108 Meter Updates – EGM-Issued Voucher Redeemed (Sheet 2 of 2)

Bal	Device	Meter	Description
Y	voucher	G2S_cashableInAmt	Increments by the voucher amount if paid to the cashable credit meter.
	voucher	G2S_cashableInCnt	Increments by 1 (one) if cashable voucher redeemed.
Y	voucher	G2S_promoInAmt	Increments by the voucher amount if paid to the promotional credit meter.
	voucher	G2S_promoInCnt	Increments by 1 (one) if promotional voucher redeemed.
Y	voucher	G2S_nonCashInAmt	Increments by the voucher amount if paid to the non-cashable credit meter.
	voucher	G2S_nonCashInCnt	Increments by 1 (one) if non-cashable voucher redeemed.

The following table lists the meter updates that MUST occur following the redemption of a system-issued voucher.

Table 21.69 G2S_VCE108 Meter Updates – System-Issued Voucher Redeemed

Bal	Device	Meter	Description
Y	cabinet	G2S_playerCashableAmt	Increments by the voucher amount if paid to the cashable credit meter.
Y	cabinet	G2S_playerPromoAmt	Increments by the voucher amount if paid to the promotional credit meter.
Y	cabinet	G2S_playerNonCashAmt	Increments by the voucher amount if paid to the non-cashable credit meter.
Y	voucher	G2S_cashableSysInAmt	Increments by the voucher amount if paid to the cashable credit meter.
	voucher	G2S_cashableSysInCnt	Increments by 1 (one) if cashable voucher redeemed.
Y	voucher	G2S_promoSysInAmt	Increments by the voucher amount if paid to the promotional credit meter.
	voucher	G2S_promoSysInCnt	Increments by 1 (one) if promotional voucher redeemed.
Y	voucher	G2S_nonCashSysInAmt	Increments by the voucher amount if paid to the non-cashable credit meter.
	voucher	G2S_nonCashSysInCnt	Increments by 1 (one) if non-cashable voucher redeemed.

Table 21.70 G2S_VCE108 Transaction Changes – voucherLog

Attribute	Set to Value …
logSequence	Unchanged.
deviceId	Unchanged.
transactionId	Unchanged.
voucherState	Set to G2S_commitSent.
voucherAction	Unchanged.
idReaderType	Unchanged.
idNumber	Unchanged.
playerId	Unchanged.
validationId	Unchanged.
voucherAmt	Unchanged.
creditType	Unchanged.
voucherSource	Unchanged.
largeWin	Unchanged.
voucherSequence	Unchanged.
expireCredits	Set to true if date/time expiration assigned to non-cashable credits.
expireDateTime	Set to date/time assigned to credits if expireCredits is set to true.
expireDays	Unchanged.
hostAction	Unchanged.
hostException	Unchanged.
transferAmt	Set to voucherLog.voucherAmt.
transferDateTime	Set to date/time the transfer was completed.
egmAction	Set to G2S_redeemed (when stacked) or G2S_rejected (when not stacked), as appropriate.
egmException	Set to 0 (zero).

21.28.17 G2S_VCE109 Voucher Rejected

This event MUST be generated by the EGM after a voucher redemption request has been denied (or a voucher has not been redeemed for some other reason) and the voucher has been stacked or rejected, as appropriate.

Table 21.71 G2S_VCE109 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	None.
Meter State Changes	None.
Log State Changes	Update `voucherLog`. See Table 21.72.
Related Info	`commitVoucher`.

Table 21.72 G2S_VCE109 Transaction Changes – voucherLog (Sheet 1 of 2)

Attribute	Set to Value …
`logSequence`	Unchanged.
`deviceId`	Unchanged.
`transactionId`	Unchanged.
`voucherState`	Set to `G2S_commitSent`.
`voucherAction`	Unchanged.
`idReaderType`	Unchanged.
`idNumber`	Unchanged.
`playerId`	Unchanged.
`validationId`	Unchanged.
`voucherAmt`	Unchanged.
`creditType`	Unchanged.
`voucherSource`	Unchanged.
`largeWin`	Unchanged.
`voucherSequence`	Unchanged.
`expireCredits`	Unchanged.
`expireDateTime`	Unchanged.
`expireDays`	Unchanged.
`hostAction`	Unchanged.
`hostException`	Unchanged.
`transferAmt`	Unchanged.
`transferDateTime`	Date/time the voucher was rejected.

Table 21.72 G2S_VCE109 Transaction Changes – voucherLog (Sheet 2 of 2)

Attribute	Set to Value …
egmAction	Set to G2S_rejected (when not stacked) or G2S_redeemed (when stacked), as appropriate.
egmException	Set to: 2 (two) if the host refused the transaction (returned error code). 3 (three) if hostException is not set to 0 (zero) and voucher rejected. 4 (four) if hostException is not set to 0 (zero) and voucher stacked. 5 (five) if the EGM aborted the transaction before it was authorized (timed out). Set to other value, as appropriate, if EGM could not perform the transfer after it was authorized.

21.28.18 G2S_VCE111 Voucher Commit Command Acknowledged

This event MUST be generated by the EGM after a commitVoucherAck response has been received from the host. See Section 21.21, commitVoucherAck Command, for more details.

Table 21.73 G2S_VCE111 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	None.
Meter State Changes	None.
Log State Changes	Update voucherLog. See Table 21.74.
Related Info	commitVoucherAck.

Table 21.74 G2S_VCE111 Transaction Changes – voucherLog (Sheet 1 of 2)

Attribute	Set to Value …
logSequence	Unchanged.
deviceId	Unchanged.
transactionId	Unchanged.
voucherState	Set to G2S_commitAcked.
voucherAction	Unchanged.
idReaderType	Unchanged.
idNumber	Unchanged.
playerId	Unchanged.
validationId	Unchanged.
voucherAmt	Unchanged.

Table 21.74 G2S_VCE111 Transaction Changes – voucherLog (Sheet 2 of 2)

Attribute	Set to Value …
creditType	Unchanged.
voucherSource	Unchanged.
largeWin	Unchanged.
voucherSequence	Unchanged.
expireCredits	Unchanged.
expireDateTime	Unchanged.
expireDays	Unchanged.
hostAction	Unchanged.
hostException	Unchanged.
transferAmt	Unchanged.
transferDateTime	Unchanged.
egmAction	Unchanged.
egmException	Unchanged.

21.28.19 G2S_VCE112 Validation System Offline

Extension in v3.0: g2sVSO

The EGM MUST generate this event when the `systemOnLine` attribute of the `voucherStatus` command is changed from `true` to `false`. The EGM MUST generate this event regardless of whether the change occurred due to voucher host offline or no receipt of an `issueVoucherAck` command. See Section 21.11.1.1, systemOnLine Attribute, for more details.

