NEW CLASS

Chapter 40

Look Inside

playerVoucher Class

Extension in v2.0: pvc

NEW CLASS

40.1 Introduction

The `playerVoucher` class is used to manage and report information about the issuance and redemption of payment vouchers by end-clients. Payment vouchers are sometimes referred to as tickets or coupons. For example, the `playerVoucher` class can be used to record the issuance of a voucher by an end-client. Subsequently, the `playerVoucher` class can be used to authorize the redemption of the voucher by the same end-client or a different end-client. The `playerVoucher` class can also be used to manage the configuration information used by end-clients when performing voucher processing operations.

This functionality maps directly to similar functionality within the G2S protocol allowing a central system to easily manage voucher processing operations across a series of edge-servers that are using the G2S protocol to manage end-client operations.

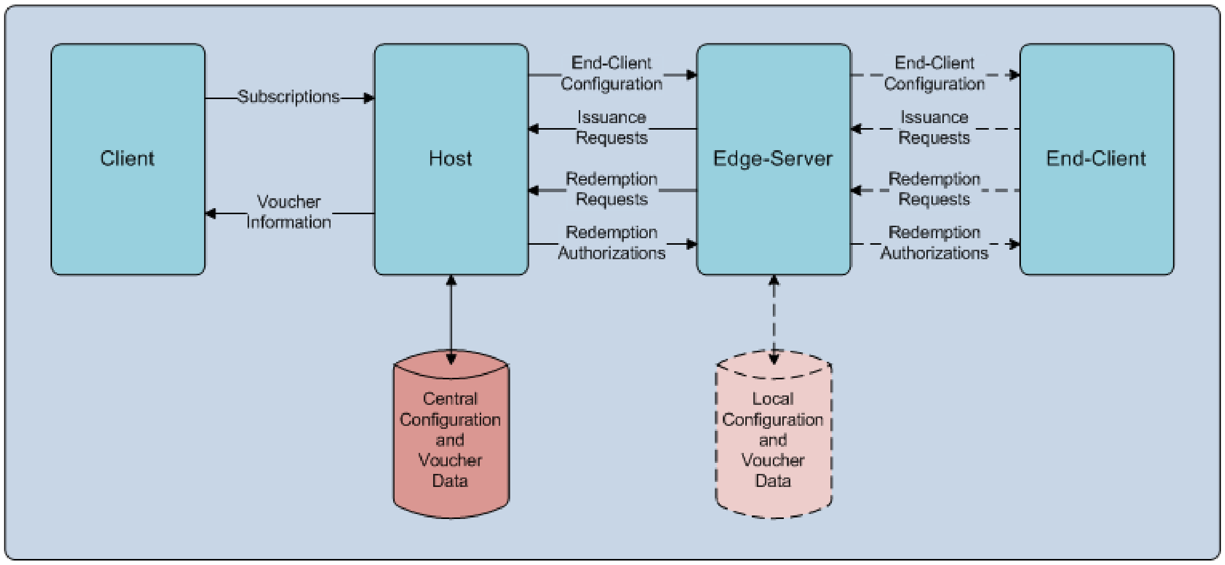
The `playerVoucher` class focuses on the actual issuance and redemption of vouchers by end-clients. The `playerInfo` class focuses on player registration and player management. See Chapter 30, playerInfo Class, for more information on those topics.

40.1.1 Terminology

Within this class, the following definitions are used when referring to hosts, clients, edge-servers, and end-clients.

Host	A central system that provides access to the database of record for voucher information. Configuration information is sent by the host to edge-servers that use the information to manage voucher processing operations on end-clients. Voucher issuance and redemption requests are gathered by edge-servers from end-clients and then sent to the host for authorization, storage, and reporting purposes. Voucher information may be sent by the host to clients, such as data warehouses.
Client	A system, such as a data warehouse, that requests voucher information from the host and/or sets subscriptions for voucher information with the host. Based on those subscriptions, voucher information may be sent from the host to the client.
Edge-Server	A system that provides voucher management services for end-clients. Configuration information for end-clients may be received by the edge-server from the host. Voucher issuance and redemption requests may be received by the edge-server from end-clients and then sent by the edge-server to the host for authorization. An optional local database of configuration and voucher information may be maintained by the edge-server.
End-Client	A system that generates voucher issuance and redemption requests. The end-client sends the requests to the edge-server for authorization. The end-client may receive configuration information from the edge-server. The commands used to communicate between the end-client and the edge-server are outside of the scope of the `playerVoucher` class. Another protocol, such as G2S, may be used for that purpose.

Figure 40.1 System Roles



40.1.2 End-Client Configuration Sequence Diagrams

The following sequence diagrams demonstrate how the `playerVoucher` class is intended to be used to manage the voucher configuration information for end-clients, such as EGMs, through an edge-server. For demonstration purposes, the edge-server is managing a series of end-clients using the G2S protocol.

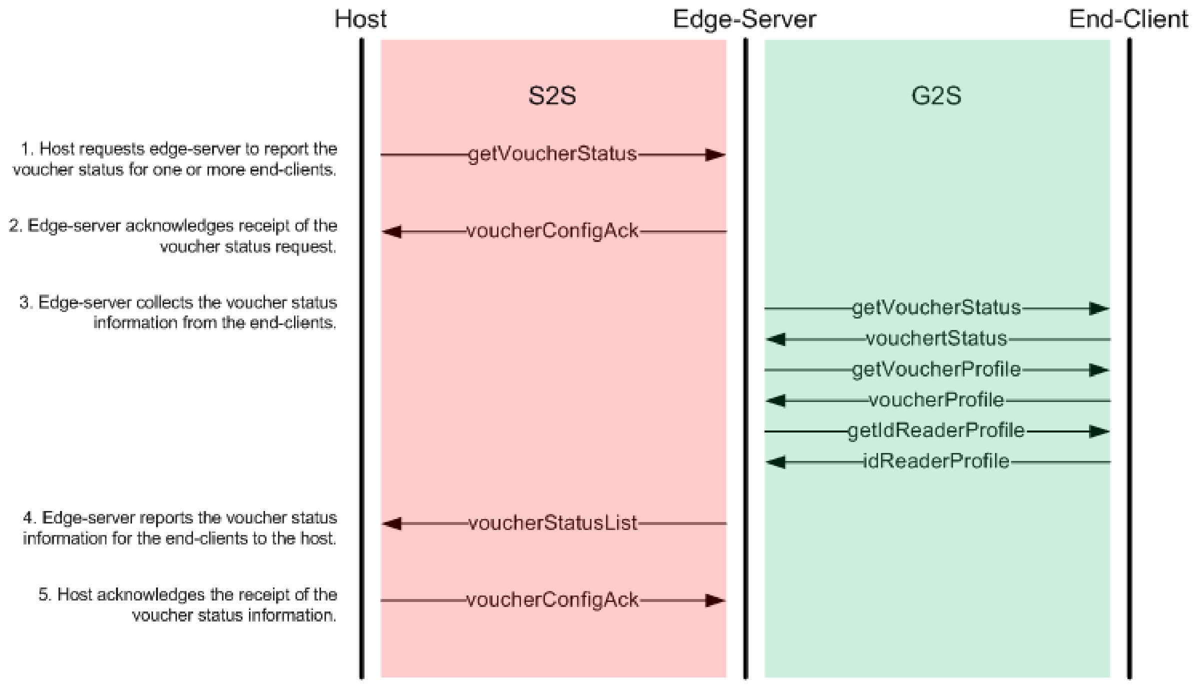
40.1.3 Voucher Status Request by Host

The following diagram demonstrates the behavior expected when the `playerVoucher` class is used by the host to request current voucher status information for a series of end-clients from an edge-server.

1. The host requests that the edge-server report the voucher status for one or more end-clients.
2. The edge-server acknowledges the receipt of the request.
3. The edge-server collects the voucher status information from the end-clients, as necessary. The voucher status information may be derived from information stored on the edge-server or it may be gathered from the end-clients.
4. The edge-server reports the voucher status information for the end-clients to the host. One or more commands may be used by the edge-server to report the information.
5. The host acknowledges the receipt of the voucher status information for the end-clients. When reported across multiple commands, multiple acknowledgements are sent.

NEW CLASS

Figure 40.2 Voucher Status Request by Host

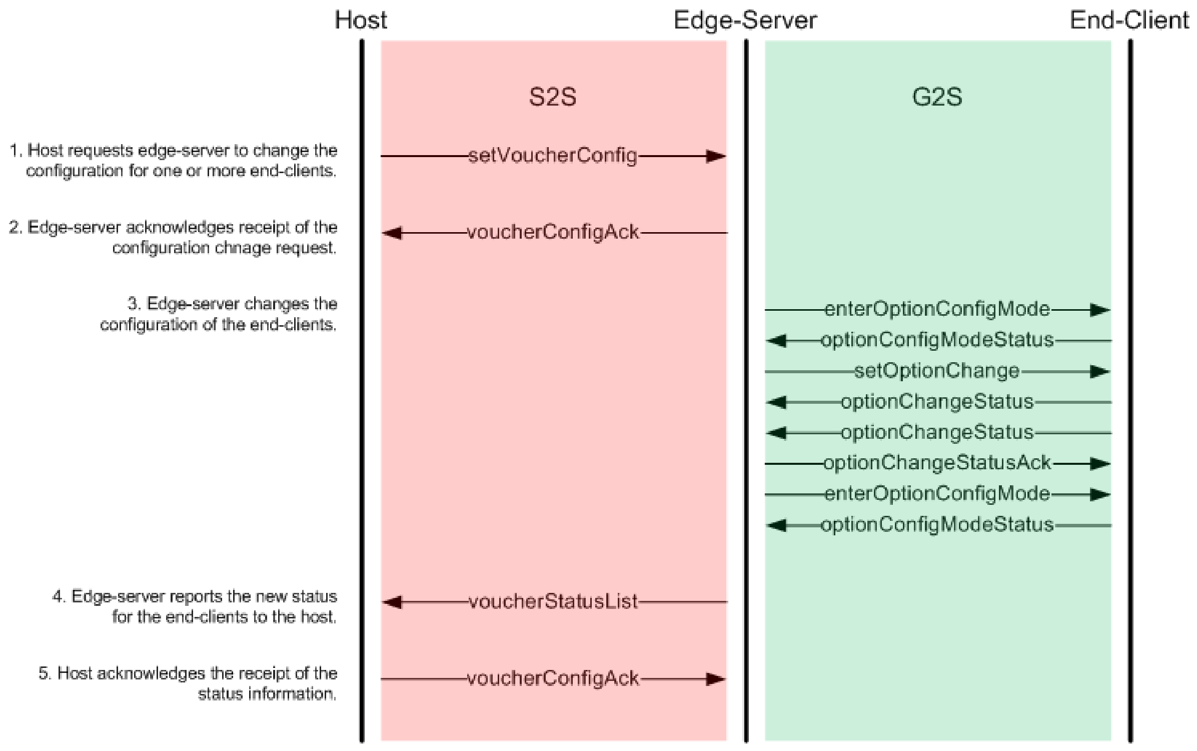

Host
Edge-Server
End-Client
S2S
G2S
1. Host requests edge-server to report the voucher status for one or more end-clients.
getVoucherStatus
2. Edge-server acknowledges receipt of the voucher status request.
voucherConfigAck
3. Edge-server collects the voucher status information from the end-clients.
getVoucherStatus
vouchertStatus
getVoucherProfile
voucherProfile
getIdReaderProfile
idReaderProfile
4. Edge-server reports the voucher status information for the end-clients to the host.
voucherStatusList
5. Host acknowledges the receipt of the voucher status information.
voucherConfigAck

40.1.4 Voucher Configuration Change by Host

The following diagram demonstrates the behavior expected when the `playerVoucher` class is used by the host to set the voucher configuration for a series of end-clients through an edge-server.

1. The host requests that the edge-server change the voucher configuration for one or more end-clients.
2. The edge-server acknowledges the receipt of the voucher configuration request.
3. The edge-server applies the voucher configuration changes to the end-clients.
4. The edge-server reports the new voucher status information for the end-clients to the host. One or more commands may be used by the edge-server to report the information.
5. The host acknowledges the receipt of the voucher status information for the end-clients. When reported across multiple commands, multiple acknowledgements are sent.

Figure 40.3 Voucher Configuration Change by Host


Host
Edge-Server
End-Client
S2S
G2S
1. Host requests edge-server to change the configuration for one or more end-clients.
setVoucherConfig
2. Edge-server acknowledges receipt of the configuration chnage request.
voucherConfigAck
3. Edge-server changes the configuration of the end-clients.
enterOptionConfigMode
optionConfigModeStatus
setOptionChange
optionChangeStatus
optionChangeStatus
optionChangeStatusAck
enterOptionConfigMode
optionConfigModeStatus
4. Edge-server reports the new status for the end-clients to the host.
voucherStatusList
5. Host acknowledges the receipt of the status information.
voucherConfigAck

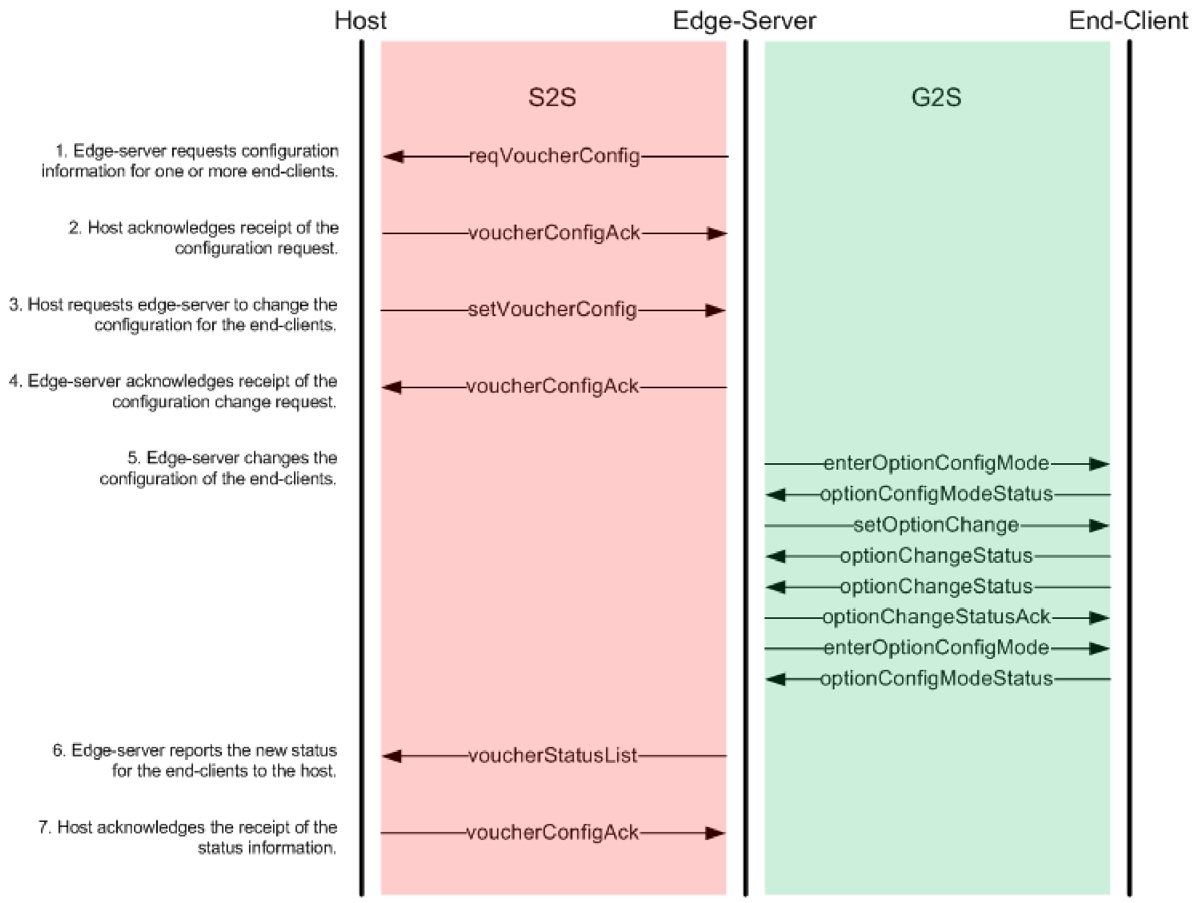
40.1.5 Voucher Configuration Request by Edge-Server

The following diagram demonstrates the behavior expected when the `playerVoucher` class is used by an edge-server to request the current voucher configuration information for a series of end-clients from the host.

1. The edge-server requests that the host send the voucher configuration for one or more end-clients.
2. The host acknowledges receipt of the voucher configuration request.
3. The host requests that the edge-server change the voucher configuration for the end-clients. One or more commands may be used by the host to request the changes.
4. The edge-server acknowledges the receipt of the voucher configuration request. When requested across multiple commands, multiple acknowledgements are sent.
5. The edge-server applies the voucher configuration changes, as necessary, to the end-clients.
6. The edge-server reports the new voucher status for the end-clients to the host. One or more commands may be used by the edge-server to report the information.
7. The host acknowledges the receipt of the voucher status information for the end-clients. When reported across multiple commands, multiple acknowledgements are sent.

NEW CLASS

Figure 40.4 Voucher Configuration Request by Edge-Server


Host
Edge-Server
End-Client
S2S
G2S
1. Edge-server requests configuration information for one or more end-clients.
reqVoucherConfig
2. Host acknowledges receipt of the configuration request.
voucherConfigAck
3. Host requests edge-server to change the configuration for the end-clients.
setVoucherConfig
4. Edge-server acknowledges receipt of the configuration change request.
voucherConfigAck
5. Edge-server changes the configuration of the end-clients.
enterOptionConfigMode
optionConfigModeStatus
setOptionChange
optionChangeStatus
optionChangeStatus
optionChangeStatusAck
enterOptionConfigMode
optionConfigModeStatus
6. Edge-server reports the new status for the end-clients to the host.
voucherStatusList
7. Host acknowledges the receipt of the status information.
voucherConfigAck

40.1.6 Voucher Issuance and Redemption Sequence Diagrams

The following sequence diagrams demonstrate how the `playerVoucher` class is intended to be used to manage the voucher issuance and redemption requests by end-clients, such as EGMs, through an edge-server. For demonstration purposes, the edge-server is managing a series of end-clients using the G2S protocol.

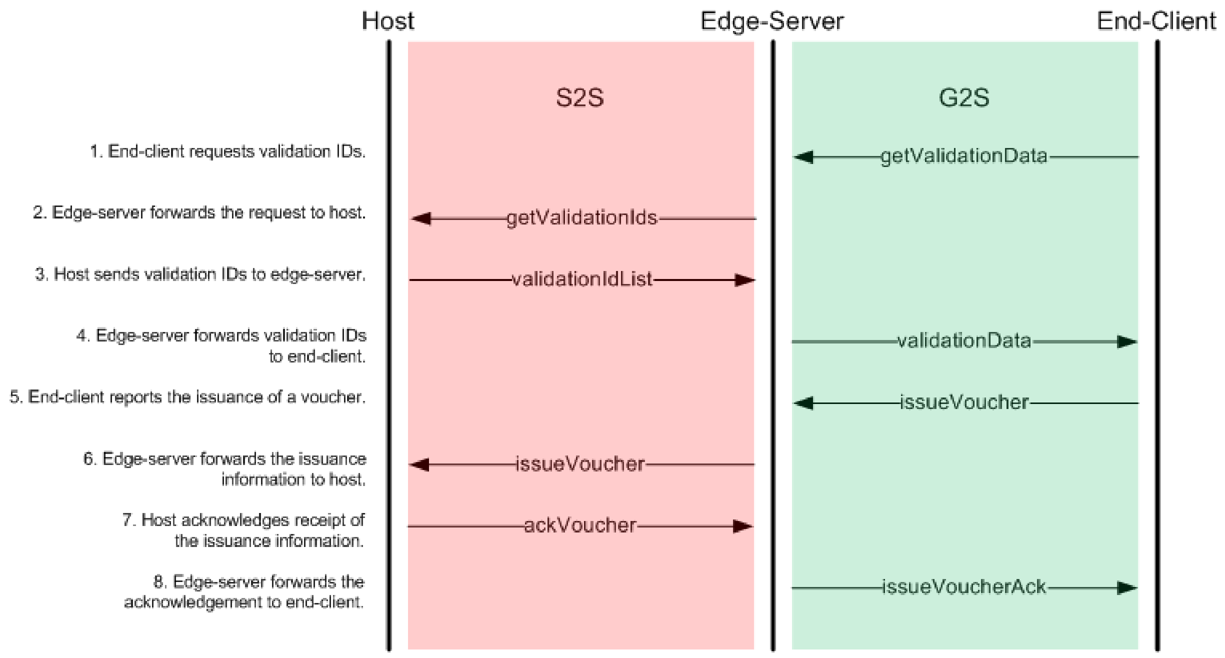
40.1.7 Voucher Issuance Request by End-Client

The following diagram demonstrates the behavior expected when the `playerVoucher` class is used by an edge-server to request voucher validation IDs and report the issuance of a voucher by an end-client.

1. The end-client sends a request for validation IDs to the edge-server.
2. The edge-server forwards the request to the host.
3. The host sends one or more validation IDs to the edge-server.
4. The edge-server forwards the validation IDs to the end-client.
5. The end-client reports the issuance of a voucher to the edge-server.
6. The edge-server forwards the voucher issuance information to the host.
7. The host acknowledges receipt of the voucher issuance information from the edge-server.

8. The edge-server forwards the acknowledgement to the end-client.

Figure 40.5 Voucher Issuance Request by End-Client


Host
Edge-Server
End-Client
S2S
G2S
1. End-client requests validation IDs.
2. Edge-server forwards the request to host.
3. Host sends validation IDs to edge-server.
4. Edge-server forwards validation IDs to end-client.
5. End-client reports the issuance of a voucher.
6. Edge-server forwards the issuance information to host.
7. Host acknowledges receipt of the issuance information.
8. Edge-server forwards the acknowledgement to end-client.
getValidationData
getValidationIds
validationIdList
validationData
issueVoucher
issueVoucher
ackVoucher
issueVoucherAck

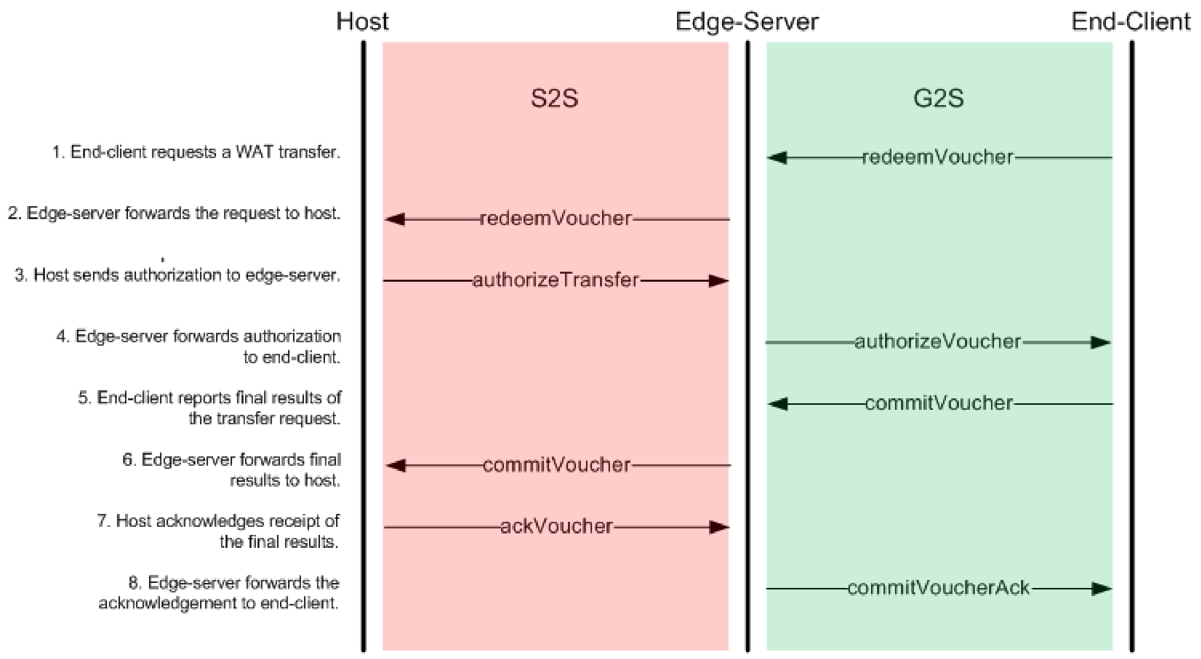
40.1.8 Voucher Redemption Request by End-Client

The following diagram demonstrates the behavior expected when the `playerVoucher` class is used to request redemption of a voucher by an end-client.