Table 21.75 G2S_VCE112 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	`voucherStatus.systemOnLine = "false"`. If `voucherProfile.printOffLine = "false"` then `voucherStatus.egmEnabled = "false"`.
Meter State Changes	None.
Log State Changes	None.
Related Info	If the value of the `voucherStatus.egmEnabled` attribute is changed to `false`, event `G2S_VCE001 Device Disabled by EGM` MUST be generated.

21.28.20 G2S_VCE113 Validation System Not Offline

Extension in v3.0: g2sVSO

The EGM MUST generate this event when the `systemOnLine` attribute of the `voucherStatus` command is changed from `false` to `true`. See Section 21.11.1.1, systemOnLine Attribute, for more details.

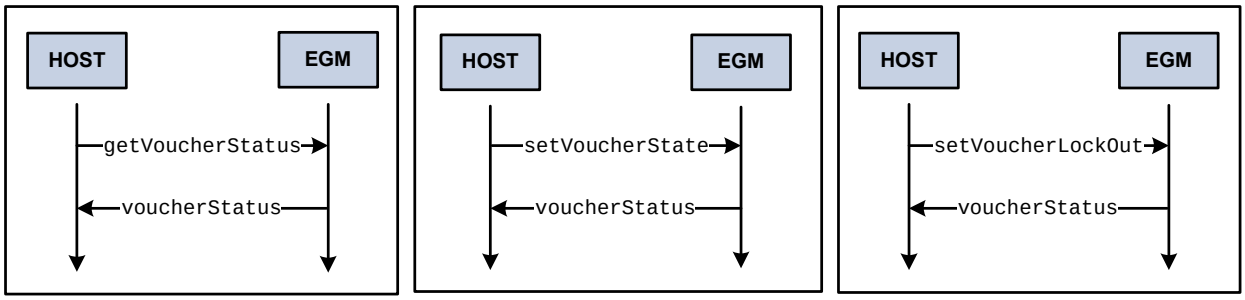
Table 21.76 G2S_VCE113 Device, Meter, Log Changes, and Related Info

	Details
Device State Changes	`voucherStatus.systemOnLine = "true"`. evaluate(`voucherStatus.egmEnabled`).
Meter State Changes	None.
Log State Changes	None.
Related Info	If the value of the `voucherStatus.egmEnabled` attribute is changed to `true`, event `G2S_VCE002 Device Not Disabled by EGM` MUST be generated.

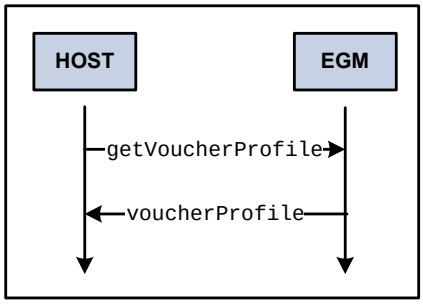
21.29 Examples

The sections below include diagrams that show the command sequences for status, profile, validation data and log commands. Section 21.29.4 includes examples, as well as the command sequence diagram, for voucher issuance and redemption commands.

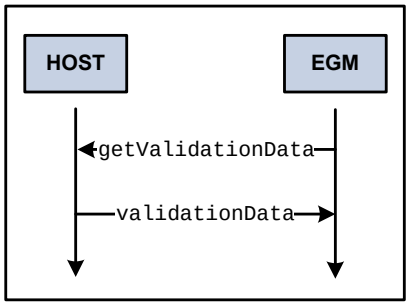
21.29.1 Voucher Device Status Commands



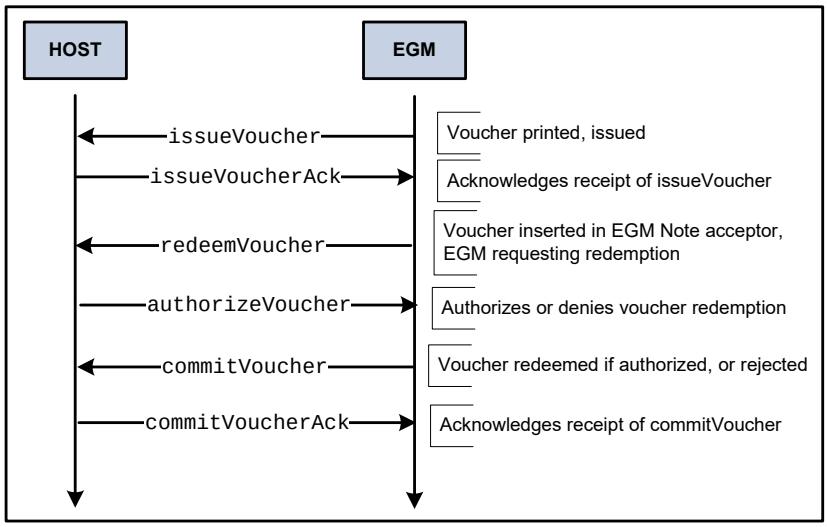
21.29.2 Voucher Device Profile Commands



21.29.3 Validation Data Commands



21.29.4 Voucher Issuance and Redemption Commands



21.29.4.1 issueVoucher: EGM to Host

The following example highlights the construction of an `issueVoucher` command sent from an EGM to a host after a $100 voucher has been issued.

```
<voucher
    deviceId = "1"
    dateTime = "2003-03-01T13:23:27.321-05:00"
    commandId = "3001"
    sessionId = "1001"
    sessionType = "G2S_request"
    timeToLive = "30000" >
    <issueVoucher
        transactionId = "23456"
        idReaderType = "G2S_magCard"
        idNumber = "987654"
        playerId = "77777"
        validationId = "0123456789012345678"
        voucherAmt = "10000000"
        creditType = "G2S_cashable"
        voucherSequence = "2"
        transferDateTime = "2006-03-25T13:23:27.321-05:00"
        egmAction = "G2S_issued" />
</voucher>
```

21.29.4.2 issueVoucherAck Response: Host to EGM

The following example highlights the construction of an `issueVoucherAck` command sent from a host to an EGM to acknowledge the receipt of an `issueVoucher` command.

```
<voucher
    deviceId = "1"
    dateTime = "2006-03-25T13:23:27.432-05:00"
    commandId = "2001"
    sessionId = "1001"
    sessionType = "G2S_response" >
```

```
    <issueVoucherAck
        transactionId = "23456" />
</voucher>
```