1. The end-client sends a voucher redemption request to the edge-server.
2. The edge-server forwards the request to the host.
3. The host authorizes the request from the edge-server.
4. The edge-server forwards the authorization to the end-client.
5. The end-client reports the final result of the voucher redemption request to the edge-server.
6. The edge-server forwards the final result to the host.
7. The host acknowledges receipt of the final result from the edge-server.
8. The edge-server forwards the acknowledgement to the end-client.

NEW CLASS

Figure 40.6 Voucher Redemption Request by End-Client


Host
Edge-Server
End-Client
S2S
G2S
1. End-client requests a WAT transfer.
redeemVoucher
2. Edge-server forwards the request to host.
redeemVoucher
3. Host sends authorization to edge-server.
authorizeTransfer
4. Edge-server forwards authorization to end-client.
authorizeVoucher
5. End-client reports final results of the transfer request.
commitVoucher
6. Edge-server forwards final results to host.
commitVoucher
7. Host acknowledges receipt of the final results.
ackVoucher
8. Edge-server forwards the acknowledgement to end-client.
commitVoucherAck

40.1.9 Voucher States

A voucher transaction will transition through a series of states while it is being processed by an end-client. The current state of the transaction is reported in the `voucherState` attribute of commands used to report voucher transaction activity.

When a voucher is issued, it transitions through two states.

- Upon issuance of the voucher, the state of the voucher transaction is set to `S2S_issueSent`.
- After the issuance has been acknowledged, the state of the voucher transaction is set to `S2S_issueAcked`.

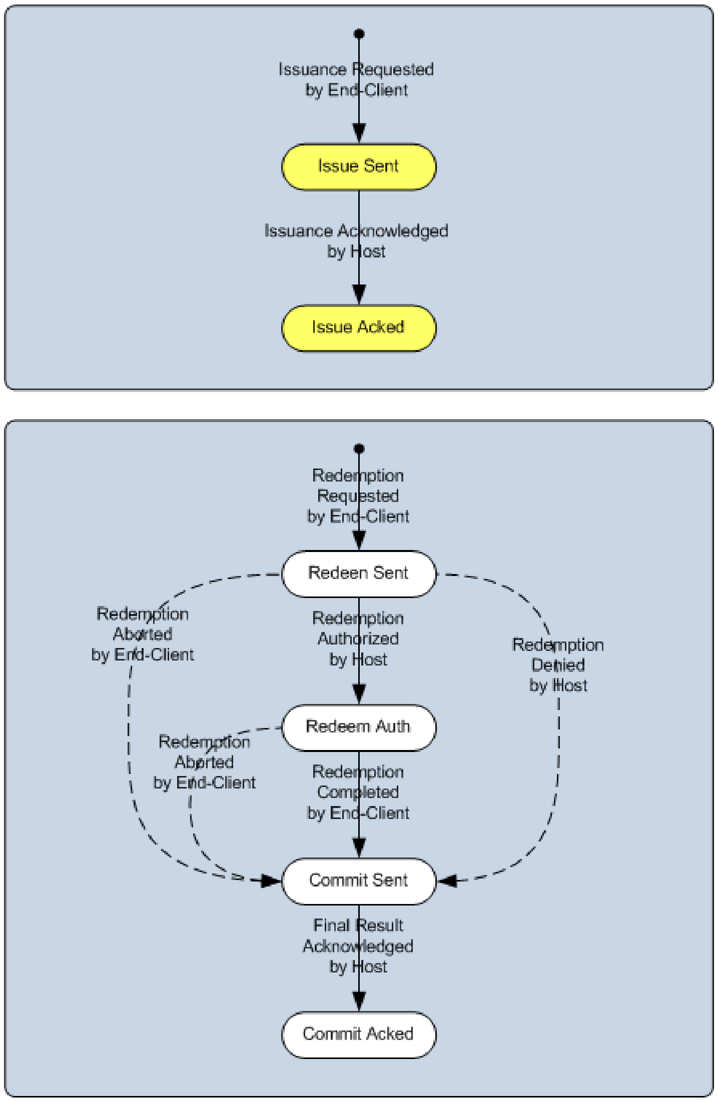
When a voucher is redeemed, it transitions through four additional states.

- When the redemption of the voucher is requested by the end-client, the state of the voucher transaction is set to `S2S_redeemSent`.
- After the redemption request has been authorized (or denied), the state of the voucher transaction is set to `S2S_redeemAuth`.
- When the final result of the redemption request is sent by the end-client, the state of the voucher transaction is set to `S2S_commitSent`.
- After the final result has been acknowledged, the state of the voucher transaction is set to `S2S_commitAcked`.

The following diagram identifies the voucher transaction states and the permitted transitions.

NEW CLASS

Figure 40.7 Voucher Transaction States


Issuance Requested by End-Client
Issue Sent
Issuance Acknowledged by Host
Issue Acked
Redemption Requested by End-Client
Redeen Sent
Redemption Aborted by End-Client
Redemption Authorized by Host
Redemption Denied by Host
Redeem Auth
Redemption Aborted by End-Client
Redemption Completed by End-Client
Commit Sent
Final Result Acknowledged by Host
Commit Acked

40.1.10 Validation Identifiers and Seeds

Validation identifiers are provided by the host to the end-clients. With each validation identifier, the host also provides a seed value. Validation identifiers MUST be 18-digit numeric values. Typically, the validation identifiers are printed on the vouchers in human-readable and bar-code form. Seed values MUST be constructed from 0 (zero) to 20 (twenty) UTF-8 encoded characters in the range U+0020 to U+007E (ASCII printable characters). Typically, the seed values are used to produce manual authentication identifiers which are also printed on the vouchers and can be used for offline validation of vouchers. See Section 40.1.11, Manual Authentication, for more details.

The edge-server is responsible for requesting validation identifiers for end-clients from the host. When the edge-server uses a protocol such as G2S to communicate with end-clients, this responsibility may be passed to the end-clients.

The voucher configuration information provided by the host to the edge-server includes a series of parameters that are used to determine the number of validation identifiers supplied to an end-client and the frequency at which new validation identifiers should be requested.

- The `maxValIds` parameter indicates the maximum number of validation identifiers that should be stored for an end-client.
- The `minLevelValIds` parameter indicates the minimum number of validation identifiers stored for an end-client before additional validation identifiers are requested.
- The `valIdListRefresh` parameter indicates the maximum time period that validation identifiers should be stored before new validation identifiers are requested.
- The `valIdListLife` parameter indicates the maximum time period before an end-client must stop using the validation identifiers.

Any time the number of validation identifiers stored for an end-client drops below `minLevelValIds`, additional validation identifiers MUST be requested from the host by the edge-server. Similarly, if the validation identifiers have been stored for a period of time that exceeds the `valIdListRefresh` or `valIdListLife` limits, a new set of validation identifiers MUST be requested. In addition, whenever voucher functionality is enabled by the host after being disabled, a new set of validation identifiers MUST be requested. When requesting validation identifiers, the number of validation identifiers requested for an end-client MUST NOT cause the number of validation identifiers stored for an end-client to exceed `maxValIds`. If the validation identifiers have been stored for a period of time that exceeds `valIdListLife`, the validation identifiers MUST NOT be used to issue vouchers until the validation identifiers have been refreshed by the host. See Section 40.20, getValidationIds Command, for more details.

The G2S protocol includes additional information about managing validation identifiers and seed values.

40.1.11 Manual Authentication

A manual authentication identifier MUST, if possible, be printed on every voucher for cashable or promotional credits produced by an end-client. And, when the `printNonCashOffLine` configuration parameter is set to `true`, a manual authentication MUST, if possible, be printed on every voucher for non-cashable credits. The manual authentication identifier is derived from a 128-bit MD5 hash of the end-client identifier, validation identifier, voucher amount, and seed value.

Operational circumstances may prevent the end-client from printing manual authentication identifiers. For example, the operator might choose to configure the end-client to not print manual authentication identifiers. If the end-client cannot print manual authentication identifiers, the end-client MUST NOT print vouchers while offline and the end-client MUST NOT allow the `printOffLine` configuration parameter to be set to `true`.

The following procedure MUST be used to produce manual authentication identifiers.

1. Construct a 90-character string composed, from left to right, of:
    a. end-client identifier (`endClientId`); 32 8-bit ASCII characters (U+0020 to U+007E) padded right with zeros (U+0030).
    b. validation identifier; 18 8-bit numeric ASCII characters (U+0030 to U+0039).
    c. seed value; 20 8-bit UTF-8 encoded characters (U+0020 to U+007E) padded left with zeros (U+0030).

NEW CLASS

d. voucher amount represented in the minor unit of the base currency of the end-client with no punctuation or currency symbols; 20 8-bit numeric ASCII characters (U+0030 to U+0039) padded left with zeros (U+0030).

2. Convert all lower case ASCII characters (U+0061 to U+007A) in the composed string to upper case ASCII characters (U+0041 to U+005A).
3. Produce a 128-bit MD5 hash value using the 90-character composed string as input.
4. Produce the manual authentication identifier by casting the 128-bit hash value into a 32-character hexadecimal representation and converting all alphabetic characters to upper case (U+0041 to U+005A).

The G2S protocol includes additional information about producing manual authentication identifiers.

40.1.12 Property Identifier

The `propertyId` attribute, which is contained in the class-level element of each command, is used to identify the property with which a command is associated. Commands within the `playerVoucher` class are property-specific—that is, the commands, as well as the data contained within the commands, are associated with a specific property. Within the `playerVoucher` class, the `propertyId` attribute of the class-level element of each command MUST be validated.

For all commands within the `playerVoucher` class:

- If the recipient of a command determines that the `propertyId` contained in the command is unknown or invalid, the recipient MUST report the error using error code `S2S_RIX201 Invalid Property Identifier`.

Commands contained within the `propertyInfo` class can be used to validate property identifiers and collect additional configuration information for properties.

40.1.13 Client Identifier

The `clientType` and `clientId` attributes, which are contained in each command within the `playerVoucher` class, uniquely identify a specific client within a gaming network. The client may be the system acting as an intermediary, such as an edge-server, or the client may be an end-client – that is, the end-client may be operating without the assistance of an edge-server. Within the `playerVoucher` class, the `clientType` and `clientId` attributes contained in each command MUST be validated.

Edge-servers and client systems are responsible for identifying the client, determining the `clientType` and `clientId` for the client, and communicating the `clientType` and `clientId` to the host system. The host system is responsible for validating the `clientType` and `clientId` reported by edge-servers and client systems and accurately processing commands based on that information.

For all commands within the `playerVoucher` class:

- If the recipient of a command determines that the combination of `clientType` and `clientId` associated with the command is unknown or invalid, the recipient MUST report the error using error code `MS13 Unknown Client`.
- If the recipient of a command determines that the client is disabled, the recipient MUST, if possible, process the command. If the recipient cannot process the command because the client is disabled, the recipient MUST report the error using error code `MS14 Client Disabled`.

NEW CLASS

- If the recipient of a command determines that the client is not registered for the property identified in the command, the recipient MUST report the error using error code `S2S_CIX202 Invalid Client Identifier for Property`.

Commands contained within the `clientInfo` class can be used to validate clients and to collect additional information about clients.

40.1.14 Player Identifier

Vouchers may be associated with a player identifier when they are issued. The edge-server is responsible for identifying the player, determining the player identifier for the player, and communicating the player identifier to the host system. The host system is responsible for validating the player identifiers reported by the edge-server and accurately reporting transactions based on that information.

For all commands within the `playerVoucher` class:

- Provided that the `playerId` is not <empty>, if the host determines that the `playerId` is invalid for the property, the host MUST report the error using error code `S2S_PIX202 Invalid Player Identifier for Property`.
- Provided that the `idNumber` is not <empty>, if the host determines that the combination of `idReaderType` and `idNumber` is invalid for the property, the host MUST report the error using error code `S2S_PIX204 Invalid Player Card for Property`.

When identifying a player, the edge-server MUST include either the `playerId` for the player or the `idReaderType` and `idNumber` of the player card presented by the player. Both sets of information MAY be included. If only one set of information is included, the host MAY include the other set of information in its response to the edge-server. If both sets of information are included, the host MUST ignore the `idReaderType` and `idNumber` and only use the `playerId` to identify the player. If neither set of information is included, the host MUST simply ignore the player information.

Note, however, that the host MUST NOT report errors related to invalid player information when processing voucher issuance requests. The host MUST make a best effort to accept voucher issuance requests even if the player information provided with the requests is invalid.

Commands contained within the `playerInfo` class can be used to validate player identifiers and to collect additional information about a player.

40.1.15 Configuration Identifier

The configuration identifier (`configurationId`) is used to identify a specific set of voucher configuration values. The host MAY specify a unique value for the `configurationId` when it sends a `setVoucherConfig` command to an edge-server requesting changes to the voucher configuration for an end-client. Once the edge-server has applied the changes, the edge-server MUST report the `configurationId` back to the host in any subsequent `voucherStatusList` or `voucherConfigList` commands that it sends to the host. The host can monitor the `configurationId` to make sure that the correct voucher configuration is in place for the end-client.

The host MAY also specify the `configurationId` when it sends a `setVoucherState` command to an edge-server. If the host specifies a non-zero value and the value does not match the `configurationId` currently in effect for the end-client, the edge-server MUST generate a `reqVoucherConfig` command to request the new voucher configuration for the end-client from the host.

The host can use whatever method it determines appropriate to assign a value to the `configurationId` provided that the value is greater than 0 (zero). The value `0` (zero) indicates that the voucher configuration was set locally at the end-client. Thus, the value `0` (zero) SHOULD NOT be used to identify a configuration set by the host.

When the G2S protocol is used to communicate with end-clients, the configuration identifier can be used as the `configurationId` for the G2S `voucher` device. More information about the G2S `configurationId` and `voucher` devices can be found in the G2S protocol.

40.1.16 Transaction Identifier

Each voucher transaction is assigned a transaction identifier (`transactionId`). The edge-server is responsible for assigning the `transactionId` to the transaction. However, the `transactionId` may be assigned by the end-client.

The `transactionId` MUST uniquely identify a specific voucher transaction for the end-client. At the edge-server, the combination of `endClientType`, `endClientId`, and `transactionId` MUST be unique. The `transactionId` itself MAY also be unique. Provided that a `transactionId` is unique to the end-client, the edge-server MAY use whatever method it determines appropriate to assign the `transactionId`. The edge-server MAY use a `transactionId` assigned by the end-client provided that the `transactionId` is unique for the end-client.

The host system MUST use the `transactionId` to detect duplicate voucher transactions, as well as updates to transactions, coming from an edge-server. Since the `transactionId` may only be unique to the edge-server, the host system MUST use the combination of `clientType`, `clientId`, `endClientType`, `endClientId`, and `transactionId` to uniquely identify individual transactions.

The `transactionId` assigned to a voucher transaction is specified by the edge-server when the transaction is requested by the edge-server. The edge-server is responsible for properly recording the `transactionId` and including it in any subsequent commands related to the transaction. The host system is responsible for properly recording the `transactionId` when the transaction is first requested and, subsequently, accurately detecting updates based on the `transactionId` provided by the edge-server.

40.1.17 Wildcard Conventions

The `playerVoucher` class supports two wildcards, `S2S_all` and `S2S_default`. The wildcards can be used to simplify end-client selection within certain commands. Not all commands support wildcards—wildcards MUST NOT be used unless specifically permitted within the description of a command.

The `S2S_all` wildcard is used to indicate that a command applies to all end-clients. Typically, end-clients are identified using the `endClientType` and `endClientId` attributes – for example, `endClientType = "S2S_egm"` and `endClientId = "ABC_123"`. Alternatively, all end-clients can be identified by specifying the `S2S_all` wildcard – for example, `endClientType = "S2S_all"` and `endClientId = "S2S_all"`. Or, all end-clients of a particular type can be specified – for example, `endClientType = "S2S_egm"` and `endClientId = "S2S_all"`.

- When the `S2S_all` wildcard is specified in a command for the `endClientType`, the command MUST be applied to all end-clients regardless of `endClientType`; otherwise, the command MUST only be applied to end-clients with the specified `endClientType`.
- When the `S2S_all` wildcard is specified in a command for the `endClientId`, the command MUST be applied to all end-clients regardless of the `endClientId`; otherwise, the command MUST only be applied to end-clients with the specified `endClientId`.

The `S2S_default` wildcard is used to indicate that a command contains a set of default configuration values for new end-clients. In the absence of any specific configuration settings for an end-client, the default configuration values MUST be applied to the end-client. For example, if the host has set a default configuration for end-clients of an edge-server, when a new end-client is registered with the edge-server, the edge-server should apply the default configuration to the end-client. Subsequently, the host may override the defaults by sending specific configuration settings for the end-client to the edge-server.

- When the `S2S_default` wildcard is specified for the `endClientType` in a command, the contents of the command MUST be used as the default for all end-clients regardless of `endClientType`; otherwise, the contents of the command MUST only be used as the default for end-clients with the specified `endClientType`.
- When the `S2S_default` wildcard is specified for the `endClientId` in a command, the contents of the command MUST be used as the default for all end-clients of the specified `endClientType`; otherwise, the contents of the command MUST NOT be used as the default for end-clients.

Note that the `S2S_all` wildcard is transient in nature. Commands containing the `S2S_all` wildcard are applied to the set of end-clients registered with an edge-server at the time that the commands are received. The `S2S_default` wildcard is persistent in nature. The commands containing the `S2S_default` wildcard are not applied until an end-client needs to be configured and there are no specific configuration settings for the end-client. Default configuration settings remain in effect and MUST be persisted, per the `persistDataType` of the edge-server, until superseded by new default configuration settings.

40.1.18 Categorization of Class

To help provide guidance to implementers regarding the maturity and stability of the S2S standard, GSA has categorized the various classes within S2S as Candidate Standards, Proposed Standards, or Recommended Standards. This class is categorized as a Candidate Standard.

- Standards identified as Candidate Standards are the least mature; changes to these standards should be expected in future releases.
- Standards identified as Proposed Standards have been reduced to practice and deployed; very few changes to these standards should be expected.
- Standards identified as Recommended are the most mature and have been widely deployed; no changes to these standards should be expected.

Further details about the categorization of standards and extensions can be found in the GSA Policy Handbook.

NEW CLASS

40.2 Data Set Summary

40.2.1 End-Client Configuration Data Sets

The following data sets are used by the host to manage the voucher configuration and status of end-clients. The host sends the voucher configuration and state information to the edge-server. The edge-server is then responsible for configuring the end-clients.

Table 40.1 End-Client Configuration Data Sets

Data Set	Section	Description
`voucherState`	Section 40.7	Used to set the current state of the voucher functionality on an end-client.
`voucherConfig`	Section 40.8	Used to set and report the current configuration of the voucher functionality on an end-client.
`voucherStatus`	Section 40.9	Used to report the current status of the voucher functionality on an end-client.

40.2.2 Voucher Reporting Data Sets

The following data set is used by the host to report voucher information to clients. The data set is used when a client requests information about a voucher. The data set is also used when a host reports voucher information based on a subscription from the client.

Table 40.2 Voucher Reporting Data Sets

Data Set	Section	Description
`voucher`	Section 40.10	Used to report voucher information to clients.

NEW CLASS

40.3 Command Summary

40.3.1 End-Client Configuration Commands

The following commands are used to query and manipulate the data sets used to manage the voucher configuration and status of end-clients.

Table 40.3 Commands Originated by Host

Command	Section	Description
`getVoucherStatus`	Section 40.11	Used to request the current voucher status of one or more end-clients from an edge-server. Causes the edge-server to generate one or more `voucherStatusList` commands.
`setVoucherState`	Section 40.12	Used to change the voucher state of one or more end-clients managed by an edge-server. Causes the edge-server to generate one or more `voucherStatusList` commands.
`getVoucherConfig`	Section 40.13	Used to request the current voucher configuration information for one or more end-clients from an edge-server. Causes the edge-server to generate one or more `voucherConfigList` commands.
`setVoucherConfig`	Section 40.14	Used to change the current voucher configuration information for one or more end-clients managed by an edge-server. Causes the edge-server to generate one or more `voucherStatusList` commands.
`voucherConfigAck`	Section 40.19	Used to acknowledge the receipt of a configuration-related request from an edge-server.

Table 40.4 Commands Originated by Edge-Server (Sheet 1 of 2)

Command	Section	Description
`reqVoucherStatus`	Section 40.15	Used to request the current voucher status for one or more end-clients from the host. Causes the host to generate one or more `setVoucherState` commands.
`voucherStatusList`	Section 40.16	Used to report the current voucher status for one or more end-clients to the host.
`reqVoucherConfig`	Section 40.17	Used to request the current voucher configuration for one or more end-clients from the host. Causes the host to generate one or more `setVoucherConfig` commands.
`voucherConfigList`	Section 40.18	Used to report the current voucher configuration for one or more end-clients to the host.

NEW CLASS

Table 40.4 Commands Originated by Edge-Server (Sheet 2 of 2)

Command	Section	Description
`voucherConfigAck`	Section 40.19	Used to acknowledge the receipt of a configuration-related request from a host.

40.3.2 Voucher Issuance & Redemption Commands

The following commands are used to manage voucher transactions between hosts and end-clients.

Table 40.5 Commands Originated by Host

Command	Section	Description
`validationIdList`	Section 40.21	Used to provide validation identifiers and seed values to an end-client.
`issueVoucherAck`	Section 40.23	Used to acknowledge receipt of a voucher issuance request by an end-client.
`authorizeVoucher`	Section 40.25	Used to authorize a voucher redemption request by an end-client.
`commitVoucherAck`	Section 40.27	Used to acknowledge receipt of the final results of a voucher redemption request by an end-client.