21.29.4.3 redeemVoucher Request: EGM to Host

The following example highlights the construction of a `redeemVoucher` command sent from an EGM to a host to initiate a voucher redemption cycle.

```
<voucher
    deviceId = "1"
    dateTime = "2006-03-25T13:23:27.543-05:00"
    commandId = "3002"
    sessionId = "1002"
    sessionType = "G2S_request"
    timeToLive = "30000" >
    <redeemVoucher
        transactionId = "23457"
        idReaderType = "G2S_magCard"
        idNumber = "987654"
        playerId = "77777"
        validationId = "012345678901234567" />
</voucher>
```

21.29.4.4 authorizeVoucher Response: Host to EGM

The following example highlights the construction of an `authorizeVoucher` command sent from a host to an EGM in response to a `redeemVoucher` command to authorize a voucher redemption for $25.

```
<voucher
    deviceId = "1"
    dateTime = "2006-03-25T13:23:27.654-05:00"
    commandId = "2002"
    sessionId = "1002"
    sessionType = "G2S_response" >
    <authorizeVoucher
        transactionId = "23457"
        validationId = "012345678901234567"
        voucherAmt = "2500000"
        creditType = "G2S_nonCash"
        expireCredits = "true"
        expireDateTime = "2006-07-03T00:00:00.000-00:00" />
</voucher>
```

21.29.4.5 commitVoucher Request: EGM to Host

The following example highlights the construction of a `commitVoucher` command sent from an EGM to a host to report that a voucher has been redeemed.

```
<voucher
    deviceId = "1"
    dateTime = "2006-03-25T13:23:27.765-05:00"
    commandId = "3003"
    sessionId = "1003"
    sessionType = "G2S_request"
```

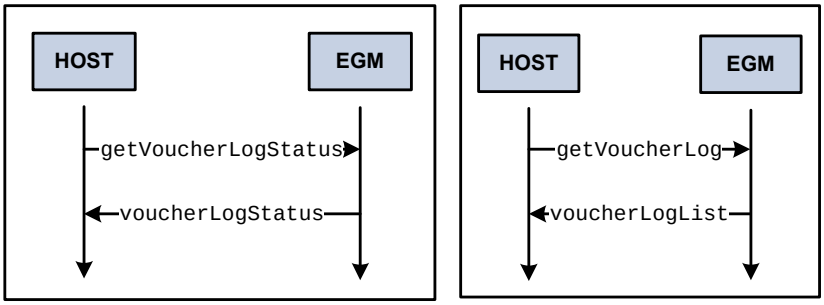
```
    timeToLive = "30000" >
    <commitVoucher
        transactionId = "23457"
        validationId = "0123456789012345678"
        voucherAmt = "2500000"
        creditType = "G2S_nonCash"
        expireCredits = "true"
        expireDateTime = "2006-07-03T00:00:00.000-00:00"
        transferAmt = "2500000"
        transferDateTime = "2006-03-25T13:23:27.765-05:00"
        egmAction = "G2S_redeemed" />
</voucher>
```

21.29.4.6 commitVoucherAck Response: Host to EGM

The following example highlights the construction of a `commitVoucherAck` command sent from a host to an EGM to acknowledge the receipt of a `commitVoucher` command.

```
<voucher
    deviceId = "1"
    dateTime = "2006-03-25T13:23:27.876-05:00"
    commandId = "2003"
    sessionId = "1003"
    sessionType = "G2S_response" >
    <commitVoucherAck
        transactionId = "23457" />
</voucher>
```

21.29.5 Voucher Log Commands



21.30 Voucher Device Option Configuration

The following tables identify the G2S-specified configuration option selections for `voucher` devices. These selections are set using commands from the `optionConfig` class. The current configuration option selections for a `voucher` device are reported via the `voucherProfile` command.

Further descriptions of the sub-parameters can be found under the `voucherProfile` command.

21.30.1 Option Group Definitions

optionGroupId	G2S_voucherOptions
optionGroupName	G2S Voucher Options

21.30.2 G2S Protocol Options Definitions

optionId	G2S_protocolOptions
securityLevel	G2S_administrator
minSelections	1
maxSelections	1
duplicates	n/a
paramKey	n/a
Parameter Type	Complex
paramId	G2S_protocolParams
paramName	G2S Protocol Parameters
paramHelp	Standard G2S protocol parameters for voucher devices

21.30.2.1 G2S Protocol Options Sub-Parameter Definitions

Table 21.77 G2S Protocol Options Sub-Parameter Definitions

paramId	paramName	Example	paramHelp
G2S_configurationId*	Configuration Identifier	123456	ID assigned by the last successful G2S option configuration
G2S_restartStatus	Enabled on Restart	true	Controls hostEnabled attribute status upon EGM restart
G2S_requiredForPlay	Required For Play	true	Device is required for game play
G2S_timeToLive	Time to Live	30000	Time to live value for requests (in milliseconds)

* The `configurationId` is included as an option parameter for the host system's convenience and is used for reference purposes only. The parameter MUST always be reported by the EGM with the `canModRemote` attribute set to `false`. As described in Section 9.16, setOptionChange Command the host MUST always specify the current value of the `configurationId` as reported by the EGM when setting this option — not the new `configurationId` specified by the host in the `setOptionChange` command. This means that the host MAY have to request the current option settings from the EGM to discover the current value of the `configurationId` before trying to change the option settings.

21.30.2.2 Example of G2S_protocolOptions Option

```
// G2S Protocol Options
<optionItem
    optionId = "G2S_protocolOptions"
    securityLevel = "G2S_administrator"
    minSelections = "1"
    maxSelections = "1" >
    <optionParameters>
        <complexParameter
            paramId = "G2S_protocolParams"
            paramName = "G2S Protocol Parameters"
            paramHelp = "Standard G2S protocol parameters for voucher devices" >
            <integerParameter
                paramId = "G2S_configurationId"
                paramName = "Configuration Identifier"
                paramHelp = "ID assigned by the last successful G2S option configuration"
                canModLocal = "false"
                canModRemote = "false" />
            <booleanParameter
                paramId = "G2S_restartStatus"
                paramName = "Enabled on Restart"
                paramHelp = "Controls hostEnabled attribute status upon EGM restart"
                canModLocal = "true"
                canModRemote = "true" />
            <booleanParameter
                paramId = "G2S_requiredForPlay"
                paramName = "Required For Play"
                paramHelp = "Device is required for game play"
                canModLocal = "true"
                canModRemote = "true" />
            <integerParameter
                paramId = "G2S_timeToLive"
                paramName = "Time To Live"
                paramHelp = "Time to live value for requests (in milliseconds)"
                canModLocal = "true"
                canModRemote = "true"
                minIncl = "0" />
        </complexParameter>
    </optionParameters>
    <optionCurrentValues>
        <complexValue paramId = "G2S_protocolParams" >
            <integerValue paramId = "G2S_configurationId" > 123456 </integerValue>
            <booleanValue paramId = "G2S_restartStatus" > true </booleanValue>
            <booleanValue paramId = "G2S_requiredForPlay" > true </booleanValue>
            <integerValue paramId = "G2S_timeToLive" >30000</integerValue>
        </complexValue>
    </optionCurrentValues>
    <optionDefaultValues>
        <complexValue paramId = "G2S_protocolParams" >
            <integerValue paramId = "G2S_configurationId" > 0 </integerValue>
```

```
            <booleanValue paramId = "G2S_restartStatus" > true </booleanValue>
            <booleanValue paramId = "G2S_requiredForPlay" > true </booleanValue>
            <integerValue paramId = "G2S_timeToLive" >30000</integerValue>
        </complexValue>
    </optionDefaultValues>
</optionItem>
```

21.30.3 Voucher Options Definitions

optionId	G2S_voucherOptions
securityLevel	G2S_operator
minSelections	1
maxSelections	1
duplicates	n/a
paramKey	n/a
Parameter Type	Complex
paramId	G2S_voucherParams
paramName	Voucher Options
paramHelp	Configuration parameters for this voucher device

21.30.3.1 Voucher Options Sub-Parameter Definitions

Table 21.78 Voucher Options Sub-Parameter Definitions (Sheet 1 of 2)

paramId	paramName	Example	paramHelp
G2S_idReaderId	ID Reader to Use	1	ID reader to use for this voucher device
G2S_combineCashableOut	Combine Cashable Credit Types	true	Combine cashable and promo credits on a single voucher
G2S_allowNonCashOut	Allow Non-Cashable Out	true	Allow vouchers for non-cashable credits
G2S_maxValIds	Maximum Validation Ids	20	Maximum validation IDs EGM may buffer
G2S_minLevelValIds	Minimum Level for Validation Ids	20	Minimum validation IDs EGM must maintain
G2S_valIdListRefresh	Validation ID Refresh Time	43200000 (12 hours)	Maximum time before EGM requests validation ID list update
G2S_valIdListLife	Validation ID List Life	86400000 (24 hours)	Maximum life of validation IDs without host intervention
G2S_voucherHoldTime	Maximum Voucher Hold Time	15000 (15 seconds)	Maximum time EGM should escrow a voucher
G2S_printOffLine	Print Offline	true	Allow vouchers to be printed while host is offline

Table 21.78 Voucher Options Sub-Parameter Definitions (Sheet 2 of 2)

paramId	paramName	Example	paramHelp
G2S_expireCashPromo	Expire Days Cash Promo	30 (days)	Number of days before cashable and promo vouchers expire
G2S_printExpCashPromo	Print Cash Promo Expirations	true	Print expiration on cashable and promo vouchers
G2S_expireNonCash	Expire Non-Cashable	30 (days)	Default number of days before non-cashable vouchers expire
G2S_printExpNonCash	Print Non-Cashable Expirations	true	Print expiration on non-cashable vouchers

21.30.3.2 Example of G2S_voucherOptions Option

```
// Voucher options
<optionItem
    optionId = "G2S_voucherOptions"
    securityLevel = "G2S_operator"
    minSelections = "1"
    maxSelections = "1" >
    <optionParameters>
        <complexParameter
            paramId = "G2S_voucherParams"
            paramName = "Voucher Options"
            paramHelp = "Configuration parameters for this voucher device" >
            <integerParameter
                paramId = "G2S_idReaderId"
                paramName = "ID Reader to Use"
                paramHelp = "ID reader to use for this voucher device"
                canModLocal = "true"
                canModRemote = "true" />
            <booleanParameter
                paramId = "G2S_combineCashableOut"
                paramName = "Combine Cashable Credit Types."
                paramHelp = "Combine cashable and promo credit in a single voucher"
                canModLocal = "true"
                canModRemote = "true" />
            <booleanParameter
                paramId = "G2S_allowNonCashOut"
                paramName = "Allow Non-Cashable Out"
                paramHelp = "Allow vouchers for non-cashable credits"
                canModLocal = "true"
                canModRemote = "true" />
            <integerParameter
                paramId = "G2S_maxValIds"
                paramName = "Maximum Validation IDs"
                paramHelp = "Maximum validation IDs EGM may buffer"
                canModLocal = "true"
                canModRemote = "true"
                minIncl = "1" />
            <integerParameter
                paramId = "G2S_minLevelValIds"
                paramName = "Minimum Level for Validation IDs"
                paramHelp = "Minimum validation IDs EGM must maintain"
                canModLocal = "true"
                canModRemote = "true"
```

```
            minIncl = "0" />
        <integerParameter
            paramId = "G2S_valIdListRefresh"
            paramName = "Validation ID Refresh Time"
            paramHelp = "Maximum time before EGM requests validation ID list update."
            minIncl = "60000"
            canModLocal = "true"
            canModRemote = "true" />
        <integerParameter
            paramId = "G2S_valIdListLife"
            paramName = "Validation ID List Life"
            paramHelp = "Maximum life of validation IDs without host intervention."
            minIncl = "120000"
            canModLocal = "true"
            canModRemote = "true" />
        <integerParameter
            paramId = "G2S_voucherHoldTime"
            paramName = "Maximum Voucher Hold Time"
            paramHelp = "Maximum time EGM should escrow a voucher"
            canModLocal = "true"
            canModRemote = "true" />
        <booleanParameter
            paramId = "G2S_printOffLine"
            paramName = "Print Offline"
            paramHelp = "Allow vouchers to be printed while host offline"
            canModLocal = "true"
            canModRemote = "true" />
        <integerParameter
            paramId = "G2S_expireCashPromo"
            paramName = "Expire Days Cash Promo"
            paramHelp = "Number of days before cashable and promo vouchers expire."
            canModLocal = "true"
            canModRemote = "true"
            minIncl = "0" />
        <booleanParameter
            paramId = "G2S_printExpCashPromo"
            paramName = "Print Cash Promo Expirations"
            paramHelp = "Print expiration on cashable and promo vouchers"
            canModLocal = "true"
            canModRemote = "true" />
        <integerParameter
            paramId = "G2S_expireNonCash"
            paramName = "Expire Non-Cashable"
            paramHelp = "Default number of days before non-cashable vouchers expire."
            canModLocal = "true"
            canModRemote = "true"
            minIncl = "0" />
        <booleanParameter
            paramId = "G2S_printExpNonCash"
            paramName = "Print Non-Cashable Expirations"
            paramHelp = "Print expiration on non-cashable vouchers"
            canModLocal = "true"
            canModRemote = "true" />
    </complexParameter>
</optionParameters>
<optionCurrentValues>
    <complexValue paramId = "G2S_voucherParams" >
        <integerValue paramId = "G2S_idReaderId" > 54321 </integerValue>
        <booleanValue paramId = "G2S_combineCashableOut" > false </booleanValue>
        <booleanValue paramId = "G2S_allowNonCashOut" > true </booleanValue>
        <integerValue paramId = "G2S_maxValIds" > 15 </integerValue>
        <integerValue paramId = "G2S_minLevelValIds" > 10 </integerValue>
```

```
            <integerValue paramId = "G2S_valIdListRefresh" > 43200000 </integerValue>
            <integerValue paramId = "G2S_valIdListLife" > 86400000 </integerValue>
            <integerValue paramId = "G2S_voucherHoldTime" > 10000 </integerValue>
            <booleanValue paramId = "G2S_printOffLine" > true </booleanValue>
            <integerValue paramId = "G2S_expireCashPromo" > 90 </integerValue>
            <booleanValue paramId = "G2S_printExpCashPromo" > true </booleanValue>
            <integerValue paramId = "G2S_expireNonCash" > 30 </integerValue>
            <booleanValue paramId = "G2S_printExpNonCash" > true </booleanValue>
        </complexValue>
    </optionCurrentValues>
    <optionDefaultValues>
        <complexValue paramId = "G2S_voucherParams" >
            <integerValue paramId = "G2S_idReaderId" > 54321 </integerValue>
            <booleanValue paramId = "G2S_combineCashableOut" > false </booleanValue>
            <booleanValue paramId = "G2S_allowNonCashOut" > true </booleanValue>
            <integerValue paramId = "G2S_maxValIds" > 15 </integerValue>
            <integerValue paramId = "G2S_minLevelValIds" > 10 </integerValue>
            <integerValue paramId = "G2S_valIdListRefresh" > 43200000 </integerValue>
            <integerValue paramId = "G2S_valIdListLife" > 86400000 </integerValue>
            <integerValue paramId = "G2S_voucherHoldTime" > 10000 </integerValue>
            <booleanValue paramId = "G2S_printOffLine" > true </booleanValue>
            <integerValue paramId = "G2S_expireCashPromo" > 90 </integerValue>
            <booleanValue paramId = "G2S_printExpCashPromo" > true </booleanValue>
            <integerValue paramId = "G2S_expireNonCash" > 30 </integerValue>
            <booleanValue paramId = "G2S_printExpNonCash" > true </booleanValue>
        </complexValue>
    </optionDefaultValues>
</optionItem>
```