Table 40.6 Commands Originated by Edge-Server

Command	Section	Description
`getValidationIds`	Section 40.20	Used to request validation identifiers and seed values for an end-client. Causes the host to generate a `validationIdList` command.
`issueVoucher`	Section 40.22	Used to report that a voucher has been issued by an end-client. Causes the host to generate an `issueVoucherAck` command.
`redeemVoucher`	Section 40.24	Used to request authorization for an end-client to redeem a voucher. Causes the host to generate an `authorizeVoucher` command.
`commitVoucher`	Section 40.26	Used to report the final results of a voucher redemption transaction for an end-client. Causes the host to generate a `commitVoucherAck` command.

NEW CLASS

40.3.3 Voucher Reporting Commands

The following commands are used to report voucher information to interested clients.

Table 40.7 Commands Originated by Host

Command	Section	Description
`voucherUpdate`	Section 40.28	Used to report updates to voucher information to subscribed clients. Causes the client to generate a `voucherUpdateAck` command.
`voucherResults`	Section 40.31	Used to report the results of voucher information queries to clients.
`postVouchersAck`	Section 40.33	Used to report that voucher information has been posted to a specified location.

Table 40.8 Commands Originated by Client

Command	Section	Description
`voucherUpdateAck`	Section 40.29	Used to acknowledge receipt of voucher information from the host.
`queryVouchers`	Section 40.30	Used to request voucher information from the host. Causes the host to generate a `voucherResults` command.
`postVouchers`	Section 40.32	Used to request that voucher information be posted to a specified location. Causes the host to generate a `postVoucherAck` command.

NEW CLASS

40.4 Request-Response Pairs

The following tables organize the commands contained within the `playerVoucher` class into request-response pairs. The tables are organized into functional groups. In each table:

- The first column identifies which commands MAY be sent as requests.
- The second column identifies which commands MUST be sent as responses to the requests.
- The third column indicates whether the request MAY be sent as a notification. If a command is not permitted to be sent as a notification, the command MUST NOT be sent as such.

40.4.1 End-Client Configuration Request-Response Pairs

This group of request-response pairs is used to manage the voucher configuration and status of end-clients.

Table 40.9 Request-Response Pairs Originated by Host

Request	Response	Notification
`getVoucherStatus`	`voucherConfigAck`	No
`setVoucherState`	`voucherConfigAck`	No
`getVoucherConfig`	`voucherConfigAck`	No
`setVoucherConfig`	`voucherConfigAck`	No

Table 40.10 Request-Response Pairs Originated by Edge-Server

Request	Response	Notification
`reqVoucherStatus`	`voucherConfigAck`	No
`voucherStatusList`	`voucherConfigAck`	No
`reqVoucherConfig`	`voucherConfigAck`	No
`voucherConfigList`	`voucherConfigAck`	No

40.4.2 Voucher Issuance & Redemption Request-Response Pairs

This group of request-response pairs is used to manage voucher transactions between hosts and end-clients.

Table 40.11 Request-Response Pairs Originated by Edge-Server

Request	Response	Notification
`getValidationIds`	`validationIdList`	No
`issueVoucher`	`issueVoucherAck`	No
`redeemVoucher`	`authorizeVoucher`	No
`commitVoucher`	`commitVoucherAck`	No

NEW CLASS

40.4.3 Voucher Reporting Request-Response Pairs

This group of request-response pairs is used to report voucher information to interested clients.

Table 40.12 Request-Response Pairs Originated by Host

Request	Response	Notification
`voucherUpdate`	`voucherUpdateAck`	Yes

Table 40.13 Request-Response Pairs Originated by Client

Request	Response	Notification
`queryVouchers`	`voucherResults`	No
`postVouchers`	`postVouchersAck`	No

NEW CLASS

40.5 System Services

The following tables organize the commands and data sets within the `playerVoucher` class into system service groups. These groups are used by a system to describe its capabilities to another system. See Section 1.8, System Capabilities, for more details about system service groups.

For the `playerVoucher` class, the host system acts as the provider of services (`S2S_host`) to client systems and edge-servers. The client systems and edge-servers act as the consumers of services (`S2S_client`) provided by the host. The host system provides services to clients that allow clients to query a database of voucher information maintained by the host and to request that updates to voucher information be sent to the clients. The host system provides services to edge-servers that are used to set voucher configurations for end-clients and to manage voucher transactions between the host and end-clients.

The following table identifies the requests and notifications that may be originated by a system that is acting in a specific role—that is, consumer or provider. The requests and notifications that may be originated by a system acting as a consumer are listed separately from the requests and notifications that may be originated by a system acting as a provider.

The requests and notifications listed for a specific role MAY be originated by a system acting in that role; the same requests and notifications MUST be supported by a system acting in the opposite role.

- The requests and notifications originated by a system acting as a consumer (`S2S_client`) MUST be supported by a system acting as a provider (`S2S_host`).
- Likewise, the requests and notifications originated by a system acting as a provider (`S2S_host`) MUST be supported by a system acting as a consumer (`S2S_client`).

Table 40.14 playerVoucher System Service Groups

classType	serviceType	Requests and Notifications Originated by Consumer (S2S_client)	Requests and Notifications Originated by Provider (S2S_host)

`S2S_playerVoucher`	`S2S_configuration`	`voucherStatusList`
		`voucherConfigList`
		`reqVoucherStatus`
		`reqVoucherConfig`
			`getVoucherStatus`
			`setVoucherState`
			`getVoucherConfig`
			`setVoucherConfig`

`S2S_playerVoucher`	`S2S_issueVouchers`	`getValidationIds` `issueVoucher`	
`S2S_playerVoucher`	`S2S_redeemVouchers`	`redeemVoucher` `commitVoucher`	
`S2S_playerVoucher`	`S2S_voucherUpdates`		`voucherUpdate`
`S2S_playerVoucher`	`S2S_queryVouchers`	`queryVouchers`	
`S2S_playerVoucher`	`S2S_postVouchers`	`postVouchers`	

40.5.1 Critical Data

When a system indicates that it retains critical data—that is, the `persistDataType` for a service is set to `S2S_indefinitely` or `S2S_untilOutage`—the following data sets are considered critical data and, to the extent required by the `persistDataType`, MUST be retained by the system: `voucherState`, `voucherConfig`, `voucherStatus`, and `voucher`.

As the system of record for voucher information, the host system MUST report that it persists the critical data indefinitely—that is, the host MUST set the `persistDataType` for its services to `S2S_indefinitely`.

Edge-servers and clients MAY choose whether to persist the critical data. When an edge-server or client chooses not to persist the critical data, the edge-server or client is responsible for recovering the critical data from the host upon restart; the edge-server or client MUST request current status and configuration information from the host upon restart as needed to refresh the critical data. The host system is not expected to change its operational mode based on whether the edge-server or client persists the critical data.

Similarly, the edge-server or client is responsible for requesting critical data from the host when a new end-client is discovered; in particular, the edge-server or client MUST request status and configuration information as needed from the host when a new end-client is discovered.

NEW CLASS

40.6 Information Updates

The following table identifies the information updates within the `playerVoucher` class that MAY be generated by a host. A system MAY subscribe to these information updates using commands within the `clientUpdate` class. See Chapter 27, clientUpdate Class, for more details on subscribing to information updates.

Table 40.15 playerVoucher Information Updates

className	commandName	elementName
`S2S_playerVoucher`	`voucherUpdate`	`voucher`

NEW CLASS

40.7 voucherState Data Set

40.7.1 Data Set Description

The following table identifies the attributes and sub-elements of the `voucherState` data set. The `voucherState` element contains information that is used by the host to control the overall state of voucher functionality on an end-client. Sub-elements of the `voucherState` element, if any, may contain additional information about the state of voucher functionality.

The `enable` and `disableText` attributes contained in the `voucherState` data set map directly to corresponding attributes in the `setVoucherState` command of the G2S protocol. The `lockout`, `lockText`, and `lockTimeOut` attributes contained in the `voucherState` data set map directly to corresponding attributes in the `setVoucherLockOut` command of the G2S protocol. More information about these attributes can be found in the G2S protocol.

40.7.2 Attribute and Element Detail

voucherState
type
pvc:c_voucherState

Table 40.16 voucherState Attributes

Attribute	Restrictions	Description
`enable`	type: `xs:boolean` use: optional default: true	Indicates whether voucher functionality should be enabled. (`setVoucherState.enable`)
`disableText`	type: `t_textMessage` use: optional default: <empty>	Text message to display while the device is disabled. (`setVoucherState.disableText`)
`lockOut`	type: `xs:boolean` use: optional default: false	Indicates whether the end-client should be locked. (`setVoucherLockOut.lockOut`)
`lockText`	type: `t_textMessage` use: optional default: <empty>	Text message to display while the end-client is locked. (`setVoucherLockOut.lockText`)
`lockTimeOut`	type: `t_milliseconds` use: optional default: 1000	Duration of the lock in milliseconds. If the lock has not been removed within the time specified, the end-client should remove the lock. (`setVoucherLockOut.lockTimeOut`)
`configurationId`	type: `t_configurationId` use: optional default: 0	Configuration identifier; used by the edge-server to determine whether the correct configuration is in use by the end-client. (`voucherStatus.configurationId`)

NEW CLASS

40.8 voucherConfig Data Set

40.8.1 Data Set Description

The following table identifies the attributes and sub-elements of the `voucherConfig` data set. The `voucherConfig` element contains information that is used by the host to set the voucher configuration for an end-client. Sub-elements of the `voucherConfig` element may contain additional information about the voucher configuration.

The attributes contained in the `voucherConfig` data set map directly to corresponding attributes in the `voucherProfile` command of the G2S protocol. More information about the attributes can be found in the G2S protocol.

When this data set is included in a request to change the configuration of an end-client, the following error conditions MAY be reported by the recipient of the request, indicating that no action was taken for the end-client:

- If the recipient determines that manual authentication identifiers cannot be printed by the end-client and offline printing has been requested, the recipient MUST report the error using error code `S2S_PVX001 Manual Authentication Identifiers Not Supported by End-Client`.

40.8.2 Attribute and Element Detail

voucherConfig
type
pvc:c_voucherConfig

Table 40.17 voucherConfig Attributes (Sheet 1 of 4)

Attribute	Restrictions	Description
`configurationId`	type: `t_configurationId` use: optional default: 0	Configuration identifier; used by the edge-server to determine whether the correct configuration is in use by the end-client. (`voucherProfile.configurationId`)
`restartStatus`	type: `xs:boolean` use: optional default: true	Indicates the state of voucher functionality upon restart. (`voucherProfile.restartStatus`)
`requiredForPlay`	type: `xs:boolean` use: optional default: false	Indicates whether the end-client must be disabled when voucher functionality is disabled. (`voucherProfile.requiredForplay`)
`timeToLive`	type: `t_milliseconds` use: optional default: 30000	Time-to-live value for voucher-related requests generated by the end-client. (`vocuherProfile.timeToLive`)
`combineCashableOut`	type: `xs:boolean` use: optional default: false	Indicates whether promotional credits are converted to cashable credits when issuing vouchers. (`voucherProfile.combineCashableOut`)

NEW CLASS

Table 40.17 voucherConfig Attributes (Sheet 2 of 4)

Attribute	Restrictions	Description
`allowNonCashOut`	type: `xs:boolean` use: optional default: true	Indicates whether the end-client is allowed to issue vouchers for non-cashable credits. (`voucherProfile.allowNonCashOut`)
`maxValIds`	type: `xs:int` use: optional default: 20 minIncl: 1	Maximum number of validation identifiers stored by the end-client. (`voucherProfile.maxValIds`)
`minLevelValIds`	type: `xs:int` use:optional default: 15 minIncl: 0	Minimum number of validation identifiers stored by the end-client before additional validation identifiers are requested. (`voucherProfile.minLevelValIds`)
`valIdListRefresh`	type: `t_milliseconds` use: optional default: 43200000	Maximum time period after validation identifiers have been refreshed before new validation identifiers are requested. (`voucherProfile.valIdListRefresh`)
`valIdListLife`	type: `t_milliseconds` use: optional default: 84600000	Maximum time period after validation identifiers have been refreshed before the end-client must stop using the validation identifiers. (`voucherProfile.valListLife`)
`voucherHoldTime`	type: `t_milliseconds` use: optional default: 15000	Maximum time period that the end-client should wait for a host authorization before returning a voucher. (`voucherProfile.voucherHoldTime`)
`printOffLine`	type: `xs:boolean` use: optional default: true	Indicates whether the end-client is allowed to issue vouchers while communications to the edge-server are lost. (`voucherProfile.printOffline`)
`expireCashPromo`	type: `xs:int` use: optional default: 30 minIncl: 0	Number of days before vouchers for cashable and promotional credits expire. (`voucherProfile.expireCashPromo`)
`printExpCashPromo`	type: `xs:boolean` use: optional default: true	Indicates whether expiration dates should be printed on vouchers for cashable and promotional credits. (`voucherProfile.printExpcashPromo`)
`expireNonCash`	type: `xs:int` use: optional default: 30 minIncl: 0	Number of days before vouchers for non-cashable credits expire. (`voucherProfile.expireNonCash`)
`printExpNonCash`	type: `xs:boolean` use: optional default: true	Indicates whether expiration dates should be printed on vouchers for non-cashable credits. (`voucherProfile.printExpNonCash`)
`propName`	type: `t_voucherTitle40` use: required	Name of the property. (`voucherProfile.propName`)
`propLine1`	type: `t_voucherTitle40` use: required	First address line for the property. (`voucherProfile.titlePropLine1`)

Table 40.17 voucherConfig Attributes (Sheet 3 of 4)

Attribute	Restrictions	Description
propLine2	type: t_voucherTitle40 use: required	Second address line for the property. (voucherProfile.titlePropLine2)
titleCash	type: t_voucherTitle16 use: required	Title printed on vouchers for cashable credits. (voucherProfile.titleCash)
titlePromo	type: t_voucherTitle16 use: optional default: <empty>	Title printed on vouchers for promotional credits; if <empty>, use titleCash. (voucherProfile.titlePromo)
titleNonCash	type: t_voucherTitle16 use: required	Title printed on vouchers for non-cashable credits. (voucherProfile.titleNonCash)
titleLargeWin	type: t_voucherTitle16 use: required	Title printed on vouchers for wins greater than the large win limit for the end-client. (voucherProfile.titleLargeWin)
titleBonusCash	type: t_voucherTitle16 use: required	Title printed on vouchers for cashable credits resulting from external bonus awards. (voucherProfile.titleBonusCash)
titleBonusPromo	type: t_voucherTitle16 use: optional default: <empty>	Title printed on vouchers for promotional credits resulting from external bonus awards; if <empty>, use titleBonusCash. (voucherProfile.titleBonusPromo)
titleBonusNonCash	type: t_voucherTitle16 use: required	Title printed on vouchers for non-cashable credits resulting from external bonus awards. (voucherProfile.titleBonusNonCash)
titleWatCash	type: t_voucherTitle16 use: required	Title printed on vouchers for cashable credits resulting from wagering account transfers. (voucherProfile.titleWatCash)
titleWatPromo	type: t_voucherTitle16 use: optional default: <empty>	Title printed on vouchers for promotional credits resulting from wagering account transfers; if <empty>, use titleWatCash. (voucherProfile.titleWatPromo)
titleWatNonCash	type: t_voucherTitle16 use: required	Title printed on vouchers for non-cashable credits resulting from wagering account transfers. (voucherProfile.titleWatNonCash)
allowVoucherIssue	type: xs:boolean use: optional default: true	Indicates whether the end-client should request validation identifiers, thus, enabling voucher issuance functionality. (voucherProfile.allowVoucherIssue)
allowVoucherRedeem	type: xs:boolean use: optional default: true	Indicates whether the end-client should support voucher redemption functionality. (voucherProfile.allowVoucherRedeem)

NEW CLASS

Table 40.17 voucherConfig Attributes (Sheet 4 of 4)

Attribute	Restrictions	Description
`maxOnLinePayOut`	type: `t_meterValue` use: optional default: 0	Maximum amount that can be paid by voucher while communications are not lost; `0` (zero) indicates that there is no limit. (`voucherProfile.maxOnLinePayOut`)
`maxOffLinePayOut`	type: `t_meterValue` use: optional default: 0	Maximum amount that can be paid by voucher while communications are lost; `0` (zero) indicates that there is no limit. (`voucherProfile.maxOffLinePayOut`)
`printNonCashOffLine`	type: `xs:boolean` use: optional default: true	Indicates whether vouchers for non-cashable credits can be issued while communications are lost; both `printOffLine` and `allowNonCashOut` must also be set to `true` for vouchers for non-cashable credits to be printed while communications are lost. (`voucherProfile.printNonCashOffLine`)
`usePlayerIdReader`	type: `xs:boolean` use: optional default: false	Indicates whether the ID reader associated with the currently active player session or a specific ID reader should be used when reporting voucher information. (`voucherProfile.usePlayerIdReader`)
`noAckTimer`	type: `t_milliseconds` use: optional default: 15000	Indicates the maximum time between when a voucher is issued and when it is acknowledged before the validation system is declared offline. (`voucherProfile.noAckTimer`)

NEW CLASS

40.9 voucherStatus Data Set

40.9.1 Data Set Description

The following table identifies the attributes and sub-elements of the `voucherStatus` data set. The `voucherStatus` element contains information that is used by the edge-server to report the current status of voucher functionality for an end-client. Sub-elements of the `voucherStatus` element, if any, may contain additional information about the current status of voucher functionality for an end-client.

The attributes contained in the `voucherStatus` data set map directly to corresponding attributes in the `voucherStatus`, `voucherProfile`, and `idReaderProfile` commands of the G2S protocol. More information about the attributes can be found in the G2S protocol.

40.9.2 Attribute and Element Detail

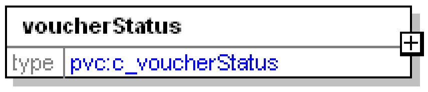


Table 40.18 voucherStatus Attributes

Attribute	Restrictions	Description

`enabled`	`type: xs:boolean`
	`use: optional`
	`default: true`
		Indicates whether voucher functionality has been enabled by the host for the end-client. (`voucherStatus.hostEnabled`)

`locked`	`type: xs:boolean`
	`use: optional`
	`default: false`
		Indicates whether the end-client has been locked by the host. (`voucherStatus.hostLocked`)

`validationListId`	`type: t_validationListId`
	`use: required`
		Validation list identifier sent by the host with the most recent set of validation identifiers. (`voucherStatus.validationListId`)

`configurationId`	`type: t_configurationId`
	`use: optional`
	`default: 0`
		Configuration identifier; used by the edge-server to determine whether the correct configuration is in use by the end-client. (`voucherStatus.configurationId`)

`idReaderType`	`type: t_idReaderTypes`
	`use: optional`
	`default: S2S_none`
		Type of ID reader used to identify players. (`idReaderProfile.idReaderType` via `voucherProfile.idReaderId`)

`systemOnLine`	`type: xs:boolean`
	`use: optional`
	`default: true`
		Indicates whether the validation system is online. (`voucherStatus.systemOnline`)

NEW CLASS

40.10 voucher Data Set

40.10.1 Data Set Description

The following table identifies the attributes and sub-elements of the `voucher` data set. The `voucher` element contains information that is used to report the status of voucher transactions between the host and end-clients. Sub-elements of the `voucher` element, if any, may contain additional information about the voucher transactions.

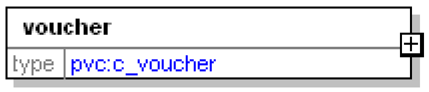
The attributes contained in the `voucher` data set map directly to corresponding attributes in various commands within the `voucher` class of the G2S protocol. More information about the attributes can be found in the G2S protocol.

- Other than syntactical errors, the presence of unknown or invalid values in attributes of the `voucher` element is not considered an error. An implementation SHOULD, if possible, accept invalid values in the `endClientType`, `voucherState`, `idReaderType`, `voucherAction`, `creditType`, `voucherSource`, `hostAction`, and `endClientAction` attributes. An implementation is expected to make a best effort to accept and process voucher information.

However, if the recipient is unable to accept an unknown or invalid value, the following error conditions MAY be reported by the recipient of the request, indicating that no action was taken:

- If the recipient determines that the `endClientType` is invalid, the recipient MUST report the error using error code `S2S_PVX002 Invalid End-Client Type`.
- If the recipient determines that the `voucherState` is invalid, the recipient MUST report the error using error code `S2S_PVX004 Invalid Voucher State`.
- If the recipient determines that the `idReaderType` is invalid, the recipient MUST report the error using error code `S2S_GBX034 Invalid ID Reader Type`.
- If the recipient determines that the `voucherAction` is invalid, the recipient MUST report the error using error code `S2S_PVX005 Invalid Voucher Action`.
- If the recipient determines that the `creditType` is invalid, the recipient MUST report the error using error code `S2S_RIX039 Invalid Credit Type`.
- If the recipient determines that the `voucherSource` is invalid, the recipient MUST report the error using error code `S2S_PVX006 Invalid Voucher Source`.
- If the recipient determines that the `hostAction` is invalid, the recipient MUST report the error using error code `S2S_PVX007 Invalid Host Action`.
- If the recipient determines that the `egmAction` is invalid, the recipient MUST report the error using error code `S2S_PVX008 Invalid End-Client Action`.

40.10.2 Attribute and Element Detail



NEW CLASS

Table 40.19 voucher Attributes (Sheet 1 of 2)

Attribute	Restrictions	Description
endClientType	type: t_clientTypes use: required	End-client type; type of end-client that originated the voucher request. Wildcards are not permitted.
endClientId	type: t_clientId use: required	End-client identifier; client identifier of the end-client that originated the voucher request. Wildcards are not permitted.
Transaction Identification and Status Information		
transactionId	type: t_transactionId use: required	Transaction identifier; assigned by the end-client. (voucherLog.transactionId)
voucherStatus	type: t_voucherStates use: required	Transaction status; current status of the transaction. (voucherLog.voucherState)
voucherAction	type: t_voucherActions use: required	Voucher action: issue or redeem. (voucherLog.voucherAction)
Player Identification Information		
idReaderType	type: t_idReaderTypes use: optional default: S2S_none	Type of ID reader used to identify the player. (voucherLog.idReaderType)
idNumber	type: t_idNumber use: optional default: <empty>	ID number associated with the player. (voucherLog.idNumber)
playerId	type: t_playerId use: optional default: <empty>	Player identifier. (voucherLog.playerId)
Voucher Attributes		
validationId	type: t_validationId use: required	Validation identifier; all but the last four digits should be masked. (voucherLog.validationId)
voucherAmt	type: t_meterValue use: required	Voucher amount. (voucherLog.vouchetAmt)
creditType	type: t_creditTypes use: required	Type of credits; cashable, promotional, or non-cashable. (voucherLog.creditType)
voucherSource	type: t_voucherSources use: optional default: S2S_endClient	Indicates the source of the voucher; end-client or system. (voucherLog.voucherSequence)
largeWin	type: xs:boolean use: optional default: false	Indicates whether the voucher was issued because the amount won exceeded the end-client's large win limit. (voucherLog.largeWin)

NEW CLASS

Table 40.19 voucher Attributes (Sheet 2 of 2)

Attribute	Restrictions	Description
voucherSequence	type: xs:int use: optional default: 0 minIncl: 0	The issuing end-client's internal voucher sequence number printed on the voucher. (voucherLog.voucherSequence)
expireCredits	type: xs:boolean use: optional default: false	Indicates whether non-cashable credits have an associated expiration date/time; ignored when creditType is not set to S2S_nonCashable. (voucherLog.expireCredits)
expireDateTime	type: t_dateTime use: optional default: 2000-01-01T00:00:00.000-00:00	Expiration date/time associated with non-cashable credits; ignored when creditType is not set to S2S_nonCashable or expireCredits is set to false. (voucherLog.expireDateTime)
hostAction	type: t_voucherHostActions use: optional default: S2S_endClientAction	Indicates whether the host preferred the voucher to be stacked, returned, or that the end-client determines the appropriate action; ignored when voucherAction is set to S2S_issue. (voucherLog.hostAction)
hostException	type: t_voucherHostExcs use: optional default: 0	Exception code set by host; ignored when voucherAction is set to S2S_issue. (voucherLog.hostException)
Final Result Information		
transferAmt	type: t_meterValue use: optional default: 0	Actual amount transferred. (voucherLog.transferAmt)
transferDateTime	type: t_dateTime use: required	Date/time that the transaction record was updated. (voucherLog.transferDateTime)
expireDays	type: xs:int use: optional default: -1 minIncl: -1	Number of days before the voucher expires; ignored if voucherAction is set to S2S_redeem or expireCredits is set to true; -1 (negative one) indicates that there is no expiration period; set to -1 (negative one) when voucherAction is set to S2S_redeem or expireCredits is set to true. (voucherLog.expireDays)
endClientAction	type: t_voucherClientActions use: required	Indicates whether the end-client issued, stacked or returned the voucher. (voucherLog.egmAction)
endClientException	type: t_voucherClientExcs use: optional default: 0	End-client exception code; 0 (zero) indicates no end-client exceptions. (voucherLog.egmException)

NEW CLASS

40.11 getVoucherStatus Command

40.11.1 Command Description

This command is used by a host to request current voucher status information for one or more end-clients. The command MUST only be used with point-to-point communications channels and MUST only be sent as a request. A `voucherConfigAck` command is generated in response to the `getVoucherStatus` command, indicating that the edge-server has received the request and will make a best-effort attempt to apply the request to the list of intended end-clients. Subsequently, as the results of the request become available, the edge-server MUST generate one or more `voucherStatusList` commands to report the results to the host.

The `propertyId` attribute of the class-level element is used to identify the property for which status information is being requested. The edge-server MUST only include status information associated with that property in its response.

In addition to other event codes that the edge-server MAY report, the edge-server MAY report the following event codes, indicating that the requested action was not taken for an end-client.

- If the edge-server determines that an end-client is not registered for the property, the edge-server MUST report the error using event code `S2S_PVE001 Invalid End-Client For Property`.
- If the edge-server is unable to report the voucher status of an end-client, the edge-server MUST report the error using event code `S2S_PVE002 Voucher Status Unavailable For End-Client`.

40.11.2 Attribute and Element Detail

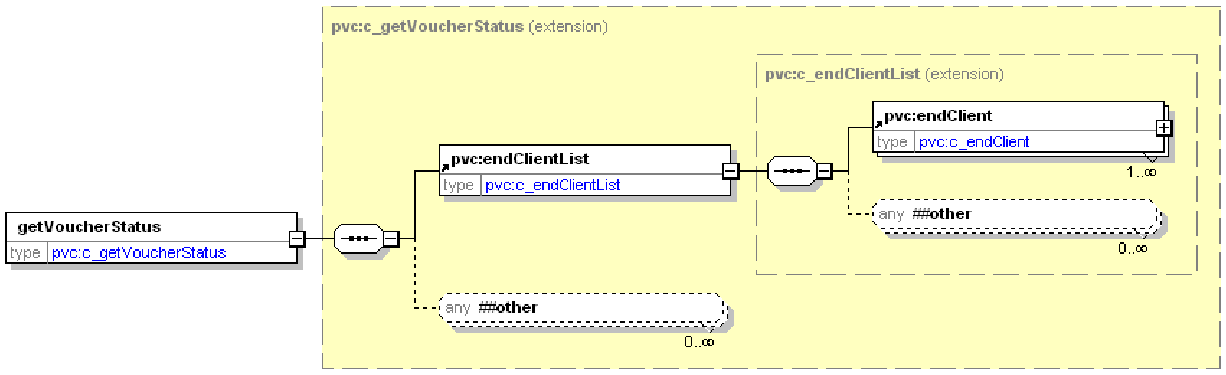


Table 40.20 getVoucherStatus Attributes

Attribute	Restrictions	Description
`clientType`	type: `t_clientTypes` use: required	Client type; the type of client to which the command is directed; typically, an edge-server. Wildcards are not permitted.
`clientId`	type: `t_clientId` use: required	Client identifier; the identifier of the client to which the command is directed; typically, an edge-server. Wildcards are not permitted.

NEW CLASS

Table 40.21 getVoucherStatus Sub-Elements

Element	Restrictions	Description

`endClientList`	`minOcc: 1`
	`maxOcc: 1`
		Contains the list of end-clients. See Table 40.22.

Table 40.22 endClientList Sub-Elements

Element	Restrictions	Description

`endClient`	`minOcc: 1`
	`maxOcc: ∞`
		Identifies an end-client. See Table 40.23.

Table 40.23 endClient Attributes

Attribute	Restrictions	Description

`endClientType`	`type: t_clientTypes`
	`use: required`
		End-client type; the type of end-client for which information is being requested. Wildcards are not permitted.

`endClientId`	`type: t_clientId`
	`use: required`
		End-client identifier; the identifier of the end-client for which information is being requested. The `S2S_all` and `S2S_default` wildcards are permitted.

NEW CLASS

40.12 setVoucherState Command

40.12.1 Command Description

This command is used by a host to set the voucher state of one or more end-clients. The command MUST only be used with point-to-point communications channels and MUST only be sent as a request. A `voucherConfigAck` command is generated in response to the `setVoucherState` command, indicating that the edge-server has received the request and will make a best-effort attempt to apply the request to the list of intended end-clients. Subsequently, as the results of the request become available, the edge-server MUST generate one or more `voucherStatusList` commands to report the results to the host.

Data sets, which are included in the `setVoucherState` command, overwrite any previous information related to the data sets. Data sets, which are not included, are not affected.

The `propertyId` attribute of the class-level element is used to identify the property for which voucher states are being set. The edge-server MUST only set the voucher states for end-clients associated with that property.

In addition to other event codes that the edge-server MAY report, the edge-server MAY report the following event codes, indicating that the requested action was not taken for an end-client.

- If the edge-server determines that an end-client is not registered for the property, the edge-server MUST report the error using event code `S2S_PVE001 Invalid End-Client For Property`.
- If the edge-server is unable to set the voucher state for an end-client, the edge-server MUST report the error using event code `S2S_PVE003 Unable To Set Voucher State For End-Client`.

40.12.2 Attribute and Element Detail

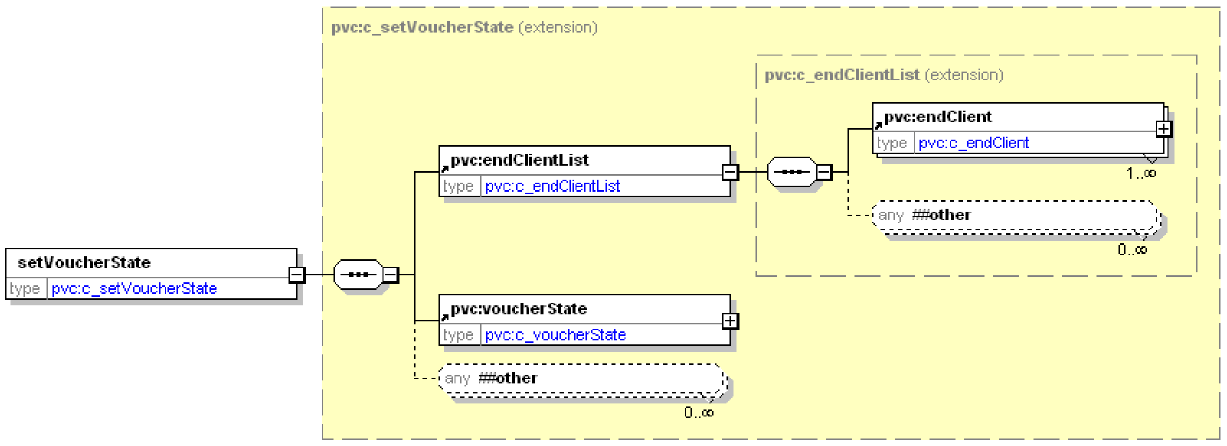


Table 40.24 setVoucherState Attributes (Sheet 1 of 2)

Attribute	Restrictions	Description

`clientType`	`type: t_clientTypes`
	`use: required`
		Client type; the type of client to which the command is directed; typically, an edge-server. Wildcards are not permitted.

NEW CLASS

Table 40.24 setVoucherState Attributes (Sheet 2 of 2)

Attribute	Restrictions	Description

`clientId`	`type: t_clientId`
	`use: required`
		Client identifier; the identifier of the client to which the command is directed; typically, an edge-server. Wildcards are not permitted.

Table 40.25 setVoucherState Sub-Elements

Element	Restrictions	Description

`endClientList`	`minOcc: 1`
	`maxOcc: 1`
		Contains the list of end-clients. See Table 40.26.

`voucherState`	`minOcc: 1`
	`maxOcc: 1`
		Contains voucher state parameters for the list of end-clients. See Section 40.7, voucherState Data Set.

Table 40.26 endClientList Sub-Elements

Element	Restrictions	Description

`endClient`	`minOcc: 1`
	`maxOcc: ∞`
		Identifies an end-client. See Table 40.27.

Table 40.27 endClient Attributes

Attribute	Restrictions	Description

`endClientType`	`type: t_clientTypes`
	`use: required`
		End-client type; the type of end-client being configured. Wildcards are not permitted.

`endClientId`	`type: t_clientId`
	`use: required`
		End-client identifier; the identifier of the end-client being configured. The `S2S_all` and `S2S_default` wildcards are permitted.

NEW CLASS

40.13 getVoucherConfig Command

40.13.1 Command Description

This command is used by a host to request the current voucher configuration information for one or more end-clients. The command MUST only be used with point-to-point communications channels and MUST only be sent as a request. A `voucherConfigAck` command is generated in response to the `getVoucherConfig` command, indicating that the edge-server has received the request and will make a best-effort attempt to apply the request to the list of intended end-clients. Subsequently, as the results of the request become available, the edge-server MUST generate one or more `voucherConfigList` commands to report the results to the host.

The `propertyId` attribute of the class-level element is used to identify the property for which configuration information is being requested. The edge-server MUST only include configuration information associated with that property in its response.

In addition to other event codes that the edge-server MAY report, the edge-server MAY report the following event codes, indicating that the requested action was not taken for an end-client.

- If the edge-server determines that an end-client is not registered for the property, the edge-server MUST report the error using event code `S2S_PVE001 Invalid End-Client For Property`.
- If the edge-server is unable to report the voucher configuration for an end-client, the edge-server MUST report the error using event code `S2S_PVE004 Voucher Configuration Not Available For End-Client`.

40.13.2 Attribute and Element Detail

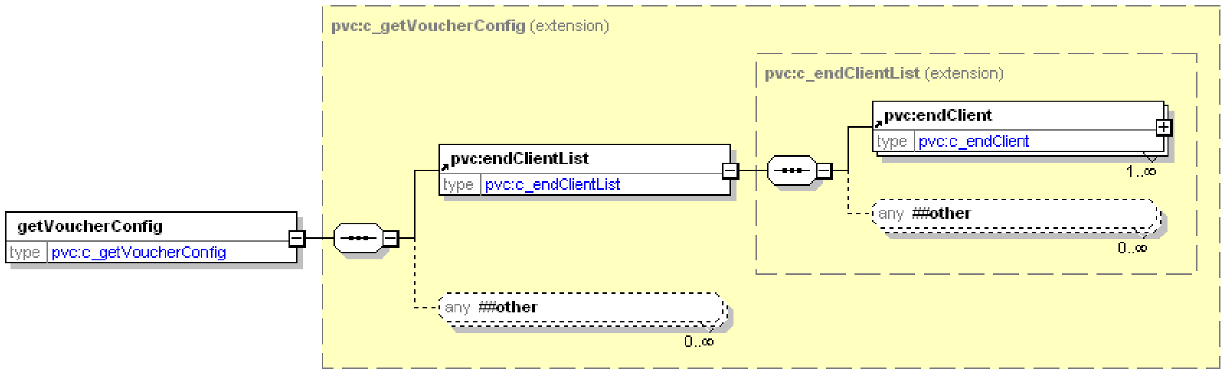


Table 40.28 getVoucherConfig Attributes

Attribute	Restrictions	Description

`clientType`	`type: t_clientTypes`
	`use: required`
		Client type; the type of client to which the command is directed; typically, an edge-server. Wildcards are not permitted.

`clientId`	`type: t_clientId`
	`use: required`
		Client identifier; the identifier of the client to which the command is directed; typically, an edge-server. Wildcards are not permitted.

Table 40.29 getVoucherConfig Sub-Elements

Element	Restrictions	Description

`endClientList`	`minOcc: 1`
	`maxOcc: 1`
		Contains the list of end-clients. See Table 40.30.

Table 40.30 endClientList Sub-Elements

Element	Restrictions	Description

`endClient`	`minOcc: 1`
	`maxOcc: ∞`
		Identifies an end-client. See Table 40.31.

Table 40.31 endClient Attributes

Attribute	Restrictions	Description

`endClientType`	`type: t_clientTypes`
	`use: required`
		End-client type; the type of end-client for which information is being requested. Wildcards are not permitted.

`endClientId`	`type: t_clientId`
	`use: required`
		End-client identifier; the identifier of the end-client for which information is being requested. The `S2S_all` and `S2S_default` wildcards are permitted.

NEW CLASS

40.14 setVoucherConfig Command

40.14.1 Command Description

This command is used by a host to set the voucher configuration for one or more end-clients. The command MUST only be used with point-to-point communications channels and MUST only be sent as a request. A `voucherConfigAck` command is generated in response to the `setVoucherConfig` command, indicating that the edge-server has received the request and will make a best-effort attempt to apply the request to the list of intended end-clients. Subsequently, as the results of the request become available, the edge-server MUST generate one or more `voucherStatusList` commands to report the results to the host.

Data sets, which are included in the `setVoucherConfig` command, overwrite any previous information related to the data sets. Data sets, which are not included, are not affected.

The `propertyId` attribute of the class-level element is used to identify the property for which configuration information is being set. The edge-server MUST only set configurations for end-clients associated with that property.

In addition to other event codes that the edge-server MAY report, the edge-server MAY report the following event codes, indicating that the requested action was not taken for an end-client.

- If the edge-server determines that an end-client is not registered for the property, the edge-server MUST report the error using event code `S2S_PVE001 Invalid End-Client For Property`.
- If the edge-server is unable to set the voucher configuration for an end-client, the edge-server MUST report the error using event code `S2S_PVE005 Unable To Set Voucher Configuration For End-Client`.

40.14.2 Attribute and Element Detail

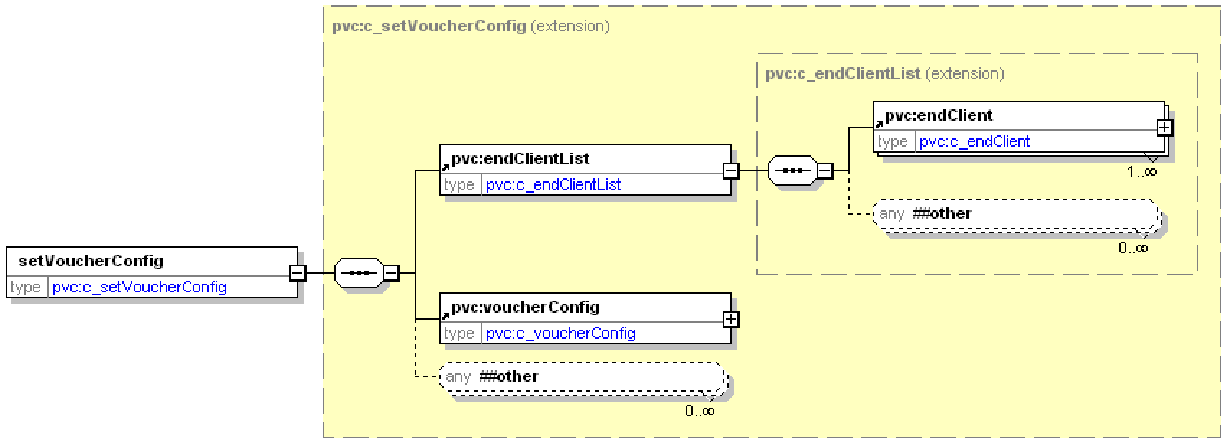


Table 40.32 setVoucherConfig Attributes (Sheet 1 of 2)

Attribute	Restrictions	Description

`clientType`	`type: t_clientTypes`
	`use: required`
		Client type; the type of client to which the command is directed; typically, an edge-server. Wildcards are not permitted.

NEW CLASS

Table 40.32 setVoucherConfig Attributes (Sheet 2 of 2)

Attribute	Restrictions	Description

`clientId`	`type: t_clientId`
	`use: required`
		Client identifier; the identifier of the client to which the command is directed; typically, an edge-server. Wildcards are not permitted.

Table 40.33 setVoucherConfig Sub-Elements

Element	Restrictions	Description

`endClientList`	`minOcc: 1`
	`maxOcc: 1`
		Contains the list of end-clients. See Table 40.34.

`voucherConfig`	`minOcc: 1`
	`maxOcc: 1`
		Contains voucher configuration parameters for the list of end-clients. See Section 40.8, voucherConfig Data Set.

Table 40.34 endClientList Sub-Elements

Element	Restrictions	Description

`endClient`	`minOcc: 1`
	`maxOcc: ∞`
		Identifies an end-client. See Table 40.35.

Table 40.35 endClient Attributes

Attribute	Restrictions	Description

`endClientType`	`type: t_clientTypes`
	`use: required`
		End-client type; the type of end-client being configured. Wildcards are not permitted.

`endClientId`	`type: t_clientId`
	`use: required`
		End-client identifier; the identifier of the end-client being configured. The `S2S_all` and `S2S_default` wildcards are permitted.

NEW CLASS

40.15 reqVoucherStatus Command

40.15.1 Command Description

This command is used by an edge-server to request the current voucher status information for one or more end-clients. The command MUST only be used with point-to-point communications channels and MUST only be sent as a request. A `voucherConfigAck` command is generated in response to the `reqVoucherStatus` command, indicating that the host has received the request and will make a best-effort attempt to provide the current voucher status for the list of intended end-clients. Subsequently, the host MUST generate one or more `setVoucherState` commands to report the information to the edge-server.

The `propertyId` attribute of the class-level element is used to identify the property for which voucher status information is being requested. The edge-server MUST only include voucher status information for end-clients associated with that property in its response.

In addition to other event codes that the host MAY report, the host MAY report the following event codes, indicating that the requested action was not taken for an end-client.

- If the host determines that an end-client is not registered for the property, the host MUST report the error using event code `S2S_PVE001 Invalid End-Client For Property`.
- If the host is unable to report the voucher status for an end-client, the host MUST report the error using event code `S2S_PVE002 Voucher Status Unavailable For End-Client`.

40.15.2 Attribute and Element Detail

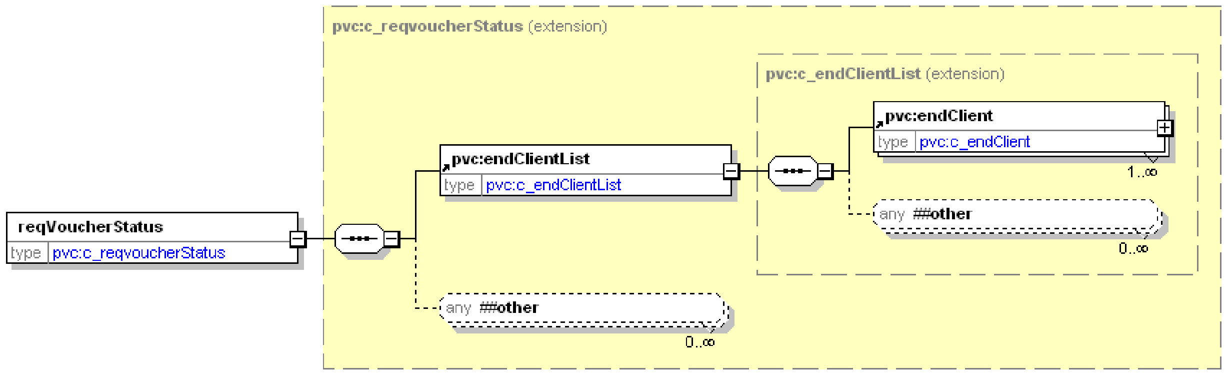


Table 40.36 reqVoucherStatus Attributes

Attribute	Restrictions	Description

`clientType`	`type: t_clientTypes`
	`use: required`
		Client type; the type of client from which the command was sent; typically, an edge-server. Wildcards are not permitted.

`clientId`	`type: t_clientId`
	`use: required`
		Client identifier; the identifier of the client from which the command was sent; typically, an edge-server. Wildcards are not permitted.

Table 40.37 reqVoucherStatus Sub-Elements

Element	Restrictions	Description

`endClientList`	`minOcc: 1`
	`maxOcc: 1`
		Contains the list of end-clients. See Table 40.38.

Table 40.38 endClientList Sub-Elements

Element	Restrictions	Description

`endClient`	`minOcc: 1`
	`maxOcc: ∞`
		Identifies an end-client. See Table 40.39.

Table 40.39 endClient Attributes

Attribute	Restrictions	Description

`endClientType`	`type: t_clientTypes`
	`use: required`
		End-client type; the type of end-client for which information is being requested. Wildcards are not permitted.

`endClientId`	`type: t_clientId`
	`use: required`
		End-client identifier; the identifier of the end-client for which information is being requested. The `S2S_all` and `S2S_default` wildcards are permitted.

NEW CLASS

40.16 voucherStatusList Command

40.16.1 Command Description

This command is used by an edge-server to report the current voucher status information for one or more end-clients. The command MUST only be used with point-to-point communications channels and MUST only be sent as a request. A `voucherConfigAck` command is generated in response to the `voucherStatusList` command, indicating that the host has received the command.

The `voucherStatusList` command is generated by an edge-server to report the results of the `getVoucherStatus`, `setVoucherState`, and `setVoucherConfig` commands. A `voucherStatusList` command MUST also be generated by an edge-server whenever the voucher status of an end-client changes for some other reason – for example, whenever the state of an end-client is changed locally at the end-client.

The `propertyId` attribute of the class-level element is used to identify the property for which voucher status information is being reported. The edge-server MUST only include voucher status information for end-clients associated with that property in this command.

In addition to other event codes that the host MAY report, the host MAY report the following event codes, indicating that no action was taken for an end-client.

- If the host determines that an end-client is not registered for the property, the host MUST report the error using event code `S2S_PVE001 Invalid End-Client For Property`.