21.30.4 Voucher Text Fields Option Definitions

optionId	G2S_voucherTextFields
securityLevel	G2S_operator
minSelections	1
maxSelections	1
duplicates	n/a
paramKey	n/a
Parameter Type	Complex
paramId	G2S_textFields
paramName	Voucher Text Fields List
paramHelp	Text fields used by this voucher device

21.30.4.1 Voucher Text Fields Sub-Parameter Definitions

Table 21.79 Voucher Text Fields Sub-Parameter Definitions (Sheet 1 of 2)

paramId	paramName	Example	paramHelp
G2S_propName	Property Name	Penny Palace	Name of the property

Table 21.79 Voucher Text Fields Sub-Parameter Definitions (Sheet 2 of 2)

paramId	paramName	Example	paramHelp
G2S_propLine1	Property Line 1	777 Winner Way	Property address line 1
G2S_propLine2	Property Line 2	Luck Town, USA	Property address line 2
G2S_titleCash	Cashable Title	CASHOUT VOUCHER	Title printed on vouchers for cashable credits
G2S_titlePromo	Promotional Title	CASHOUT VOUCHER	Title printed on vouchers for promotional cashable credits
G2S_titleNonCash	Non-Cashable Title	PLAYABLE ONLY	Title printed on vouchers for non-cashable credits
G2S_titleLargeWin	Large Win Title	JACKPOT VOUCHER	Title printed on vouchers for wins greater than cabinetProfile.largeWinLimit
G2S_titleBonusCash	Bonus Cashable Title	CASHOUT VOUCHER	Title printed on bonus award vouchers for cashable credits
G2S_titleBonusPromo	Bonus Promotional Title	CASHOUT VOUCHER	Title printed on bonus award vouchers for promotional cashable credits
G2S_titleBonusNonCash	Bonus Non-Cashable Title	PLAYABLE ONLY	Title printed on bonus award vouchers for non-cashable credits
G2S_titleWatCash	WAT Cashable Title	CASHOUT VOUCHER	Title printed on WAT transfer vouchers for cashable credits
G2S_titleWatPromo	WAT Promotional Title	CASHOUT VOUCHER	Title printed on WAT transfer vouchers for promotional cashable credits
G2S_titleWatNonCash	WAT Non-Cashable Title	PLAYABLE ONLY	Title printed on WAT transfer vouchers for non-cashable credits