40.16.2 Attribute and Element Detail

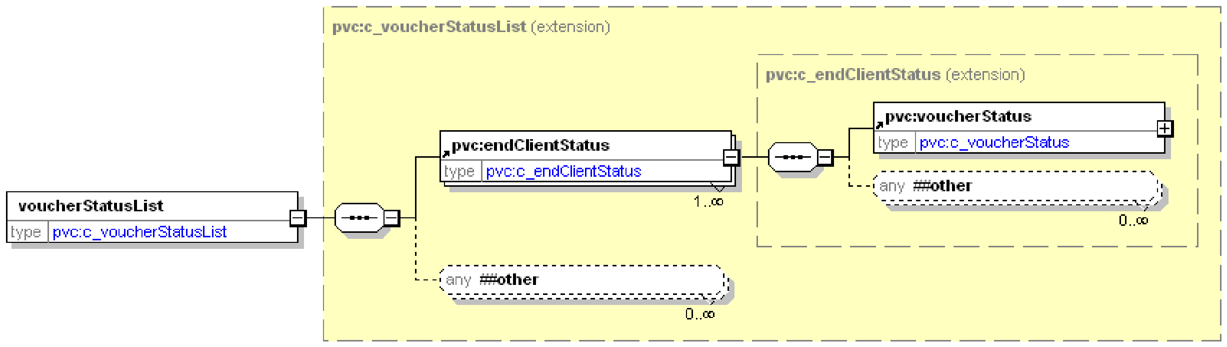

pvc:c_voucherStatusList (extension)
pvc:c_endClientStatus (extension)
pvc:voucherStatus
type pvc:c_voucherStatus
pvc:endClientStatus
type pvc:c_endClientStatus
1..∞
voucherStatusList
type pvc:c_voucherStatusList
any ##other
0..∞
any ##other
0..∞

Table 40.40 voucherStatusList Attributes

Attribute	Restrictions	Description
`clientType`	type: `t_clientTypes` use: required	Client type; the type of client from which the command was sent; typically, an edge-server. Wildcards are not permitted.
`clientId`	type: `t_clientId` use: required	Client identifier; the identifier of the client from which the command was sent; typically, an edge-server. Wildcards are not permitted.

Table 40.41 voucherStatusList Sub-Elements

Element	Restrictions	Description

`endClientStatus`	`minOcc: 1`
	`maxOcc: ∞`
		Contains the voucher status for an end-client. See Table 40.42.

Table 40.42 endClientStatus Attributes

Attribute	Restrictions	Description

`endClientType`	`type: t_clientTypes`
	`use: required`
		End-client type; the type of end-client for which information is being reported. Wildcards are not permitted.

`endClientId`	`type: t_clientId`
	`use: required`
		End-client identifier; the identifier of the end-client for which information is being reported. The `S2S_all` and `S2S_default` wildcards are permitted.

Table 40.43 endClientStatus Sub-Elements

Element	Restrictions	Description

`voucherStatus`	`minOcc: 1`
	`maxOcc: 1`
		Contains the voucher status for an end-client. See Section 40.9, voucherStatus Data Set.

NEW CLASS

40.17 reqVoucherConfig Command

40.17.1 Command Description

This command is used by an edge-server to request the current voucher configuration information for one or more end-clients. The command MUST only be used with point-to-point communications channels and MUST only be sent as a request. A `voucherConfigAck` command is generated in response to the `reqVoucherConfig` command, indicating that the host has received the request and will make a best-effort attempt to provide the current voucher configuration information for the list of intended end-clients. Subsequently, the host MUST generate one or more `setVoucherConfig` commands to report the information to the edge-server.

The `propertyId` attribute of the class-level element is used to identify the property for which configuration information is being requested. The host MUST only include configuration information for end-clients associated with that property in its response.

In addition to other event codes that the host MAY report, the host MAY report the following event codes, indicating that the requested action was not taken for an end-client.

- If the host determines that an end-client is not registered for the property, the host MUST report the error using event code `S2S_PVE001 Invalid End-Client For Property`.
- If the host is unable to report the voucher configuration for an end-client, the host MUST report the error using event code `S2S_PVE004 Voucher Configuration Not Available For End-Client`.

40.17.2 Attribute and Element Detail

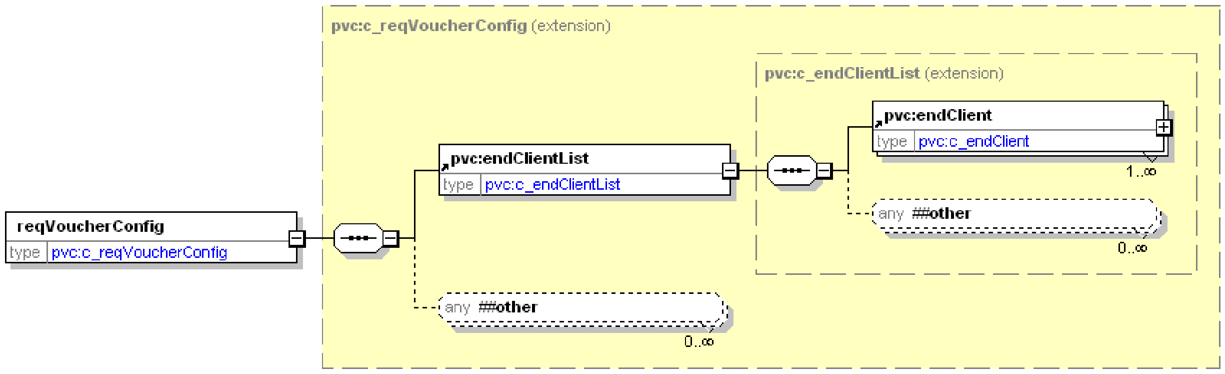


Table 40.44 reqVoucherConfig Attributes

Attribute	Restrictions	Description
`clientType`	type: `t_clientTypes` use: required	Client type; the type of client from which the command was sent; typically, an edge-server. Wildcards are not permitted.
`clientId`	type: `t_clientId` use: required	Client identifier; the identifier of the client from which the command was sent; typically, an edge-server. Wildcards are not permitted.

Table 40.45 reqVoucherConfig Sub-Elements

Element	Restrictions	Description

`endClientList`	`minOcc: 1`
	`maxOcc: 1`
		Contains the list of end-clients. See Table 40.46.

Table 40.46 endClientList Sub-Elements

Element	Restrictions	Description

`endClient`	`minOcc: 1`
	`maxOcc: ∞`
		Identifies an end-client. See Table 40.47.

Table 40.47 endClient Attributes

Attribute	Restrictions	Description

`endClientType`	`type: t_clientTypes`
	`use: required`
		End-client type; the type of end-client for which information is being requested. Wildcards are not permitted.

`endClientId`	`type: t_clientId`
	`use: required`
		End-client identifier; the identifier of the end-client for which information is being requested. The `S2S_all` and `S2S_default` wildcards are permitted.

NEW CLASS

40.18 voucherConfigList Command

40.18.1 Command Description

This command is used by an edge-server to report the current voucher configuration information for one or more end-clients. The command MUST only be used with point-to-point communications channels and MUST only be sent as a request. A `voucherConfigAck` command is generated in response to the `voucherConfigList` command, indicating that the host has received the command.

The `voucherConfigList` command is generated by an edge-server to report the results of the `getVoucherConfig` command. A `voucherConfigList` command MUST also be generated by an edge-server whenever the voucher configuration of an end-client changes for some other reason – for example, whenever the configuration of an end-client is changed locally at the end-client.

The `propertyId` attribute of the class-level element is used to identify the property for which configuration information is being reported. The edge-server MUST only include configuration information for end-clients associated with that property in this command.

In addition to other event codes that the host MAY report, the host MAY report the following event codes, indicating that no action was taken for an end-client.

- If the host determines that an end-client is not registered for the property, the host MUST report the error using event code `S2S_PVE001 Invalid End-Client For Property`.

40.18.2 Attribute and Element Detail

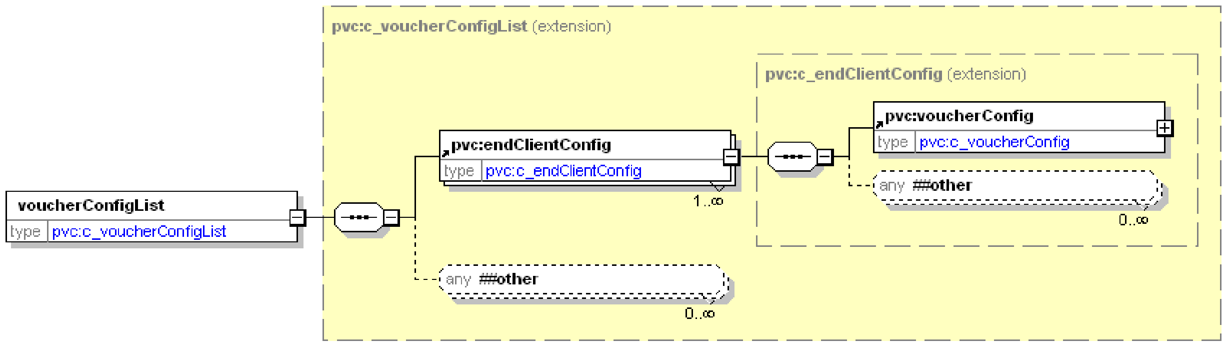

pvc:c_voucherConfigList (extension)
pvc:c_endClientConfig (extension)
pvc:voucherConfig
type pvc:c_voucherConfig
pvc:endClientConfig
type pvc:c_endClientConfig
1..∞
any ##other
0..∞
voucherConfigList
type pvc:c_voucherConfigList
any ##other
0..∞

Table 40.48 voucherConfigList Attributes

Attribute	Restrictions	Description

`clientType`	`type: t_clientTypes`
	`use: required`
		Client type; the type of client from which the command was sent; typically, an edge-server. Wildcards are not permitted.

`clientId`	`type: t_clientId`
	`use: required`
		Client identifier; the identifier of the client from which the command was sent; typically, an edge-server. Wildcards are not permitted.

Table 40.49 voucherConfigList Sub-Elements

Element	Restrictions	Description

`endClientConfig`	`minOcc: 1`
	`maxOcc: ∞`
		Contains the voucher configuration for an end-client. See Table 40.50.

Table 40.50 endClientConfig Attributes

Attribute	Restrictions	Description

`endClientType`	`type: t_clientTypes`
	`use: required`
		End-client type; the type of end-client for which information is being reported. Wildcards are not permitted.

`endClientId`	`type: t_clientId`
	`use: required`
		End-client identifier; the identifier of the end-client for which information is being reported. The `S2S_all` and `S2S_default` wildcards are permitted.

Table 40.51 endClientConfig Sub-Elements

Element	Restrictions	Description

`voucherConfig`	`minOcc: 1`
	`maxOcc: 1`
		Contains the voucher configuration for an end-client. See Section 40.8, voucherConfig Data Set.

NEW CLASS

40.19 voucherConfigAck Command

40.19.1 Command Description

This command is used by a host or edge-server to acknowledge the receipt of a request, indicating that the host or edge-server will make a best effort attempt to apply the request. The command MUST only be used with point-to-point communications channels and MUST only be sent as a response. The `voucherConfigAck` command is generated in response to the `getVoucherStatus`, `setVoucherState`, `getVoucherConfig`, `setVoucherConfig`, `reqVoucherStatus`, `voucherStatusList`, `reqVoucherConfig`, and `voucherConfigList` commands.

The `propertyId` attribute of the class-level element is used to identify the property for which a command is being acknowledged. The host MUST include the property specified in the original request in its acknowledgement.

40.19.2 Attribute and Element Detail

voucherConfigAck
type
pvc:c_voucherConfigAck

Table 40.52 voucherConfigAck Attributes

Attribute	Restrictions	Description
`clientType`	type: `t_clientTypes` use: required	Client type; the type of client referenced in the `clientType` attribute of the request. Wildcards are not permitted.
`clientId`	type: `t_clientId` use: required	Client identifier; the identifier of the client referenced in the `clientId` attribute of the request. Wildcards are not permitted.

NEW CLASS

40.20 getValidationIds Command

40.20.1 Command Description

This command is used by an edge-server to request validation identifiers for an end-client. The command MUST only be used with point-to-point communications channels and MUST only be sent as a request. A `validationIdList` command is generated in response to the `getValidationIds` command.

All vouchers that have been issued by the end-client MUST be acknowledged by the host before requests for new validation identifiers are generated. Requests for new validation identifiers MUST NOT be generated while the voucher functionality is disabled by the host.

Provided that all vouchers issued by the end-client have been acknowledged by the host and the voucher functionality for the end-client is enabled, the edge-server MUST generate a `getValidationIds` request under the following circumstances:

- When the number of validation identifiers stored for the end-client falls below the `minLevelValIds`,
- When the `valIdListRefresh` or `valIdListLife` time period expires, or
- When the voucher functionality is enabled by the host after being disabled.

Once the edge-server has determined that the validation identifiers need to be refreshed, the edge-server MUST make a best effort to retry the `getValidationIds` command at the frequency set in the `timeToLive` configuration parameter until a valid `validationIdList` command is received. Once the `valIdListLife` time period has expired, the edge-server MUST set the `validationIdsExpired` status attribute to `true`.

Provided that the `valIdListLife` time period has not expired, the edge-server may continue to allow the end-client to issue vouchers until all available validation identifiers have been consumed. However, after the voucher functionality is enabled by the host after being disabled, the edge-server MUST NOT allow the end-client to issue any vouchers until the validation identifiers have been refreshed—that is, the edge-server MUST treat the existing validation identifiers as if the `valIdListLife` time period had expired, setting the `validationIdsExpired` status attribute to `true`.

The `numValidationIds` attribute of the `getValidationIds` command MUST be set to the difference between the `maxValIds` configuration parameter and the `validationIdsRemaining` status attribute, but not less than 0 (zero). The `valIdListExpired` attribute MUST be set to the value of the `validationIdsExpired` status attribute. And, the `validationListId` attribute MUST be set to the value of the `validationListId` status attribute.

When the `allowVoucherIssue` configuration attribute is set to `false`, the edge-server MUST NOT generate any `getValidationIds` commands for the end-client.

In addition to other errors that the host MAY report, the following error conditions MAY be reported by the host, indicating that no action was taken:

- If the host determines that the `endClientType` is invalid, the host MUST report the error using error code `S2S_PVX002 Invalid End-Client Type`.
- If the host determines that the `endClientId` is invalid for the property, the host MUST report the error using error code `S2S_PVX003 Invalid End-Client for Property`.

More information about managing validation identifiers can be found in the G2S protocol.

40.20.1.1 Duplicate Commands

A `getValidationIds` command is not considered to be logically equivalent to any previous `getValidationIds` command. The host MUST treat each `getValidationIds` command as if it was logically unique.

40.20.2 Attribute and Element Detail

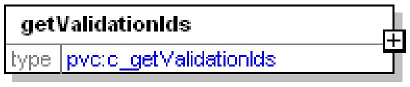


Table 40.53 getValidationIds Attributes

Attribute	Restrictions	Description

`clientType`	`type: t_clientTypes`
	`use: required`
		Client type; the type of client from which the command was sent; typically, an edge-server. Wildcards are not permitted.

`clientId`	`type: t_clientId`
	`use: required`
		Client identifier; the identifier of the client from which the command was sent; typically, an edge-server. Wildcards are not permitted.

`endClientType`	`type: t_clientTypes`
	`use: required`
		End-client type; type of end-client requiring validation identifiers. Wildcards are not permitted.

`endClientId`	`type: t_clientId`
	`use: required`
		End-client identifier; client identifier of the end-client requiring validation identifiers. Wildcards are not permitted.

`validationListId`	`type: t_validationListId`
	`use: required`
		The `validationListId` received in the last `validationIdList` command for the end-client; set to `0` (zero) if no such command has been received.

`numValidationIds`	`type: xs:int`
	`use: optional`
	`default: 0`
	`minIncl: 0`
		The number of validation identifiers required to restore the number of available validation identifiers to the `maxValIds` level, but not less than `0` (zero).

`valIdListExpired`	`type: xs:boolean`
	`use: optional`
	`default: false`
		Indicates whether the `valIdListLife` time period is considered to have expired; set to `true` if no validation identifiers have ever been received for the end-client.

`configurationId`	`type: t_configurationId`
	`use: optional`
	`default: 0`
		Configuration identifier; used by the edge-server to determine whether the correct configuration is in use by the end-client.

NEW CLASS

40.21 validationIdList Command

40.21.1 Command Description

This command is used by a host to update the set of validation identifiers for an end-client. The command MUST only be used with point-to-point communications channels and MUST only be sent as a response. The `validationIdList` command is generated in response to a `getValidationIds` command.

The host may use the `deleteCurrent` attribute to indicate that all remaining validation identifiers should be discarded before adding the new validation identifiers provided in the `validationIdList` command. If the `deleteCurrent` attribute is set to `true`, all remaining validation identifiers MUST be discarded. If the `deleteCurrent` attribute is set to `false`, all remaining validation identifiers MUST be retained. Validation identifiers MUST be consumed in the order provided. Any new validation identifiers provided in the `validationIdList` command MUST be consumed after any validation identifiers retained for the end-client. If any validation identifiers are found in the `validationIdList` command are already recorded for the end-client, the ordering of the validation identifiers MUST NOT be changed, however, the seed values associated with the validation identifiers MUST be updated.

When the `deleteCurrent` attribute is set to `false`, the host MUST include the number of validation identifiers specified in the `numValidationIds` attribute. When the `deleteCurrent` attribute is set to `true`, the host MUST include the number of validation identifiers specified in the `maxValIds` configuration parameter.

If any of the new validation identifiers or seed values cannot be used – for example, a validation identifier includes non-numeric characters or the host provided more validation identifiers than required – the entire new set of validation identifiers MUST NOT be used – the edge-server MUST generate event `S2S_PVE010 Validation Data Error` and the edge-server MUST NOT use any of the new validation identifiers, delete any remaining validation identifiers, or update the `validationListId`, `validationIdsRemaining`, or `validationIdsExpired` status attributes.

After successfully recording a new set of validation identifiers, the edge-server MUST update the voucher status to indicate latest `validationListId` received from the host and the correct number of `validationIdsRemaining`; the edge-server MUST set `validationIdsExpired` status attribute to `false` and restart any timers associated with the `valIdsListRefresh` and `valIdsListLife` time periods; and, the edge-server MUST generate event `S2S_PVE011 Validation Data Updated`.

40.21.1.1 Duplicate Commands

A `validationIdList` command is not considered to be logically equivalent to any previous `validationIdList` command. The host MUST treat each `validationIdList` command as if it was logically unique.

NEW CLASS

40.21.2 Attribute and Element Detail

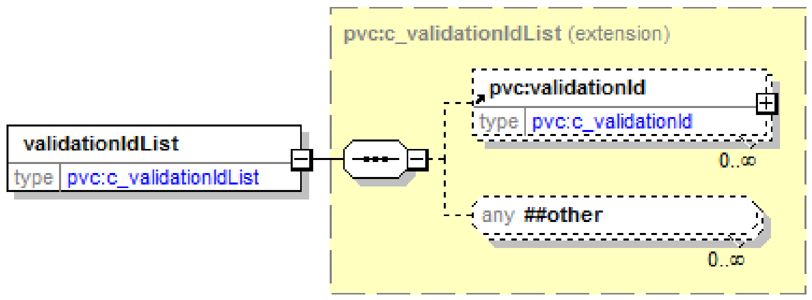


Table 40.54 validationIdList Attributes

Attribute	Restrictions	Description

`clientType`	`type: t_clientTypes`
	`use: required`
		Client type; the type of client to which the command is directed; typically, an edge-server. Wildcards are not permitted.

`clientId`	`type: t_clientId`
	`use: required`
		Client identifier; the identifier of the client to which the command is directed; typically, an edge-server. Wildcards are not permitted.

`endClientType`	`type: t_clientTypes`
	`use: required`
		End-client type; type of end-client requiring validation identifiers. Wildcards are not permitted.

`endClientId`	`type: t_clientId`
	`use: required`
		End-client identifier; client identifier of the end-client requiring validation identifiers. Wildcards are not permitted.

`validationListId`	`type: t_validationListId`
	`use: required`
		Host-assigned identifier for the set of validation identifiers.

`deleteCurrent`	`type: xs:boolean`
	`use: optional`
	`default: false`
		Indicates whether all remaining validation identifiers for the end-client should be discarded.

Table 40.55 validationIdList Sub-Elements

Element	Restrictions	Description

`validationId`	`minOcc: 0`
	`maxOcc: ∞`
		Contains a validation identifier and seed value. See Table 40.56.

Table 40.56 validationId Attributes (Sheet 1 of 2)

Attribute	Restrictions	Description

`validationId`	`type: t_validationId`
	`use: required`
		Validation identifier.

Table 40.56 validationId Attributes (Sheet 2 of 2)

Attribute	Restrictions	Description
validationSeed	type: t_validationSeed use: required	Manual authentication seed value.

NEW CLASS

40.22 issueVoucher Command

40.22.1 Command Description

This command is used by an edge-server to report that a voucher has been issued by an end-client. The command MUST only be used with point-to-point communications channels and MUST only be sent as a request. An `issueVoucherAck` command is generated in response to the `issueVoucher` command.

The `issueVoucher` command SHOULD be generated as soon as the end-client is irreversibly committed to the voucher issuance operation and the associated credits have been removed from the credit meter. The end-client 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 `endClientException` attribute of the `issueVoucher` command to `1` (one). However, reporting any such errors SHOULD NOT delay the reporting of the voucher issuance operation.

The edge-server MUST make a best effort to retry the `issueVoucher` command at the frequency set in the `timeToLive` configuration parameter until a valid `issueVoucherAck` command is received.

When issuing vouchers for non-cashable credits, the following rules MUST be applied:

- If the `allowNonCashOut` configuration parameter is set to `true`:
    - If there is no expiration associated with the non-cashable credits, the end-client MUST produce the voucher for the non-cashable credits and the `expireNonCash` configuration parameter MUST be used to determine the expiration period for the voucher (not the expiration date/time for the non-cashable credits).
    - If there is an expiration associated with the non-cashable credits and the current date/time is the same as or prior to that expiration, the end-client MUST produce the voucher for the non-cashable credits.
    - If there is an expiration associated with the non-cashable credits and the current date/time is after that expiration, the end-client MUST NOT produce a voucher for the non-cashable credits.
- If the `allowNonCashOut` configuration parameter is set to `false`, the end-client MUST NOT produce a voucher for the non-cashable credits.

When the `combineCashableOut` configuration attribute is set to `true`, the end-client MUST convert any promotional credits to cashable credits when issuing a voucher—a single combined voucher for cashable credits MUST be issued. When the `combineCashableOut` configuration attribute is set to `false`, the end-client MUST NOT convert any promotional credits to cashable credits when issuing a voucher—separate vouchers for cashable and promotional credits MUST be issued, if necessary.

The `idReaderType`, `idNumber`, and `playerId` attributes of the `issueVoucher` command MUST specify the player currently identified by the ID reader associated with voucher the functionality.

The `propertyId` attribute of the class-level element is used to identify the property for which a voucher issuance is being reported. The edge-server MUST only report voucher issuances for end-clients associated with that property in this command.

More information about recording voucher issuances can be found in the G2S protocol.

40.22.1.1 Duplicate Commands

An `issueVoucher` command is considered to be logically equivalent to a previous `issueVoucher` command if the host detects that the `transactionId` associated with the request was reported in a previous `issueVoucher` command for the same end-client. In such cases, the host MUST generate a logically equivalent `issueVoucherAck` command in response to the `issueVoucher` command.

The host SHOULD also verify that the `validationId` has not been reported in a previous `issueVoucher` command. If the `validationId` has been previously reported and is not a duplicate, the host SHOULD also, if possible, alert the operator to the duplicate voucher.

40.22.2 Attribute and Element Detail

issueVoucher
type
pvc:c_issueVoucher

Table 40.57 issueVoucher Attributes (Sheet 1 of 3)

Attribute	Restrictions	Description

`clientType`	`type: t_clientTypes`
	`use: required`
		Client type; the type of client from which the command was sent; typically, an edge-server. Wildcards are not permitted.

`clientId`	`type: t_clientId`
	`use: required`
		Client identifier; the identifier of the client from which the command was sent; typically, an edge-server. Wildcards are not permitted.

`endClientType`	`type: t_clientTypes`
	`use: required`
		End-client type; type of end-client that issued the voucher. Wildcards are not permitted.

`endClientId`	`type: t_clientId`
	`use: required`
		End-client identifier; client identifier of the end-client that issued the voucher. Wildcards are not permitted.

`transactionId`	`type: t_transactionId`
	`use: required`
		Transaction identifier. (`issueVoucher.transactionId`)

`idReaderType`	`type: t_idReaderTypes`
	`use: optional`
	`default: S2S_none`
		Type of ID reader used to identify the player. (`issueVoucher.idReaderType`)

`idNumber`	`type: t_idNumber`
	`use: optional`
	`default: <empty>`
		ID number associated with the player. (`issueVoucher.idNumber`)

`playerId`	`type: t_playerId`
	`use: optional`
	`default: <empty>`
		Player identifier. (`issueVoucher.playerId`)

`validationId`	`type: t_validationId`
	`use: required`
		Validation identifier. (`issueVoucher.validationId`)

NEW CLASS

Table 40.57 issueVoucher Attributes (Sheet 2 of 3)

Attribute	Restrictions	Description

`voucherAmt`	`type: t_meterValue`
	`use: required`
		Voucher amount. (`issueVoucher.vouchetAmt`)

`creditType`	`type: t_creditTypes`
	`use: required`
		Type of credits; cashable, promotional, or non-cashable. (`issueVoucher.creditType`)

`voucherSource`	`type: t_voucherSources`
	`use: optional`
	`default: S2S_endClient`
		Indicates the source of the voucher; end-client or system. (`issueVoucher.voucherSequence`)

`largeWin`	`type: xs:boolean`
	`use: optional`
	`default: false`
		Indicates whether the voucher was issued because the amount won exceeded the end-client's large win limit. (`issueVoucher.largeWin`)

`voucherSequence`	`type: xs:int`
	`use: optional`
	`default: 0`
	`minIncl: 0`
		The issuing end-client's internal voucher sequence number printed on the voucher. (`issueVoucher.voucherSequence`)

`expireCredits`	`type: xs:boolean`
	`use: optional`
	`default: false`
		Indicates whether non-cashable credits have an associated expiration date/time; ignored when `creditType` is not set to `S2S_nonCashable`. (`issueVoucher.expireCredits`)

`expireDateTime`	`type: t_dateTime`
	`use: optional`
	`default: 2000-01-01T00:00:00.000-00:00`
		Expiration date/time associated with non-cashable credits; ignored when `creditType` is not set to `S2S_nonCashable` or `expireCredits` is set to `false`. (`issueVoucher.expireDateTime`)

`transferAmt`	`type: t_meterValue`
	`use: optional`
	`default: 0`
		Actual amount transferred. (`issueVoucher.transferAmt`)

`transferDateTime`	`type: t_dateTime`
	`use: required`
		Date/time that the transfer record was updated. (`issueVoucher.transferDateTime`)

`expireDays`	`type: xs:int`
	`use: optional`
	`default: -1`
	`minIncl: -1`
		Number of days before the voucher expires; ignored if `expireCredits` is set to true; `-1` (negative one) indicates that there is no expiration period. (`issueVoucher.expireDays`)

`endClientAction`	`type: t_voucherClientActions`
	`use: required`
		Indicates whether the end-client issued, stacked or returned the voucher. (`issueVoucher.egmAction`)

NEW CLASS

Table 40.57 issueVoucher Attributes (Sheet 3 of 3)

Attribute	Restrictions	Description

`endClientException`	`type: t_voucherClientExcs`
	`use: optional`
	`default: 0`
		End-client exception code; `0` (zero) indicates no end-client exceptions. (`issueVoucher.egmException`)

NEW CLASS

40.23 issueVoucherAck Command

40.23.1 Command Description

This command is used by a host to acknowledge the issuance of a voucher by an end-client. The command MUST only be used with point-to-point communications channels and MUST only be sent as a response. The `issueVoucherAck` command is generated in response to an `issueVoucher` command.

The host MUST make a best effort to acknowledge the `issueVoucher` command. Failure to acknowledge the `issueVoucher` command will cause the edge-server to retry the `issueVoucher` command indefinitely.

The `propertyId` attribute of the class-level element is used to identify the property for which a voucher issuance is being acknowledged. The host MUST only include acknowledgements for end-clients associated with that property in this command.

40.23.1.1 Duplicate Commands

An `issueVoucherAck` command is considered to be logically equivalent to a previous `issueVoucherAck` command if the edge-server detects that the voucher issuance request associated with the `transactionId` was already acknowledged—that is, the state of the transaction request is no longer `S2S_issueSent`. Duplicate `issueVoucherAck` commands can be ignored.

40.23.2 Attribute and Element Detail

issueVoucherAck
type
pvc:c_issueVoucherAck

Table 40.58 issueVoucherAck Attributes

Attribute	Restrictions	Description
`clientType`	type: `t_clientTypes` use: required	Client type; the type of client to which the command is directed; typically, an edge-server. Wildcards are not permitted.
`clientId`	type: `t_clientId` use: required	Client identifier; the identifier of the client to which the command is directed; typically, an edge-server. Wildcards are not permitted.
`endClientType`	type: `t_clientTypes` use: required	End-client type; type of end-client that issued the voucher. Wildcards are not permitted.
`endClientId`	type: `t_clientId` use: required	End-client identifier; client identifier of the end-client that issued the voucher. Wildcards are not permitted.
`transactionId`	type: `t_transactionId` use: required	Transaction identifier. (`issueVoucherAck.transactionId`)

NEW CLASS

40.24 redeemVoucher Command

40.24.1 Command Description

This command is used by an edge-server to request authorization for an end-client to redeem a voucher. The command MUST only be used with point-to-point communications channels and MUST only be sent as a request. An `authorizeVoucher` command is generated in response to the `redeemVoucher` command.

If the request is not authorized within the time period specified in the `voucherHoldTime` configuration attribute, the end-client MUST return the voucher and the edge-server MUST generate a `commitVoucher` command indicating that the voucher was returned due to a timeout (`endClientException = "5"`). If an `authorizeVoucher` command is received after the voucher has been returned (or, in general, at any time a voucher is not being held in escrow), the edge-server MUST ignore the `authorizeVoucher` command.

While waiting for the `voucherHoldTime` to expire, the end-client MUST make a best effort to retry the `redeemVoucher` command at the frequency set in the `timeToLive` configuration attribute until a valid `authorizeVoucher` command is received.

After generating a `redeemVoucher` command and the voucher has stacked or returned, the edge-server MUST always generate a `commitVoucher` command to report the final disposition of the voucher redemption request. Even if the edge-server does not receive an `authorizeVoucher` command or receives an error in response to the `redeemVoucher` command, the edge-server MUST still generate a `commitVoucher` command for the host to confirm the outcome of the redemption request.

The `idReaderType`, `idNumber`, and `playerId` attributes MUST specify the player currently identified by the ID reader device associated with the voucher functionality.

When the `allowVoucherRedeem` configuration attribute is set to `false`, the edge-server MUST NOT generate any `redeemVoucher` commands.

The `propertyId` attribute of the class-level element is used to identify the property for which a voucher redemption is being requested. The edge-server MUST only include voucher redemption requests for end-clients associated with that property in this command.

In addition to other errors that the host MAY report, the following error conditions MAY be reported by the host, indicating that no action was taken, in which case, the redemption MUST be aborted with the `endClientException` attribute set to `2`, indicating a redemption error from host – voucher returned:

- If the host determines that the `endClientType` is invalid, the host MUST report the error using error code `S2S_PVX002 Invalid End-Client Type`.
- If the host determines that the `endClientId` is invalid for the property, the host MUST report the error using error code `S2S_PVX003 Invalid End-Client for Property`.
- If the host determines that the `idReaderType` is invalid, the host MUST report the error using error code `S2S_GBX034 Invalid ID Reader Type`.

More information about voucher redemptions can be found in the G2S protocol.

NEW CLASS

40.24.1.1 Duplicate Commands

A `redeemVoucher` command is considered to be logically equivalent to a previous `redeemVoucher` command if the host detects that the `transactionId` associated with the redemption request was reported in a previous `redeemVoucher` or `commitVoucher` command for the same end-client. In such cases, the host MUST generate a logically equivalent `authorizeVoucher` command in response to the `redeemVoucher` command.

40.24.2 Attribute and Element Detail

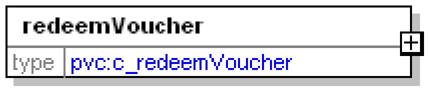


Table 40.59 redeemVoucher Attributes

Attribute	Restrictions	Description

`clientType`	`type: t_clientTypes`
	`use: required`
		Client type; the type of client from which the command was sent; typically, an edge-server. Wildcards are not permitted.

`clientId`	`type: t_clientId`
	`use: required`
		Client identifier; the identifier of the client from which the command was sent; typically, an edge-server. Wildcards are not permitted.

`endClientType`	`type: t_clientTypes`
	`use: required`
		End-client type; type of end-client that originated the redemption request. Wildcards are not permitted.

`endClientId`	`type: t_clientId`
	`use: required`
		End-client identifier; client identifier of the end-client that originated the redemption request. Wildcards are not permitted.

`transactionId`	`type: t_transactionId`
	`use: required`
		Transaction identifier. (`redeemVoucher.transactionId`)

`idReaderType`	`type: t_idReaderTypes`
	`use: optional`
	`default: S2S_none`
		Type of ID reader used to identify the player. (`redeemVoucher.idReaderType`)

`idNumber`	`type: t_idNumber`
	`use: optional`
	`default: <empty>`
		ID number associated with the player. (`redeemVoucher.idNumber`)

`playerId`	`type: t_playerId`
	`use: optional`
	`default: <empty>`
		Player identifier. (`redeemVoucher.playerId`)

`validationId`	`type: t_validationId`
	`use: required`
		Validation identifier. (`redeemVoucher.validationId`)

NEW CLASS

40.25 authorizeVoucher Command

40.25.1 Command Description

This command is used by a host to authorize (or deny) a voucher redemption request for an end-client. The command MUST only be used with point-to-point communications channels and MUST only be sent as a response. 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 `creditType`, `voucherSource`, `largeWin`, `voucherSequence`, `expireCredits`, and `expireDateTime` attributes MUST be set to the semantically correct values for the voucher being redeemed.

To deny redemption of a voucher, the host MUST set the `voucherAmt` to `0` (zero) and MUST set the `hostException` attribute to a non-zero value indicating the reason for denial. The `creditType`, `voucherSource`, `largeWin`, `voucherSequence`, `expireCredits`, and `expireDateTime` attributes are not relevant and may be set to any syntactically correct values.

The host can set the `voucherSource` attribute to `S2S_systemIssued` to indicate that the voucher was issued by the system (typically, for promotional purposes) or `S2S_endClientIssued` to indicate the voucher was issued by an end-client, such as an EGM or kiosk. This feature can be used in jurisdictions where the expense for end-client-issued vouchers is deducted at the time of redemption. In such situations, system-issued vouchers are not deductible and, therefore, are accounted for separately from end-client-issued vouchers. In jurisdictions where this is not an issue, all vouchers can be redeemed and accounted for as end-client-issued vouchers.

The host may use the `hostAction` attribute to force an end-client to stack a voucher that is not valid or to force an end-client to return the voucher following a valid redemption. If the host authorizes redemption and the end-client is unable to redeem the voucher for any reason, the end-client MUST return 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:

- `S2S_endClientAction` tells the end-client to perform its normal action of stacking or returning a voucher; for example, stacking a redeemed voucher and returning all others.
- `S2S_stack` tells the end-client to stack a voucher following successful completion of the `authorizeVoucher` command.
    - If the host does not authorize redemption (i.e. `hostException` is set to a non-zero value), the end-client MUST still stack the voucher, if possible.
    - If the host authorizes redemption of the voucher and the end-client is unable to stack the voucher for any reason, the end-client MUST NOT redeem the voucher.
    - If the host authorizes redemption of the voucher and the end-client is unable to redeem the voucher for any reason, the end-client MUST NOT stack the voucher.
- `S2S_return` tells the end-client to return the voucher regardless of whether it was successfully redeemed or not. If `hostAction` is set to `S2S_return`, the end-client MUST NOT stack the voucher under any circumstances.

The `endClientAction` attribute indicates the final disposition of the voucher—that is, whether the voucher was stacked or returned. If the voucher was stacked by the end-client, the `endClientAction` attribute MUST be set to `S2S_redeemed`. Otherwise, if the voucher was returned (not stacked) by the end-client, the endClientAction attribute MUST be set to S2S_returned. The endClientAction attribute MUST be set based on the actual action performed by the end-client, not the action requested by the host in the hostAction attribute.

When a voucher redemption is authorized, the host MUST record that a redemption request is pending for the voucher. Until a commitVoucher command is received or the status of the voucher is manually reset, additional redemptions MUST NOT be permitted for that voucher by the host.

After the end-client has transferred any required credits to the credit meter and the voucher has been stacked or returned, the edge-server MUST generate a commitVoucher command. In all cases, following the generation of a redeemVoucher command, the edge-server MUST generate a commitVoucher command to report the final results of the voucher redemption request, even if no funds were transferred.

The propertyId attribute of the class-level element is used to identify the property for which a voucher redemption is being authorized or denied. The host MUST only include authorizations and denials for end-clients associated with that property in this command.

40.25.1.1 Duplicate Commands

An authorizeVoucher command is considered to be logically equivalent to a previous authorizeVoucher command if the edge-server detects that the redemption request associated with the transactionId was already authorized or denied—that is, the state of the redemption request is no longer S2S_redeemSent. Duplicate authorizeVoucher commands MUST be ignored.

40.25.2 Attribute and Element Detail

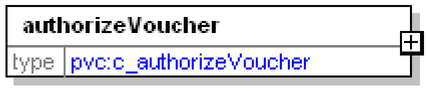


Table 40.60 authorizeVoucher Attributes (Sheet 1 of 2)

Attribute	Restrictions	Description
clientType	type: t_clientTypes use: required	Client type; the type of client to which the command is directed; typically, an edge-server. Wildcards are not permitted.
clientId	type: t_clientId use: required	Client identifier; the identifier of the client to which the command is directed; typically, an edge-server. Wildcards are not permitted.
endClientType	type: t_clientTypes use: required	End-client type; type of end-client that originated the redemption request. Wildcards are not permitted.
endClientId	type: t_clientId use: required	End-client identifier; client identifier of the end-client that originated the redemption request. Wildcards are not permitted.
transactionId	type: t_transactionId use: required	Transaction identifier. (authorizeVoucher.transactionId)

NEW CLASS

Table 40.60 authorizeVoucher Attributes (Sheet 2 of 2)

Attribute	Restrictions	Description

`validationId`	`type: t_validationId`
	`use: required`
		Validation identifier. (`authorizeVoucher.validationId`)

`voucherAmt`	`type: t_meterValue`
	`use: required`
		Voucher amount. (`authorizeVoucher.vouchetAmt`)

`creditType`	`type: t_creditTypes`
	`use: required`
		Type of credits; cashable, promotional, or non-cashable. (`authorizeVoucher.creditType`)

`voucherSource`	`type: t_voucherSources`
	`use: optional`
	`default: S2S_endClient`
		Indicates the source of the voucher; end-client or system. (`authorizeVoucher.voucherSequence`)

`largeWin`	`type: xs:boolean`
	`use: optional`
	`default: false`
		Indicates whether the voucher was issued because the amount won exceeded the end-client's large win limit. (`authorizeVoucher.largeWin`)

`voucherSequence`	`type: xs:int`
	`use: optional`
	`default: 0`
	`minIncl: 0`
		The issuing end-client's internal voucher sequence number printed on the voucher. (`authorizeVoucher.voucherSequence`)

`expireCredits`	`type: xs:boolean`
	`use: optional`
	`default: false`
		Indicates whether non-cashable credits have an associated expiration date/time; ignored when `creditType` is not set to `S2S_nonCashable`. (`authorizeVoucher.expireCredits`)

`expireDateTime`	`type: t_dateTime`
	`use: optional`
	`default: 2000-01-01T00:00:00.000-00:00`
		Expiration date/time associated with non-cashable credits; ignored when `creditType` is not set to `S2S_nonCashable` or `expireCredits` is set to `false`. (`authorizeVoucher.expireDateTime`)

`hostAction`	`type: t_voucherHostActions`
	`use: optional`
	`default: S2S_endClientAction`
		Indicates whether the host prefers the voucher to be stacked, returned, or that the end-client determines the appropriate action. (`authorizeVoucher.hostAction`)

`hostException`	`type: t_voucherHostExcs`
	`use: optional`
	`default: 0`
		Exception code set by host. (`authorizeVoucher.hostException`)

NEW CLASS

40.26 commitVoucher Command

40.26.1 Command Description

This command is used by an edge-server to report the final results of a voucher redemption request to the host. The command MUST NOT be generated until all payments associated with the voucher redemption request have been made. The command MUST only be used with point-to-point communications channels and MUST only be sent as a request. A `commitVoucherAck` command is generated in response to the `commitVoucher` command.

The `commitVoucher` command is generated regardless of whether the redemption was successful. If unsuccessful, the `transferAmt` attribute MUST be set to `0` (zero) and the `endClientException` MUST be set to a non-zero value. If successful, the `transferAmt` attribute MUST be set to the actual amount transferred and the `endClientException` attribute MUST be set to `0` (zero). If the redemption was unsuccessful, the host MUST reset the status of the voucher so that it can be redeemed elsewhere.

The edge-server MUST make a best effort to retry the `commitVoucher` command at the frequency specified in the `timeToLive` configuration parameter for the end-client until a valid `commitVoucherAck` command is received.

The `propertyId` attribute of the class-level element is used to identify the property for which the final results of the redemption request are being reported. The edge-server MUST only include results for end-clients associated with that property in this command.

40.26.1.1 Duplicate Commands

A `commitVoucher` command is considered to be logically equivalent to a previous `commitVoucher` command if the host detects that the `transactionId` associated with the command was reported in a previous `commitVoucher` command for the same end-client. In such cases, the host MUST generate a logically equivalent `commitVoucherAck` command in response to the `commitVoucher` command.

40.26.2 Attribute and Element Detail

commitVoucher
type
pvc:c_commitVoucher

Table 40.61 commitVoucher Attributes (Sheet 1 of 3)

Attribute	Restrictions	Description

`clientType`	`type: t_clientTypes`
	`use: required`
		Client type; the type of client from which the command was sent; typically, an edge-server. Wildcards are not permitted.

`clientId`	`type: t_clientId`
	`use: required`
		Client identifier; the identifier of the client from which the command was sent; typically, an edge-server. Wildcards are not permitted.

`endClientType`	`type: t_clientTypes`
	`use: required`
		End-client type; type of end-client that originated the redemption request. Wildcards are not permitted.

NEW CLASS

Table 40.61 commitVoucher Attributes (Sheet 2 of 3)

Attribute	Restrictions	Description
`endClientId`	type: `t_clientId` use: required	End-client identifier; client identifier of the end-client that originated the redemption request. Wildcards are not permitted.
`transactionId`	type: `t_transactionId` use: required	Transaction identifier; assigned by the end-client. (`commitVoucher.transactionId`)
`validationId`	type: `t_validationId` use: required	Validation identifier. (`commitVoucher.validationId`)
`voucherAmt`	type: `t_meterValue` use: required	Voucher amount. (`commitVoucher.vouchetAmt`)
`creditType`	type: `t_creditTypes` use: required	Type of credits; cashable, promotional, or non-cashable. (`commitVoucher.creditType`)
`voucherSource`	type: `t_voucherSources` use: optional default: S2S_endClient	Indicates the source of the voucher; end-client or system. (`commitVoucher.voucherSequence`)
`largeWin`	type: `xs:boolean` use: optional default: false	Indicates whether the voucher was issued because the amount won exceeded the end-client's large win limit. (`commitVoucher.largeWin`)
`voucherSequence`	type: `xs:int` use: optional default: 0 minIncl: 0	The issuing end-client's internal voucher sequence number printed on the voucher. (`commitVoucher.voucherSequence`)
`expireCredits`	type: `xs:boolean` use: optional default: false	Indicates whether non-cashable credits have an associated expiration date/time; ignored when `creditType` is not set to `S2S_nonCashable`. (`commitVoucher.expireCredits`)
`expireDateTime`	type: `t_dateTime` use: optional default: 2000-01-01T00:00:00.000-00:00	Expiration date/time associated with non-cashable credits; ignored when `creditType` is not set to `S2S_nonCashable` or `expireCredits` is set to `false`. (`commitVoucher.expireDateTime`)
`transferAmt`	type: `t_meterValue` use: optional default: 0	Actual amount transferred. (`commitVoucher.transferAmt`)
`transferDateTime`	type: `t_dateTime` use: required	Date/time that the transaction was committed. (`commitVoucher.transferDateTime`)
`endClientAction`	type: `t_voucherClientActions` use: required	Indicates whether the end-client stacked or returned the voucher. (`commitVoucher.egmAction`)

Table 40.61 commitVoucher Attributes (Sheet 3 of 3)

Attribute	Restrictions	Description

`endClientException`	`type: t_voucherClientExcs`
	`use: optional`
	`default: 0`
		End-client exception code; `0` (zero) indicates no end-client exceptions. (`commitVoucher.egmException`)

NEW CLASS

40.27 commitVoucherAck Command

40.27.1 Command Description

This command is used by a host to acknowledge receipt of the final results of a voucher redemption request. The command MUST only be used with point-to-point communications channels and MUST only be sent as a response. The `commitVoucherAck` command is generated in response to a `commitVoucher` command.

The host must make a best effort to acknowledge `commitVoucher` commands. Class-specific application-level error codes MUST NOT be used. Failure to acknowledge the commands may cause loss of voucher functionality.

The `propertyId` attribute of the class-level element is used to identify the property for which the receipt of the final results of a voucher redemption request is being acknowledged. The host MUST only include acknowledgements for end-clients associated with that property in this command.

40.27.1.1 Duplicate Commands

A `commitVoucherAck` command is considered to be logically equivalent to a previous `commitVoucherAck` command if the edge-server detects that the final result of the redemption associated with the `transactionId` has already been acknowledged—that is, the state of the transaction is no longer `S2S_commitSent`. Duplicate `commitVoucherAck` commands can be ignored.