21.30.4.2 Example of G2S_voucherTextFields Option

```
// Voucher Text Field options
<optionItem
    optionId = "G2S_voucherTextFields"
    securityLevel = "G2S_operator"
    minSelections = "1"
    maxSelections = "1" >
    <optionParameters>
        <complexParameter
            paramId = "G2S_textFields"
            paramName = "Voucher Text Fields List"
            paramHelp = "Text fields used by this voucher device" >
            <stringParameter
                paramId = "G2S_propName"
                paramName = "Property Name"
                paramHelp = "Name of the property"
                canModLocal = "true"
                canModRemote = "true"
                minLen = "0"
                maxLen = "40"/>
            <stringParameter
                paramId = "G2S_propLine1"
```

```
    paramName = "Property Line 1"
    paramHelp = "Property address line 1"
    canModLocal = "true"
    canModRemote = "true"
    minLen = "0"
    maxLen = "40"/>
<stringParameter
    paramId = "G2S_propLine2"
    paramName = "Property Line 2"
    paramHelp = "Property address line 2"
    canModLocal = "true"
    canModRemote = "true"
    minLen = "0"
    maxLen = "40"/>
<stringParameter
    paramId = "G2S_titleCash"
    paramName = "Cashable Title"
    paramHelp = "Title printed on vouchers for cashable credits"
    canModLocal = "true"
    canModRemote = "true"
    minLen = "0"
    maxLen = "16"/>
<stringParameter
    paramId = "G2S_titlePromo"
    paramName = "Promotional Title"
    paramHelp = "Title printed on vouchers for promotional cashable credits."
    canModLocal = "true"
    canModRemote = "true"
    minLen = "0"
    maxLen = "16"/>
<stringParameter
    paramId = "G2S_titleNonCash"
    paramName = "Non-Cashable Title"
    paramHelp = "Title printed on vouchers for non-cashable credits"
    canModLocal = "true"
    canModRemote = "true"
    minLen = "0"
    maxLen = "16"/>
<stringParameter
    paramId = "G2S_titleLargeWin"
    paramName = "Large Win Title"
    paramHelp = "Title printed on vouchers for large wins (jackpots)"
    canModLocal = "true"
    canModRemote = "true"
    minLen = "0"
    maxLen = "16"/>
<stringParameter
    paramId = "G2S_titleBonusCash"
    paramName = "Bonus Cashable Title"
    paramHelp = "Title printed on bonus award vouchers for cashable credits."
    canModLocal = "true"
    canModRemote = "true"
    minLen = "0"
    maxLen = "16"/>
<stringParameter
    paramId = "G2S_titleBonusPromo"
    paramName = "Bonus Promotional Title"
    paramHelp = "Title printed on bonus award vouchers for promotional
               cashable credits"
    canModLocal = "true"
    canModRemote = "true"
    minLen = "0"
```

```
            maxLen = "16"/>
        <stringParameter
            paramId = "G2S_titleBonusNonCash"
            paramName = "Bonus Non-Cashable Title"
            paramHelp = "Title printed on bonus award vouchers for non-cashable credits"
            canModLocal = "true"
            canModRemote = "true"
            minLen = "0"
            maxLen = "16"/>
        <stringParameter
            paramId = "G2S_titleWatCash"
            paramName = "WAT Cashable Title"
            paramHelp = "Title printed on WAT transfer vouchers for cashable credits"
            canModLocal = "true"
            canModRemote = "true"
            minLen = "0"
            maxLen = "16"/>
        <stringParameter
            paramId = "G2S_titleWatPromo"
            paramName = "WAT Promotional Title"
            paramHelp = "Title printed on WAT transfer vouchers for promotional cashable
                credits"
            canModLocal = "true"
            canModRemote = "true"
            minLen = "0"
            maxLen = "16"/>
        <stringParameter
            paramId = "G2S_titleWatNonCash"
            paramName = "WAT Non-Cashable Title"
            paramHelp = "Title printed on WAT transfer vouchers for non-cashable
                credits."
            canModLocal = "true"
            canModRemote = "true"
            minLen = "0"
            maxLen = "16"/>
        </complexParameter>
    </optionParameters>
<optionCurrentValues>
    <complexValue paramId = "G2S_textFields" >
        <stringValue paramId = "G2S_propName" > Penny Palace </stringValue>
        <stringValue paramId = "G2S_propLine1" > 777 Winner Way </stringValue>
        <stringValue paramId = "G2S_propLine2" > Lucky Town, USA </stringValue>
        <stringValue paramId = "G2S_titleCash" > CASHOUT VOUCHER </stringValue>
        <stringValue paramId = "G2S_titlePromo" > CASHOUT VOUCHER </stringValue>
        <stringValue paramId = "G2S_titleNonCash" > PLAYABLE ONLY </stringValue>
        <stringValue paramId = "G2S_titleLargeWin" > JACKPOT VOUCHER </stringValue>
        <stringValue paramId = "G2S_titleBonusCash" > CASHOUT VOUCHER </stringValue>
        <stringValue paramId = "G2S_titleBonusPromo" > CASHOUT VOUCHER </stringValue>
        <stringValue paramId = "G2S_titleBonusNonCash" > PLAYABLE ONLY </stringValue>
        <stringValue paramId = "G2S_titleWatCash" > CASHOUT VOUCHER </stringValue>
        <stringValue paramId = "G2S_titleWatPromo" > CASHOUT VOUCHER </stringValue>
        <stringValue paramId = "G2S_titleWatNonCash" > PLAYABLE ONLY </stringValue>
    </complexValue>
</optionCurrentValues>
<optionDefaultValues>
    <complexValue paramId = "G2S_textFields" >
        <stringValue paramId = "G2S_propName" > </stringValue>
        <stringValue paramId = "G2S_propLine1" > </stringValue>
        <stringValue paramId = "G2S_propLine2" > </stringValue>
        <stringValue paramId = "G2S_titleCash" > </stringValue>
        <stringValue paramId = "G2S_titlePromo" > </stringValue>
        <stringValue paramId = "G2S_titleNonCash" > </stringValue>
```

```
            <stringValue paramId = "G2S_titleLargeWin" > </stringValue>
            <stringValue paramId = "G2S_titleBonusCash" > </stringValue>
            <stringValue paramId = "G2S_titleBonusPromo" > </stringValue>
            <stringValue paramId = "G2S_titleBonusNonCash" > </stringValue>
            <stringValue paramId = "G2S_titleWatCash" > </stringValue>
            <stringValue paramId = "G2S_titleWatPromo" > </stringValue>
            <stringValue paramId = "G2S_titleWatNonCash" > </stringValue>
        </complexValue>
    </optionDefaultValues>
</optionItem>
```

21.30.5 Host Aware Backwards Compatibility

Extension in v3.0: g2s3

The host MUST be aware that an EGM may support the `G2S_protocolOptions3` option, as well as the `G2S_configComplete` parameter, even though they have been deprecated. Therefore, the host must not assume that the EGM does not support the `G2S_protocolOptions3` option or the `G2S_configComplete` parameter; hosts MUST support the `G2S_protocolOptions3` option when reported by the EGM through the `optionConfig` class.

21.30.6 Voucher Options 2 Definitions

Extension in v2.0.0: g2s1

optionId	G2S_voucherOptions2
securityLevel	G2S_operator
minSelections	1
maxSelections	1
duplicates	n/a
paramKey	n/a
Parameter Type	Complex
paramId	G2S_voucherParams2
paramName	Additional voucher Options
paramHelp	Additional configuration parameters for this voucher device

21.30.6.1 Voucher Options 2 Sub-Parameter Definitions

Table 21.80 Voucher Options 2 Sub-Parameter Definitions (Sheet 1 of 2)

paramId	paramName	Example	paramHelp
G2S_allowVoucherIssue	Allow Validation Data	true	Indicates whether the voucher device is allowed to request validation data; does not (directly) control voucher issuance

Table 21.80 Voucher Options 2 Sub-Parameter Definitions (Sheet 2 of 2)

paramId	paramName	Example	paramHelp
G2S_allowVoucherRedeem	Voucher Redemption Allowed	true	Indicates whether the voucher device is allowed to redeem vouchers
G2S_printNonCashOffLine	Print Non-Cashable Vouchers When Offline	true	Indicates whether vouchers for non-cashable credits may be printed while offline