40.27.2 Attribute and Element Detail

commitVoucherAck
type
pvc:c_commitVoucherAck

Table 40.62 commitVoucherAck Attributes

Attribute	Restrictions	Description
`clientType`	type: `t_clientTypes` use: required	Client type; the type of client to which the command is directed; typically, an edge-server. Wildcards are not permitted.
`clientId`	type: `t_clientId` use: required	Client identifier; the identifier of the client to which the command is directed; typically, an edge-server. Wildcards are not permitted.
`endClientType`	type: `t_clientTypes` use: required	End-client type; type of end-client that originated the redemption request. Wildcards are not permitted.
`endClientId`	type: `t_clientId` use: required	End-client identifier; client identifier of the end-client that originated the redemption request. Wildcards are not permitted.
`transactionId`	type: `t_transactionId` use: required	Transaction identifier. (`commitVoucherAck.transactionId`)

NEW CLASS

40.28 voucherUpdate Command

40.28.1 Command Description

This command is used by a host to broadcast updates to voucher information to clients that have set subscriptions for that information. The command MUST only be used with point-to-point communications channels and MUST only be sent as a request or as a notification. When sent as a request, a `voucherUpdateAck` command is generated in response to the `voucherUpdate` command.

The `propertyId` attribute of the class-level element is used to identify the property for which voucher information is being reported. The host MUST only include voucher information associated with that property in this command.

40.28.2 Attribute and Element Detail

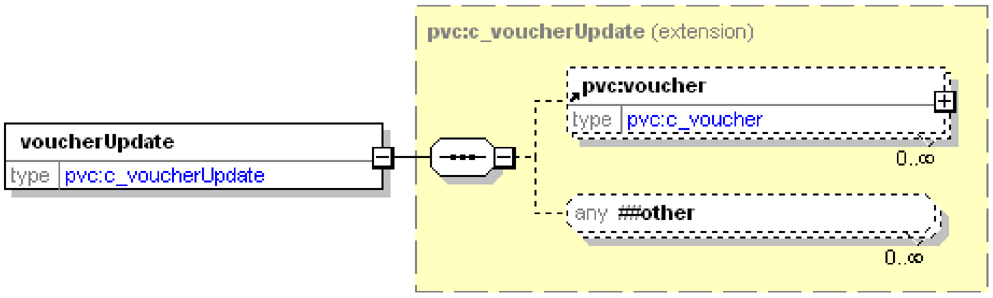


Table 40.63 voucherUpdate Attributes

Attribute	Restrictions	Description

`clientType`	`type: t_clientTypes`
	`use: required`
		Client type; the type of client to which the command is directed. Wildcards are not permitted.

`clientId`	`type: t_clientId`
	`use: required`
		Client identifier; the identifier of the client to which the command is directed. Wildcards are not permitted.

Table 40.64 voucherUpdate Sub-Elements

Element	Restrictions	Description

`voucher`	`minOcc: 0`
	`maxOcc: ∞`
		Contains voucher information. See Section 40.10, voucher Data Set.

NEW CLASS

40.29 voucherUpdateAck Command

40.29.1 Command Description

This command is used by a client to acknowledge the receipt of voucher updates from the host. The command MUST only be used with point-to-point communications channels and MUST only be sent as a response. The `voucherUpdateAck` command is generated in response to the `voucherUpdate` command.

The `propertyId` attribute of the class-level element is used to identify the property for which the updates were acknowledged. The host MUST include the property specified in the original request in its acknowledgement.

40.29.2 Attribute and Element Detail

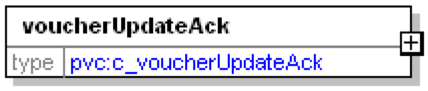


Table 40.65 voucherUpdateAck Attributes

Attribute	Restrictions	Description

`clientType`	`type: t_clientTypes`
	`use: required`
		Client type; the type of client from which the command was sent. Wildcards are not permitted.

`clientId`	`type: t_clientId`
	`use: required`
		Client identifier; the identifier of the client from which the command was sent. Wildcards are not permitted.

NEW CLASS

40.30 queryVouchers Command

40.30.1 Command Description

This command is used by a client to request a list of vouchers from the host. Vouchers can be selected by date/time, status, player, and/or end-client. The command MUST only be used with point-to-point communications channels and MUST only be sent as a request. A `voucherResults` command is generated in response to the `queryVouchers` command.

The `propertyId` attribute of the class-level element is used to identify the property for which vouchers are being requested. The host MUST only include vouchers associated with that property in its response.

Vouchers can be filtered based on various criteria including date/time, status, end-client, and/or the player. A voucher MUST qualify under all relevant criteria before it is included in the response.

- The `beginDateTime` and `endDateTime` MUST always be specified by the client and MUST always be used by the host; only vouchers with a `transDateTime` that is greater than or equal to the `beginDateTime` and that is less than or equal to the `endDateTime` MUST be included in the response.
- If the `voucherState` is not set to `S2S_all`, only vouchers with the specified status MUST be included in the response; otherwise, filtering on `voucherState` MUST not be performed by the host.
- If the `endClientType` is not set to `S2S_all`, only vouchers associated with the specified `endClientType` MUST be included in the response; otherwise, filtering on `endClientType` MUST NOT be performed by the host.
- If the `endClientId` is not set to `S2S_all`, only vouchers associated with the specified `endClientId` MUST be included in the response; otherwise, filtering on `endClientId` MUST NOT be performed by the host.
- If the `idNumber` is not empty, only vouchers with the specified `idReaderType` and `idNumber` MUST be included in the response; otherwise, filtering on `idReaderType` and `idNumber` MUST NOT be performed by the host.
- If the `playerId` is not empty, only vouchers for the specified player MUST be included in the response; otherwise, filtering on `playerId` MUST NOT be performed by the host.

In addition to other error codes that the host MAY report, the host MAY report the following error codes to the client, indicating that the request could not be processed.

- If the `voucherState` is not set to `S2S_all` and the host determines that the `voucherState` is invalid, the host MUST report the error using error code `S2S_PVX004 Invalid Voucher State`.
- If the `endClientType` is not set to `S2S_all` and the host determines that the `endClientType` is invalid, the host MUST report the error using error code `S2S_PVX002 Invalid End-Client Type`.
- If the `endClientId` is not set to `s2s_all` and the host determines that the end-client is not registered for the property, the host MUST report the error using error code `S2S_PVX003 Invalid End-Client for Property`.
- If the host determines that the `idReaderType` is invalid, the host MUST report the error using error code `S2S_GBX034 Invalid ID Reader Type`.

If the specified selection criteria result in an empty list of vouchers, the host MUST simply return an empty list to the client.

40.30.1.1 Managing Results Lists

The `maxQueryResults` attribute indicates the maximum number of vouchers that the host may include in the result list returned to the client. The host MUST NOT report more vouchers in the result list than the number specified in the `maxQueryResults` attribute. If the `maxQueryResults` attribute is set to `0` (zero), the host MAY return any number of vouchers in the result list.

The combination of the `lastEndClientType`, `lastEndClientId`, and `lastTransactionId` attributes MAY be used to identify the last voucher received in a previous query. This option MAY be used to request additional vouchers when the previous query did not include the complete result list—that is, the `remainingResults` attribute reported with the previous query was greater than `0` (zero). In such cases, the `lastEndClientType`, `lastEndClientId`, and `lastTransactionId` attributes reported in the previous query SHOULD be included as the `lastEndClientType`, `lastEndClientId`, and `lastTransactionId` attributes of the subsequent query.

When a `lastEndClientType` other than `S2S_all` is specified, all vouchers with an `endClientType` less than the `lastEndClientType` MUST be excluded from the result list; all vouchers with an `endClientType` equal to the `lastEndClientType` and an `endClientId` less than the `lastEndClientId` MUST be excluded from the result list; and, all vouchers with an `endClientType` equal to the `lastEndClientType` and an `endClientId` equal to the `lastEndClientId` and a `transactionId` less than or equal to the `lastTransactionId` MUST be excluded from the result list. When the `lastEndClientType` is set to `S2S_all`, all vouchers are eligible to be included in the result list.

The host MUST organize the vouchers contained in the result list in ascending order by `endClientType`, `endClientId`, and then `transactionId`. The `lastEndClientType`, `lastEndClientId`, and `lastTransactionId` attributes MUST be set to the `endClientType`, `endClientId`, and `transactionId` of the last voucher contained in the result list. The `currentResults` attribute MUST be set to the total number of vouchers in the result list.

When the number of vouchers selected by a query exceeds the number of vouchers included in the result list, the host MUST organize the vouchers into ascending order before constructing the result list, returning the vouchers with the lowest `endClientTypes`, `endClientIds`, and then `transactionIds` in the result list. In such cases, the `remainingResults` attribute MUST be set to a value greater than 0 (zero). The value MAY reflect the actual number of vouchers that satisfied the query but were not included in the result list or the value MAY simply indicate that not all vouchers were included in the result list. If all vouchers that satisfied the query were included in the result list, the `remainingResults` attribute MUST be set to `0` (zero).

When the number of vouchers selected by a query exceeds the number of vouchers included in the result list, the host MAY use the `lastQueryId` attribute to identify a static query that was used to satisfy the request. In such cases, when additional results from the same query are needed, the `lastQueryId` received in the response to a previous request SHOULD be used as the `lastQueryId` in subsequent requests. This indicates that the host SHOULD continue to report results from the same query. However, the host MAY generate a new query when necessary – for example, if a query times out. If the `lastQueryId` attribute is set to <empty> or the value of the `lastQueryId` attribute is invalid, the host MUST generate a new query.

Before all of the results of a static query have been requested by the client, the client MAY terminate the static query early by setting the `maxQueryResults` attribute to `-1` (negative one) in its request. This indicates that the client does not require any additional results from the static query and that the host can release any resources associated with the static query. When responding to such a request, the host MUST set the `currentResults` and `remainingResults` attributes to `0` (zero) and set the `lastQueryId` to <empty>.

NEW CLASS

40.30.2 Attribute and Element Detail

queryVouchers
type
pvc:c_queryVouchers

Table 40.66 queryVouchers Attributes (Sheet 1 of 2)

Attribute	Restrictions	Description

`clientType`	`type: t_clientTypes`
	`use: required`
		Client type; the type of client from which the command was sent. Wildcards are not permitted.

`clientId`	`type: t_clientId`
	`use: required`
		Client identifier; the identifier of the client from which the command was sent. Wildcards are not permitted.

`beginDateTime`	`type: t_dateTime`
	`use: required`
		Beginning date/time for the query.

`endDateTime`	`type: t_dateTime`
	`use: required`
		Ending date/time for the query.

`voucherStatus`	`type: t_voucherStates`
	`use: optional`
	`default: S2S_all`
		Voucher status; `S2S_all` wildcard permitted.

`endClientType`	`type: t_clientTypes`
	`use: optional`
	`default: S2S_all`
		End-client type; the type of end-client for which vouchers should be reported; `S2S_all` wildcard permitted.

`endClientId`	`type: t_clientId`
	`use: optional`
	`default: S2S_all`
		End-client identifier; the identifier of the end-client for which vouchers should be reported; `S2S_all` wildcard permitted.

`idReaderType`	`type: t_idReaderTypes`
	`use: optional`
	`default: S2S_none`
		Type of ID reader device.

`idNumber`	`type: t_idNumber`
	`use: optional`
	`default: <empty>`
		ID number.

`playerId`	`type: t_playerId`
	`use: optional`
	`default: <empty>`
		Player identifier.

`maxQueryResults`	`type: t_quantity`
	`use: optional`
	`default: 0`
		The maximum number of vouchers that the host should include in the result list; fewer results may be included.

`lastEndClientType`	`type: t_clientTypes`
	`use: optional`
	`default: S2S_all`
		The last `endClientType` received by the client in a previous query.

`lastEndClientId`	`type: t_clientId`
	`use: optional`
	`default: S2S_all`
		The last `endClientId` received by the client in a previous query.

NEW CLASS

Table 40.66 queryVouchers Attributes (Sheet 2 of 2)

Attribute	Restrictions	Description

`lastTransactionId`	`type: t_transactionId`
	`use: optional`
	`default: 0`
		The last `transactionId` received by the client in a previous query.

`lastQueryId`	`type: t_s2sId32`
	`use: optional`
	`default: <empty>`
		The identifier of the last static query performed by the host; can be set by the client to continue gathering results from the static query; if set to <empty>, the host MUST perform a new query.

NEW CLASS

40.31 voucherResults Command

40.31.1 Command Description

This command is used by the host to report a list of vouchers to a client. The command MUST only be used with point-to-point communications channels and MUST only be sent as a response. The `voucherResults` command is generated in response to a `queryVouchers` command.

The `propertyId` attribute of the class-level element is used to identify the property for which vouchers are being reported. The host MUST only include vouchers associated with that property in this command.

See Section 40.30, queryVouchers Command, for more details regarding the `currentResults`, `remainingResults`, `lastEndClientType`, `lastEndClientId`, `lastTransactionId`, and `lastQueryId` attributes.

40.31.2 Attribute and Element Detail

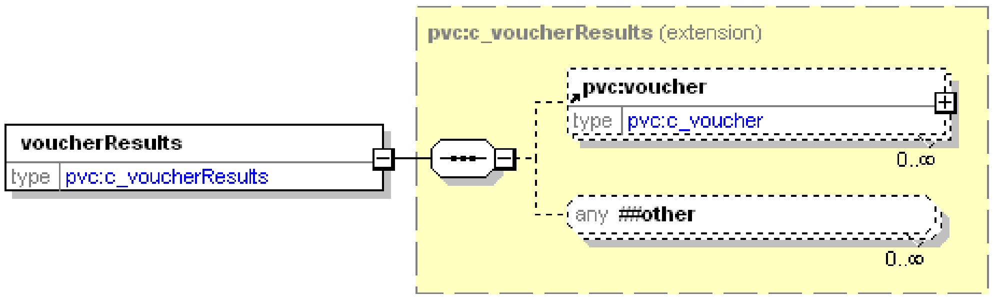


Table 40.67 voucherResults Attributes (Sheet 1 of 2)

Attribute	Restrictions	Description
`clientType`	type: t_clientTypes use: required	Client type; the type of client to which the command is directed. Wildcards are not permitted.
`clientId`	type: t_clientId use: required	Client identifier; the identifier of the client to which the command is directed. Wildcards are not permitted.
`currentResults`	type: t_quantity use: optional default: 0	The number of vouchers included in the response.
`remainingResults`	type: t_quantity use: optional default: 0	The number of vouchers remaining in the query that were not included in the response; any value greater than 0 (zero) indicates that all transactions were not included; the value may represent the actual number of remaining records or the value may simply indicate that additional records are available; when set to `0` (zero), no additional records are available.
`lastEndClientType`	type: t_clientTypes use: optional default: S2S_all	The `endClientType` of the last record contained in the response; if no records, set to `S2S_all`.

NEW CLASS

Table 40.67 voucherResults Attributes (Sheet 2 of 2)

Attribute	Restrictions	Description

`lastEndClientId`	`type: t_clientId`
	`use: optional`
	`default: S2S_all`
		The `endClientId` of the last record contained in the response; if no records, set to `S2S_all`.

`lastTransactionId`	`type: t_transactionId`
	`use: optional`
	`default: 0`
		The `transactionId` of the last record contained in the query; if no records, set to `0` (zero).

`lastQueryId`	`type: t_s2sId32`
	`use: optional`
	`default: <empty>`
		The identifier of the query performed by the host; can used to report the results of a static query over a series of responses; if set to <empty>, a static query was not used—the host will perform a new query for each new request.

Table 40.68 voucherResults Sub-Elements

Element	Restrictions	Description

`voucher`	`minOcc: 0`
	`maxOcc: ∞`
		Contains voucher information. See Section 40.10, voucher Data Set.

NEW CLASS

40.32 postVouchers Command

40.32.1 Command Description

This command is used by a client to request that the host post a list of vouchers to a specified location. The command MUST only be used with point-to-point communications channels and MUST only be sent as a request. A `postVouchersAck` command is generated in response to the `postVouchers` command.

The `propertyId` attribute of the class-level element is used to identify the property for which vouchers are being requested. The host MUST only include vouchers associated with that property in its response.

The `beginDateTime`, `endDateTime`, `voucherState`, `endClientType`, `endClientId`, `idReaderType`, `idNumber`, and `playerId` attributes of the `postVouchers` command MUST operate the same as similar attributes within the `queryVouchers` command. Likewise, error conditions related to those attributes MUST operate the same as similar error conditions specified for the `queryVouchers` command. See Section 40.30, queryVouchers Command, for more details.

The `transferType`, `transferLocation`, and `transferParams` attributes MUST operate the same as similar attributes within the `uploadPackage` command within the `download` class. See Chapter 36, download Class, for more details.

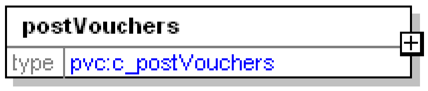
In addition to other error codes that the host MAY report, the host MAY report the following error codes to the client, indicating that the request could not be processed.

- If the transfer location is invalid, the host MUST report the error using error code `S2S_PVX020 Invalid Transfer Location`.
- If the transfer parameters are invalid, the host MUST report the error using error code `S2S_PVX021 Invalid Transfer Parameters`.
- If the transfer fails for any other reason, the host MUST report the error using error code `S2S_PVX022 File Transfer Failed`.

The list of vouchers that is posted to the specified location MUST be constructed as an XML file. The root element of the XML file MUST be a `postVouchers` element. The attributes of the `postVouchers` element MUST be set to the values as contained in the `postVouchers` request. Each voucher MUST be contained in a `voucher` sub-element of the `postVouchers` element. The syntax of the `postVouchers` and `voucher` elements MUST conform to the syntax defined for those elements within the S2S protocol; the only exception is that `voucher` elements are permitted as sub-elements of `postVouchers` elements.

If the specified selection criteria result in an empty list of vouchers, the host MUST simply post a `postVouchers` element containing an empty list of vouchers to the specified location.

40.32.2 Attribute and Element Detail



NEW CLASS

Table 40.69 postVouchers Attributes

Attribute	Restrictions	Description
`clientType`	type: `t_clientTypes` use: required	Client type; the type of client from which the command was sent. Wildcards are not permitted.
`clientId`	type: `t_clientId` use: required	Client identifier; the identifier of the client from which the command was sent. Wildcards are not permitted.
`beginDateTime`	type: `t_dateTime` use: required	Beginning date/time for the query.
`endDateTime`	type: `t_dateTime` use: required	Ending date/time for the query.
`voucherStatus`	type: `t_voucherStates` use: optional default: S2S_all	Voucher status; `S2S_all` wildcard permitted.
`endClientType`	type: `t_clientTypes` use: optional default: S2S_all	End-client type; the type of end-client for which vouchers should be reported; `S2S_all` wildcard permitted.
`endClientId`	type: `t_clientId` use: optional default: S2S_all	End-client identifier; the identifier of the end-client for which vouchers should be reported; `S2S_all` wildcard permitted.
`idReaderType`	type: `t_idReaderTypes` use: optional default: S2S_none	Type of ID reader device.
`idNumber`	type: `t_idNumber` use: optional default: <empty>	ID number.
`playerId`	type: `t_playerId` use: optional default: <empty>	Player identifier.
`transferType`	type: `t_transferTypes` use: required	Transfer type; `S2S_uploadPut`.
`transferLocation`	type: `t_transportLocation` use: required	URI to which the results should be transferred. See Section 36.1.10, Supported Transfer Protocols, for more details.
`transferParams`	type: `xs:string` use: optional maxLen: 128 default: <empty>	Optional parameters required for the transfer.

NEW CLASS

40.33 postVouchersAck Command

40.33.1 Command Description

This command is used by the host to report that a list of vouchers has been successfully posted. The command MUST only be used with point-to-point communications channels and MUST only be sent as a response. The `postVouchersAck` command is generated in response to a `postVouchers` command.

The `propertyId` attribute of the class-level element is used to identify the property for which vouchers are being reported. The host MUST only include vouchers associated with that property in this command.

40.33.2 Attribute and Element Detail

postVouchersAck
type
pvc:c_postVouchersAck

Table 40.70 postVouchersAck Attributes

Attribute	Restrictions	Description

`clientType`	`type: t_clientTypes`
	`use: required`
		Client type; the type of client to which the command is directed. Wildcards are not permitted.

`clientId`	`type: t_clientId`
	`use: required`
		Client identifier; the identifier of the client to which the command is directed. Wildcards are not permitted.

NEW CLASS

40.34 Data Types

The following tables describe the data types specific to the `playerVoucher` class. See Appendix A for other data types used within the `playerVoucher` class as well as other classes.

Table 40.71 playerVoucher Data Types

Data Type	Restrictions	Description

`t_validationId`	`type: xs:string`
	`minLen: 18`
	`maxLen: 18`
		Validation identifier.

`t_validationListId`	`type: xs:long`
	`minIncl: 0`
		Host-assigned validation list identifier.

`t_validationSeed`	`type: xs:string`
	`maxLen: 20`
	`pattern: [ -~]{0,20}`
		Validation seed.

`t_voucherActions`	`type: xs:string`
	`enumerations:`
	`S2S_issue`
	`S2S_redeem`
		Type of voucher action.

`t_voucherBoolean`	`type: xs:string`
	`enumerations:`
	`S2S_true`
	`S2S_false`
	`S2S_unknown`
		Voucher tri-state boolean value.

`t_voucherClientActions`	`type: t_extensibleList`
	`enumerations:`
	See Section 40.34.1.
		Voucher client actions.

`t_voucherClientExcs`	`type: t_exceptionCode` See Section 40.34.2.	Voucher end-client exception code.

`t_voucherHostActions`	`type: t_extensibleList`
	`enumerations:`
	See Section 40.34.3.
		Voucher host actions.

`t_voucherHostExcs`	`type: t_exceptionCode` See Section 40.34.4.	Voucher host exception code.

`t_voucherSources`	`type: xs:string`
	`enumertaions:`
	`S2S_endClient`
	`S2S_system`
		Issuance source for the voucher.

`t_voucherStates`	`type: t_extensibleList`
	`enumerations:`
	See Section 40.34.5.
		Voucher transaction states.

`t_voucherTitle16`	`type: xs:string`
	`maxLen: 16`
		16-charcater string.

`t_voucherTitle40`	`type: xs:string`
	`maxLen: 40`
		40-character string.

NEW CLASS

40.34.1 Enumeration Values for t_voucherClientActions

Table 40.72 Enumeration Values for t_voucherClientActions

Enumeration	Description
`S2S_issued`	Voucher issued.
`S2S_pending`	Voucher redemption requested.
`S2S_redeemed`	Voucher stacked.
`S2S_returned`	Voucher returned (not stacked).

40.34.2 Exception Codes for t_voucherClientExcs

Table 40.73 Exception Codes for t_voucherClientExcs

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

40.34.3 Enumeration Values for t_voucherHostActions

Table 40.74 Enumeration Values for t_voucherHostActions

Enumeration	Description
`S2S_endClientAction`	Stack or return to be determined by end-client.
`S2S_stack`	Force voucher to be stacked.
`S2S_return`	Force voucher to be returned (not stacked).

NEW CLASS

40.34.4 Exception Codes for t_voucherHostExceptions

Table 40.75 Exception Codes for t_voucherHostExceptions

Exception	Description
`0`	Redemption authorized.
`1`	Redemption in process at another end-client.
`2`	Voucher already redeemed.
`3`	Voucher expired.
`4`	Voucher not found.
`5`	Voucher cannot be redeemed at this end-client.
`6`	Incorrect player for voucher.
`99`	Redemption denied – reason unknown.