21.30.6.2 Example of G2S_voucherOptions2 Option

```
// Voucher Options2
<optionItem
    optionId = "G2S_voucherOptions2"
    securityLevel = "G2S_operator"
    minSelections = "1"
    maxSelections = "1" >
    <optionParameters>
        <complexParameter
            paramId = "G2S_voucherParams2"
            paramName = "Additional Voucher Options"
            paramHelp = "Additional configuration parameters for this voucher device" >
        <booleanParameter
            paramId = "G2S_allowVoucherIssue"
            paramName = "Allow Voucher Issuance"
            paramHelp = "Indicates whether validation data may be requested."
            canModLocal = "true"
            canModRemote = "true" />
        <booleanParameter
        paramId = "G2S_allowVoucherRedeem"
            paramName = "Allow Voucher Redemption"
            paramHelp = "Indicates whether vouchers may be redeemed."
            canModLocal = "true"
            canModRemote = "true" />
        <booleanParameter
            paramId = "G2S_printNonCashOffLine"
            paramName = "Print Non-Cashable Voucher When Offline"
            paramHelp = "Indicates whether vouchers for non-cashable credits may be printed
            while offline."
            canModLocal = "true"
            canModRemote = "true" />
        </complexParameter>
    </optionParameters>
    <optionCurrentValues>
        <complexValue paramId = "G2S_voucherParams2" >
            <booleanValue paramId = "G2S_allowVoucherIssue" > true </booleanValue>
            <booleanValue paramId = "G2S_allowVoucherRedeem" > true </booleanValue>
            <booleanValue paramId = "G2S_printNonCashOffLine" > true </booleanValue>
        </complexValue>
    </optionCurrentValues>
    <optionDefaultValues>
        <complexValue paramId = "G2S_voucherParams2" >
            <booleanValue paramId = "G2S_allowVoucherIssue" > true </booleanValue>
            <booleanValue paramId = "G2S_allowVoucherRedeem" > true </booleanValue>
            <booleanValue paramId = "G2S_printNonCashOffLine" > true </booleanValue>
        </complexValue>
    </optionDefaultValues>
```

```
</optionItem>
```

21.30.7 Voucher Limits Option Definitions

Extension in v2.0.0: g2s1

optionId	G2S_voucherLimits
securityLevel	G2S_administrator
minSelections	1
maxSelections	1
duplicates	n/a
paramKey	n/a
Parameter Type	Complex
paramId	G2S_limitParams
paramName	Voucher Limit Parameters
paramHelp	Maximum values for printed vouchers

21.30.7.1 Voucher Limits Sub-Parameter Definitions

Table 21.81 Voucher Limits Sub-Parameter Definitions

paramId	paramName	Example	paramHelp
G2S_maxOnLinePayOut	Maximum Online Voucher	150000000	Maximum value of an online voucher (in millicents)
G2S_maxOffLinePayOut	Maximum Offline Voucher	50000000	Maximum value of an offline voucher (in millicents)

21.30.7.2 Example of G2S_voucherLimits Option

```
// Voucher Limit Options
<optionItem
    optionId = "G2S_voucherLimits"
    securityLevel = "G2S_administrator"
    minSelections = "1"
    maxSelections = "1" >
    <optionParameters>
        <complexParameter
            paramId = "G2S_limitParams"
            paramName = "Voucher Limit Parameters"
            paramHelp = "Maximum values for printed vouchers" >
            <integerParameter
                paramId = "G2S_maxOnLinePayOut"
                paramName = "Maximum Online Voucher"
                paramHelp = " Maximum value of an online voucher (in millicents)"
                canModLocal = "true"
                canModRemote = "true" />
            <integerParameter
```

```
            paramId = "G2S_maxOffLinePayOut"
                paramName = "Maximum Offline Voucher"
                paramHelp = " Maximum value of an offline voucher (in millicents)"
                canModLocal = "true"
                canModRemote = "true" />
        </complexParameter>
    </optionParameters>
    <optionCurrentValues>
        <complexValue paramId = "G2S_limitParams" >
            <integerValue paramId = "G2S_maxOnLinePayOut" > 150000000 </integerValue>
            <integerValue paramId = "G2S_maxOffLinePayOut" > 50000000 </integerValue>
        </complexValue>
    </optionCurrentValues>
    <optionDefaultValues>
        <complexValue paramId = "G2S_limitParams" >
            <integerValue paramId = "G2S_maxOnLinePayOut" > 150000000 </integerValue>
            <integerValue paramId = "G2S_maxOffLinePayOut" > 50000000 </integerValue>
        </complexValue>
    </optionDefaultValues>
</optionItem>
```

21.30.8 Use Player ID Reader Option Definition

Extension in v3.0: g2s3

optionId	G2S_usePlayerIdReaderOptions
securityLevel	G2S_operator
minSelections	1
maxSelections	1
duplicates	n/a
paramKey	n/a
Parameter Type	Complex
paramId	G2S_usePlayerIdReaderParams
paramName	Use Player ID Reader
paramHelp	Parameters associated with Use-Player-ID-Reader option.

21.30.8.1 Use Player ID Reader Sub-Parameter Definitions

Table 21.82 Use Player ID Reader Sub-Parameter Definitions

paramId	paramName	Example	paramHelp
G2S_usePlayerIdReader	Use Player ID Reader	true	Indicates whether the ID reader associated with the currently active player session should be used.

21.30.8.2 Example of G2S_usePlayerIdReaderOptions Option

```
// Use Player ID Reader Option
<optionItem
    optionId = "G2S_usePlayerIdReaderOptions"
    securityLevel = "G2S_operator"
    minSelections = "1"
    maxSelections = "1" >
    <optionParameters>
        <complexParameter
            paramId = "G2S_usePlayerIdReaderParams"
            paramName = "Use Player ID Reader Parameters"
            paramHelp = "Parameters associated with Use-Player-ID-Reader option." >
            <booleanParameter
                paramId = "G2S_usePlayerIdReader"
                paramName = "Use Player ID Reader"
                paramHelp = "Indicates whether the ID reader associated with the currently
                active player session should be used."
                canModLocal = "true"
                canModRemote = "true" />
        </complexParameter>
    </optionParameters>
    <optionCurrentValues>
        <complexValue paramId = "G2S_usePlayerIdReaderParams" >
            <booleanValue paramId = "G2S_usePlayerIdReader" > true </booleanValue>
        </complexValue>
    </optionCurrentValues>
    <optionDefaultValues>
        <complexValue paramId = "G2S_usePlayerIdReaderParams" >
            <booleanValue paramId = "G2S_usePlayerIdReader" > false </booleanValue>
        </complexValue>
    </optionDefaultValues>
</optionItem>
```

21.30.9 Offline Handpay Voucher Option Definitions

Extension in v3.0: g2sVSO1

optionId	G2S_handpayVoucherOptions
securityLevel	G2S_operator
minSelections	1
maxSelections	1
duplicates	n/a
paramKey	n/a
Parameter Type	Complex
paramId	G2S_handpayVoucherParams
paramName	Handpay Voucher Parameters
paramHelp	Parameters for enabling handpay vouchers.

21.30.9.1 Offline Handpay Voucher Options Sub-Parameter Definitions

Table 21.83 Offline Handpay Voucher Sub-Parameter Definitions

paramId	paramName	Example	paramHelp
G2S_enableHandpayVoucher	Enable offline handpay vouchers	false	Indicates whether offline handpay vouchers are enabled.
G2S_titleHandpayVoucher	Offline handpay voucher title	HANDPAY VOUCHER	Title printed on offline handpay vouchers.

21.30.9.2 Example of G2S_handpayVoucherOptions Option

```
// Offline Handpay Voucher Options
<optionItem
    optionId = "G2S_handpayVoucherOptions"
    securityLevel = "G2S_operator"
    minSelections = "1"
    maxSelections = "1" >
    <optionParameters>
        <complexParameter
            paramId = "G2S_handpayVoucherParams"
            paramName = "Offline Handpay Voucher Parameters"
            paramHelp = "Parameters for enabling offline handpay vouchers" >
            <booleanParameter
                paramId = "G2S_enableHandpayVoucher"
                paramName = "Enable offline handpay vouchers"
                paramHelp = "Indicates offline handpay vouchers are enabled"
                canModLocal = "true"
                canModRemote = "true" />
            <stringParameter
                paramId = "G2S_titleHandpayVoucher"
                paramName = "Offline handpay voucher title"
                paramHelp = "Title printed on offline handpay vouchers"
                canModLocal = "true"
                canModRemote = "true"
                minLen = "0"
                maxLen = "16" />
        </complexParameter>
    </optionParameters>
    <optionCurrentValues>
        <complexValue paramId = "G2S_handpayVoucherParams" >
            <booleanValue paramId = "G2S_enableHandpayVoucher" > false </booleanValue>
            <stringValue paramId = "G2S_titleHandpayVoucher" > HANDPAY VOUCHER </stringValue>
        </complexValue>
    </optionCurrentValues>
    <optionDefaultValues>
        <complexValue paramId = "G2S_handpayVoucherParams" >
            <booleanValue paramId = "G2S_enableHandpayVoucher" > false </booleanValue>
            <stringValue paramId = "G2S_titleHandpayVoucher" > HANDPAY VOUCHER </stringValue>
        </complexValue>
    </optionDefaultValues>
</optionItem>
```

21.31 Certification Requirements

The following tables identify the functional groups for this class in which a product may be certified. The tables also identify the configuration options that must be implemented for a product to be certified. See Section 1.28, Certification Requirements for more details.

Table 21.84 voucher Class Functional Groups

Functional Group	Associated Commands

Core Voucher Functionality (g2s)	`setVoucherState`
	`voucherStatus`
	`setVoucherLockOut`
	`getVoucherStatus`
	`getVoucherProfile`
	`voucherProfile`
	`getVoucherLogStatus`
	`voucherLogStatus`
	`getVoucherLog`
	`voucherLogList`

Issue Voucher Support (g2s)	`getValidationData`
	`validationData`
	`issueVoucher`
	`issueVoucherAck`

Redeem Voucher Support (g2s)	`redeemVoucher`
	`authorizeVoucher`
	`commitVoucher`
	`commitVoucherAck`

Validation System Offline Support (g2sVSO)	None
Offline Handpay Voucher Support (g2sVSO1)	None

Table 21.85 voucher Class Configuration Options

Element	Option	Configuration
`voucherProfile`	`configurationId`	Required.
	`restartStatus`	Required.
	`useDefaultConfig`	Optional; SHOULD be prohibited and SHOULD be set to false.
	`requiredForPlay`	Required.
	`minLogEntries`	Optional.
	`timeToLive`	Required.
	`idReaderId`	Required.
	`combineCashableOut`	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.

Table 21.85 voucher Class Configuration Options

Element	Option	Configuration
	allowNonCashOut	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	maxValIds	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	minLevelValIds	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	valIdListRefresh	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	valIdListLife	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	voucherHoldTime	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	printOffLine	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	expireCashPromo	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	printExpCashPromo	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	expireNonCash	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	printExpNonCash	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	propName	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	propLine1	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.

Table 21.85 voucher Class Configuration Options

Element	Option	Configuration
	`propLine2`	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	`titleCash`	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	`titlePromo`	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	`titleNonCash`	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	`titleLargeWin`	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	`titleBonusCash`	Conditional; required when Issue Voucher Support and Core Bonus Functionality are to be certified; prohibited when Issue Voucher Support or Core Bonus Functionality are not implemented.
	`titleBonusPromo`	Conditional; required when Issue Voucher Support and Core Bonus Functionality are to be certified; prohibited when Issue Voucher Support or Core Bonus Functionality are not implemented.
	`titleBonusNonCash`	Conditional; required when Issue Voucher Support and Core Bonus Functionality are to be certified; prohibited when Issue Voucher Support or Core Bonus Functionality are not implemented.
	`titleWatCash`	Conditional; required when Issue Voucher Support and Core Wagering Account Functionality are to be certified; prohibited when Issue Voucher Support or Core Wagering Account Functionality are not implemented.
	`titleWatPromo`	Conditional; required when Issue Voucher Support and Core Wagering Account Functionality are to be certified; prohibited when Issue Voucher Support or Core Wagering Account Functionality are not implemented.

Table 21.85 voucher Class Configuration Options

Element	Option	Configuration
	titleWatNonCash	Conditional; required when Issue Voucher Support and Core Wagering Account Functionality are to be certified; prohibited when Issue Voucher Support or Core Wagering Account Functionality are not implemented.
	configDateTime	Optional.
	configComplete	Optional; SHOULD be prohibited and SHOULD be set to true.
	allowVoucherIssue	Conditional; required when Issue Voucher Support is to be certified; prohibited, and MUST be set to false, when Issue Voucher Support is not implemented.
	allowVoucherRedeem	Conditional; required when Redeem Voucher Support is to be certified; prohibited, and MUST be set to false, when Redeem Voucher Support is not implemented.
	maxOnLinePayOut	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	maxOffLinePayOut	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	printNonCashOffLine	Conditional; required when Issue Voucher Support is to be certified; prohibited when Issue Voucher Support is not implemented.
	cashOutToVoucher	Optional.
	redeemPrefix	Optional.
	usePlayerIdReader	Required.
	noAckTimer	Conditional; required when Validation System Offline Support is to be certified; prohibited when Validation System Offline Support is not implemented.
	enableHandpayVoucher	Conditional; required when Offline Handpay Voucher Support is to be certified; prohibited, and MUST be set to false, when Offline Handpay Voucher Support is not implemented.
	titleHandpayVoucher	Conditional; required when Offline Handpay Voucher Support is to be certified; prohibited when Offline Handpay Voucher Support is not implemented.