40.34.5 Enumeration Values for t_voucherStates

Table 40.76 Enumeration Values for t_voucherStates

Enumeration	Description
`S2S_issueSent`	Voucher issued, waiting for acknowledgement.
`S2S_issueAcked`	Voucher issued and acknowledged.
`S2S_redeemSent`	Redemption requested, waiting for authorization.
`S2S_redeemAuth`	Redemption authorized, transfer in process.
`S2S_commitSent`	Transaction completed/aborted, waiting for acknowledgement.
`S2S_commitAcked`	Transaction completed/aborted and acknowledged.

NEW CLASS

40.35 Error Codes

The following table includes the error codes contained within the `playerVoucher` class. The descriptions of individual commands and data sets indicate when a specific error code is used. See Appendix A for other error codes used within the `playerVoucher` class as well as other classes.

Table 40.77 playerVoucher Error Codes

Error Code	Suggested Error Text
S2S_PVX001	Manual Authentication Identifiers Not Supported by End-Client
S2S_PVX002	Invalid End-Client Type
S2S_PVX003	Invalid End-Client for Property
S2S_PVX004	Invalid Voucher State
S2S_PVX005	Invalid Voucher Action
S2S_PVX006	Invalid Voucher Source
S2S_PVX007	Invalid Host Action
S2S_PVX008	Invalid End-Client Action
S2S_PVX020	Invalid Transfer Location
S2S_PVX021	Invalid Transfer Parameters
S2S_PVX022	File Transfer Failed

NEW CLASS

40.36 Event Codes

The following table includes the event codes contained within the `playerVoucher` class. The descriptions of individual commands and data sets indicate when a specific event code is used.

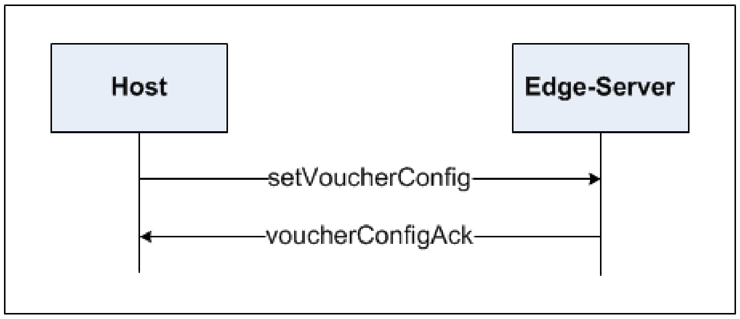
Table 40.78 playerVoucher Event Codes

Event Code	Suggested Event Text
S2S_PVE001	Invalid End-Client For Property
S2S_PVE002	Voucher Status Unavailable For End-Client
S2S_PVE003	Unable To Set Voucher State For End-Client
S2S_PVE004	Voucher Configuration Not Available For End-Client
S2S_PVE005	Unable To Set Voucher Configuration For End-Client
S2S_PVE010	Validation Data Error
S2S_PVE011	Validation Data Updated

40.37 Examples

The following examples demonstrate how commands within the `playerVoucher` class are constructed.

40.37.1 setVoucherConfig-voucherConfigAck



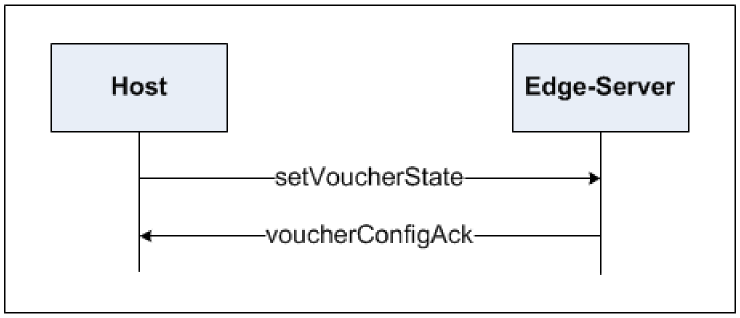
The following example illustrates the construction of a `setVoucherConfig` request from the host and a `voucherConfigAck` response from an edge-server.

```
<s2s:s2sMessage
    xmlns:s2s = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/"
    xmlns:pvc = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/pvc"
    >
    <s2s:s2sHeader
        s2s:toSystem = "https://voucherEdgeServer"
        s2s:fromSystem = "https://voucherHost"
        s2s:messageId = "11235813"
        s2s:dateTimeSent = "2013-12-31T14:11:25.000-05:00"
        />
    <s2s:s2sBody>
        <pvc:playerVoucher
            s2s:propertyId = "099"
            s2s:dateTime = "2013-12-31T14:11:25.000-05:00"
            s2s:commandId = "11235813"
            s2s:sessionType = "request"
            s2s:sessionId = "1"
            >
            <pvc:setVoucherConfig
                pvc:clientType = "S2S_system"
                pvc:clientId = "ABC_voucherEdgeServer"
                >
                <pvc:endClientList>
                    <pvc:endClient
                        pvc:endClientType = "S2S_egm"
                        pvc:endClientId = "ABC_123"
                        />
                    <pvc:endClient
                        pvc:endClientType = "S2S_egm"
                        pvc:endClientId = "ABC_456"
                        />
                </pvc:endClientList>
                <pvc:voucherConfig
                    pvc:configurationId = "12345"
                    pvc:restartStatus = "true"
                    pvc:requiredForPlay = "true"
                    pvc:timeToLive = "30000"
                    pvc:combineCashableOut = "true"
                    pvc:allowNonCashOut = "true"
                    pvc:maxValIds = "10"
                    pvc:minLevelIds = "5"
                    pvc:valIdListRefresh = "43200000"
                    pvc:valIdListLife = "86400000"
                    pvc:voucherHoldTime = "10000"
                    pvc:printOffLine = "true"
                    pvc:expireCashPromo = "30"
                    pvc:printExpCashPromo = "true"
                    pvc:expireNonCash = "1"
                    pvc:printExpNonCash = "true"
                    pvc:propName = "ABC Casino"
                    pvc:propLine1 = "1 Casino Way"
                    pvc:propLine2 = "Las Vegas"
                    pvc:titleCash = "Cash Out"
                    pvc:titlePromo = "Promo Voucher"
                    pvc:titleNonCash = "Restricted"
                    pvc:titleLargeWin = "Large Win"
                    pvc:titleBonusCash = "Cash Bonus"
                    pvc:titleBonusPromo = "Promo Bonus"
                    pvc:titleBonusNonCash = "Restricted"
                    pvc:titleWatCash = "Cash Transfer"
                    pvc:titleWatPromo = "Promo Transfer"
                    pvc:titleWatNonCash = "Restricted"
                    pvc:allowVoucherIssue = "true"
                    pvc:allowVoucherRedeem = "true"
                    pvc:maxOnLinePayOut = "1000000000"
                    pvc:maxOffLinePayOut = "1000000000"
                    pvc:printNonCashOffLine = "false"
                    />
            </pvc:setVoucherConfig>
        </pvc:playerVoucher>
    </s2s:s2sBody>
</s2s:s2sMessage>

<s2s:s2sMessage
    xmlns:s2s = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/"
    xmlns:pvc = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/pvc"
    >
    <s2s:s2sHeader
        s2s:toSystem = "https://voucherHost"
        s2s:fromSystem = "https://voucherEdgeServer"
        s2s:messageId = "23581321"
        s2s:dateTimeSent = "2013-12-31T14:11:26.000-05:00"
    />
    <s2s:s2sBody>
        <pvc:playerVoucher
            s2s:propertyId = "099"
            s2s:dateTime = "2013-12-31T14:11:26.000-05:00"
            s2s:commandId = "23581321"
            s2s:sessionType = "response"
            s2s:sessionId = "1"
            >
            <pvc:voucherConfigAck
                pvc:clientType = "S2S_system"
                pvc:clientId = "ABC_voucherEdgeServer"
                />
        </pvc:playerVoucher>
    </s2s:s2sBody>
</s2s:s2sMessage>
```

NEW CLASS

40.37.2 setVoucherState-voucherConfigAck



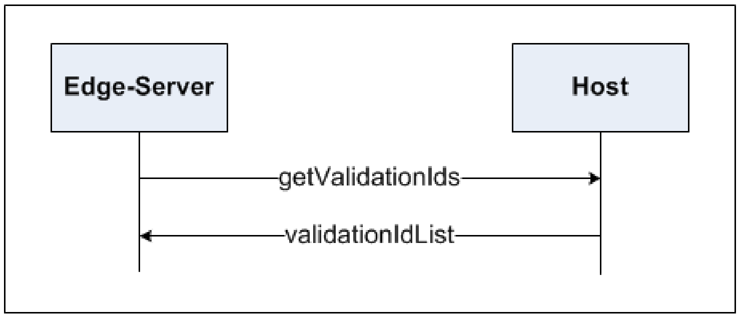
The following example illustrates the construction of a `setVoucherState` request from the host and a `voucherConfigAck` response from an edge-server.

```
<s2s:s2sMessage
    xmlns:s2s = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/"
    xmlns:pvc = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/pvc"
    >
    <s2s:s2sHeader
        s2s:toSystem = "https://voucherEdgeServer"
        s2s:fromSystem = "https://voucherHost"
        s2s:messageId = "11235814"
        s2s:dateTimeSent = "2013-12-31T14:41:55.000-05:00"
        />
    <s2s:s2sBody>
        <pvc:playerVoucher
            s2s:propertyId = "099"
            s2s:dateTime = "2013-12-31T14:41:55.000-05:00"
            s2s:commandId = "11235814"
            s2s:sessionType = "request"
            s2s:sessionId = "2"
            >
            <pvc:setVoucherState
                pvc:clientType = "S2S_system"
                pvc:clientId = "ABC_voucherEdgeServer"
                >
                <pvc:endClientList>
                    <pvc:endClient
                        pvc:endClientType = "S2S_egm"
                        pvc:endClientId = "ABC_123"
                        />
                    <pvc:endClient
                        pvc:endClientType = "S2S_egm"
                        pvc:endClientId = "ABC_456"
                        />
                </pvc:endClientList>
                <pvc:voucherState
                    pvc:enable = "true"
                    pvc:disableText = ""
                    pvc:lockOut = "false"
                    pvc:lockText = ""
                    pvc:lockTimeOut = "0"
                    pvc:configurationId = "12345"
                    />
            </pvc:setVoucherState>
        </pvc:playerVoucher>
    </s2s:s2sBody>
```

```
</s2s:s2sMessage>

<s2s:s2sMessage
    xmlns:s2s = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/"
    xmlns:pvc = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/pvc"
    >
    <s2s:s2sHeader
        s2s:toSystem = "https://voucherHost"
        s2s:fromSystem = "https://voucherEdgeServer"
        s2s:messageId = "23581322"
        s2s:dateTimeSent = "2013-12-31T14:41:56.000-05:00"
        />
    <s2s:s2sBody>
        <pvc:playerVoucher
            s2s:propertyId = "099"
            s2s:dateTime = "2013-12-31T14:41:56.000-05:00"
            s2s:commandId = "23581322"
            s2s:sessionType = "response"
            s2s:sessionId = "2"
            >
            <pvc:voucherConfigAck
                pvc:clientType = "S2S_system"
                pvc:clientId = "ABC_voucherEdgeServer"
                />
        </pvc:playerVoucher>
        </s2s:s2sBody>
</s2s:s2sMessage>
```

40.37.3 getValidationIds-validationIdList



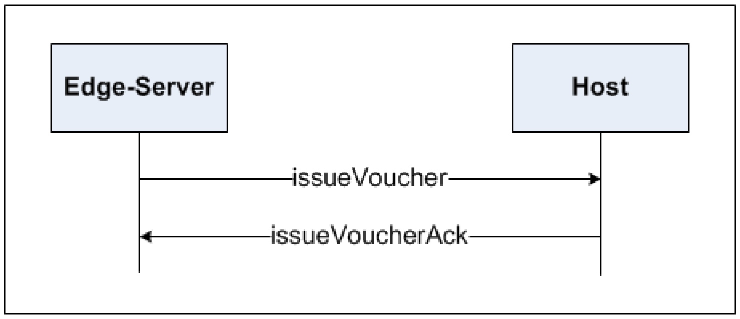
The following example illustrates the construction of a `getValidationIds` request from an edge-server and a `validationIdList` response from the host.

```
<s2s:s2sMessage
    xmlns:s2s = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/"
    xmlns:pvc = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/pvc"
    >
    <s2s:s2sHeader
        s2s:toSystem = "https://voucherHost"
        s2s:fromSystem = "https://voucherEdgeServer"
        s2s:messageId = "23581323"
        s2s:dateTimeSent = "2013-12-31T14:46:00.000-05:00"
        />
    <s2s:s2sBody>
        <pvc:playerVoucher
```

```
            s2s:propertyId = "099"
            s2s:dateTime = "2013-12-31T14:46:00.000-05:00"
            s2s:commandId = "23581323"
            s2s:sessionType = "request"
            s2s:sessionId = "3"
            >
            <pvc:getValidationIds
                pvc:clientType = "S2S_system"
                pvc:clientId = "ABC_voucherEdgeServer"
                pvc:endClientType = "S2S_egm"
                pvc:endClientId = "ABC_123"
                pvc:validationListId = "0"
                pvc:numValidationIds = "10"
                pvc:valIdListExpired = "true"
                pvc:configurationId = "12345"
                />
        </pvc:playerVoucher>
    </s2s:s2sBody>
</s2s:s2sMessage>

<s2s:s2sMessage
    xmlns:s2s = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/"
    xmlns:pvc = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/pvc"
    >
    <s2s:s2sHeader
    s2s:toSystem = "https://voucherEdgeServer"
    s2s:fromSystem = "https://voucherHost"
    s2s:messageId = "11235815"
    s2s:dateTimeSent = "2013-12-31T14:46:01.000-05:00"
    />
    <s2s:s2sBody>
        <pvc:playerVoucher
        s2s:propertyId = "099"
        s2s:dateTime = "2013-12-31T14:46:01.000-05:00"
        s2s:commandId = "11235815"
        s2s:sessionType = "response"
        s2s:sessionId = "3"
        >
            <pvc:validationIdList
                pvc:clientType = "S2S_system"
                pvc:clientId = "ABC_voucherEdgeServer"
                pvc:endClientType = "S2S_egm"
                pvc:endClientId = "ABC_123"
                pvc:validationListId = "8642"
                pvc:deleteCurrent = "true"
                >
                <pvc:validationId
                    pvc:validationId = "09912345678912345"
                    pvc:validationSeed = "1A2B3C4D5C6D7E8F9"
                    />
                <pvc:validationId
                    pvc:validationId = "09912345678912346"
                    pvc:validationSeed = "1A2B3C4D5C6D7E8F9"
                    />
                <pvc:validationId
                    pvc:validationId = "09912345678912347"
                    pvc:validationSeed = "1A2B3C4D5C6D7E8F9"
                    />
                <pvc:validationId
                    pvc:validationId = "09912345678912348"
                    pvc:validationSeed = "1A2B3C4D5C6D7E8F9"
                    />
                <pvc:validationId
                    pvc:validationId = "09912345678912349"
                    pvc:validationSeed = "1A2B3C4D5C6D7E8F9"
                    />
                <pvc:validationId
                    pvc:validationId = "09912345678912350"
                    pvc:validationSeed = "1A2B3C4D5C6D7E8F9"
                    />
                <pvc:validationId
                    pvc:validationId = "09912345678912351"
                    pvc:validationSeed = "1A2B3C4D5C6D7E8F9"
                    />
                <pvc:validationId
                    pvc:validationId = "09912345678912352"
                    pvc:validationSeed = "1A2B3C4D5C6D7E8F9"
                    />
                <pvc:validationId
                    pvc:validationId = "09912345678912353"
                    pvc:validationSeed = "1A2B3C4D5C6D7E8F9"
                    />
                <pvc:validationId
                    pvc:validationId = "09912345678912354"
                    pvc:validationSeed = "1A2B3C4D5C6D7E8F9"
                    />
                <pvc:validationId
                    pvc:validationId = "09912345678912355"
                    pvc:validationSeed = "1A2B3C4D5C6D7E8F9"
                    />
            </pvc:validationIdList>
        </pvc:playerVoucher>
    </s2s:s2sBody>
</s2s:s2sMessage>
```

40.37.4 issueVoucher-issueVoucherAck

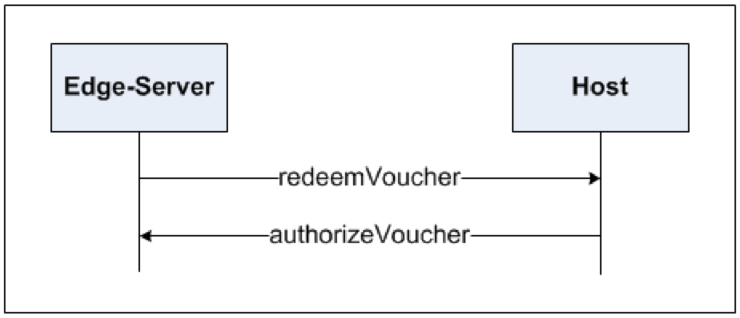


The following example illustrates the construction of an `issueVoucher` request from an edge-server and an `issueVoucherAck` response from the host.

```
<s2s:s2sMessage
    xmlns:s2s = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/"
    xmlns:pvc = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/pvc"
    >
    <s2s:s2sHeader
        s2s:toSystem = "https://voucherHost"
        s2s:fromSystem = "https://voucherEdgeServer"
        s2s:messageId = "23581324"
        s2s:dateTimeSent = "2013-12-31T15:01:16.000-05:00"
        />
    <s2s:s2sBody>
        <pvc:playerVoucher
            s2s:propertyId = "099"
            s2s:dateTime = "2013-12-31T15:01:16.000-05:00"
            s2s:commandId = "23581324"
            s2s:sessionType = "request"
            s2s:sessionId = "4"
            >
            <pvc:issueVoucher
                pvc:clientType = "S2S_system"
                pvc:clientId = "ABC_voucherEdgeServer"
                pvc:endClientType = "S2S_egm"
                pvc:endClientId = "ABC_123"
                pvc:transactionId = "97531"
                pvc:idReaderType = "S2S_magCard"
                pvc:idNumber = "09900101977"
                pvc:playerId = "00101977"
                pvc:validationId = "099123456789l2345"
                pvc:voucherAmt = "12500000"
                pvc:creditType = "S2S_cashable"
                pvc:voucherSource = "S2S_endClient"
                pvc:largeWin = "false"
                pvc:voucherSequence = "123"
                pvc:expireCredits = "false"
                pvc:expireDateTime = "2000-01-01T00:00:00.000-00:00"
                pvc:transferAmt = "12500000"
                pvc:transferDateTime = "2013-12-31T15:01:16.000-05:00"
                pvc:expireDays = "30"
                pvc:endClientAction = "S2S_issued"
                pvc:endClientException = "0"
                />
        </pvc:playerVoucher>
    </s2s:s2sBody>
</s2s:s2sMessage>

<s2s:s2sMessage
    xmlns:s2s = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/"
    xmlns:pvc = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/pvc"
    >
    <s2s:s2sHeader
        s2s:toSystem = "https://voucherEdgeServer"
        s2s:fromSystem = "https://voucherHost"
        s2s:messageId = "11235816"
        s2s:dateTimeSent = "2013-12-31T15:01:17.000-05:00"
        />
    <s2s:s2sBody>
        <pvc:playerVoucher
            s2s:propertyId = "099"
            s2s:dateTime = "2013-12-31T15:01:17.000-05:00"
            s2s:commandId = "11235816"
            s2s:sessionType = "response"
            s2s:sessionId = "4"
            >
            <pvc:issueVoucherAck
                pvc:clientType = "S2S_system"
                pvc:clientId = "ABC_voucherEdgeServer"
                pvc:endClientType = "S2S_egm"
                pvc:endClientId = "ABC_123"
                pvc:transactionId = "97531"
                />
        </pvc:playerVoucher>
    </s2s:s2sBody>
</s2s:s2sMessage>
```

40.37.5 redeemVoucher-authorizeVoucher

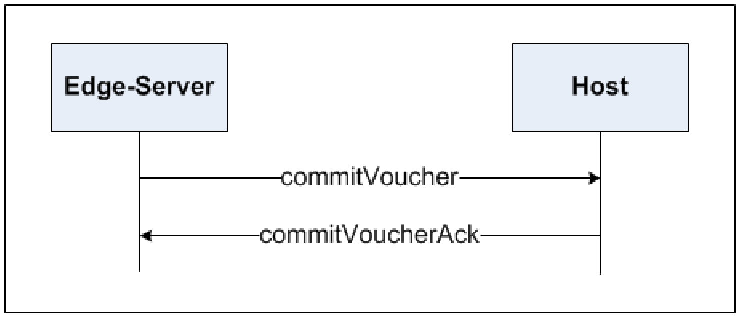


The following example illustrates the construction of a `redeemVoucher` request from an edge-server and an `authorizeVoucher` response from the host.

```
<s2s:s2sMessage
    xmlns:s2s = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/"
    xmlns:pvc = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/pvc"
    >
    <s2s:s2sHeader
        s2s:toSystem = "https://voucherHost"
        s2s:fromSystem = "https://voucherEdgeServer"
        s2s:messageId = "23581325"
        s2s:dateTimeSent = "2013-12-31T15:10:25.000-05:00"
        />
    <s2s:s2sBody>
        <pvc:playerVoucher
            s2s:propertyId = "099"
            s2s:dateTime = "2013-12-31T15:10:25.000-05:00"
            s2s:commandId = "23581325"
            s2s:sessionType = "request"
            s2s:sessionId = "5"
            >
            <pvc:redeemVoucher
                pvc:clientType = "S2S_system"
                pvc:clientId = "ABC_voucherEdgeServer"
                pvc:endClientType = "S2S_egm"
                pvc:endClientId = "ABC_123"
                pvc:transactionId = "97532"
                pvc:idReaderType = "S2S_magCard"
                pvc:idNumber = "09900101977"
                pvc:playerId = "00101977"
                pvc:validationId = "099123456789l2345"
                />
        </pvc:playerVoucher>
    </s2s:s2sBody>
</s2s:s2sMessage>

<s2s:s2sMessage
    xmlns:s2s = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/"
    xmlns:pvc = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/pvc"
    >
    <s2s:s2sHeader
        s2s:toSystem = "https://voucherEdgeServer"
        s2s:fromSystem = "https://voucherHost"
        s2s:messageId = "11235817"
        s2s:dateTimeSent = "2013-12-31T15:10:26.000-05:00"
        />
    <s2s:s2sBody>
        <pvc:playerVoucher
            s2s:propertyId = "099"
            s2s:dateTime = "2013-12-31T15:10:26.000-05:00"
            s2s:commandId = "11235817"
            s2s:sessionType = "response"
            s2s:sessionId = "5"
            >
            <pvc:authorizeVoucher
                pvc:clientType = "S2S_system"
                pvc:clientId = "ABC_voucherEdgeServer"
                pvc:endClientType = "S2S_egm"
                pvc:endClientId = "ABC_123"
                pvc:transactionId = "97532"
                pvc:validationId = "099123456789123456"
                pvc:voucherAmt = "12500000"
                pvc:creditType = "S2S_cashable"
                pvc:voucherSource = "S2S_endClient"
                pvc:largeWin = "false"
                pvc:voucherSequence = "123"
                pvc:expireCredits = "false"
                pvc:expireDateTime = "2000-01-01T00:00:00.000-00:00"
                pvc:hostAction = "S2S_endClientAction"
                pvc:hostException = "0"
                />
        </pvc:playerVoucher>
    </s2s:s2sBody>
</s2s:s2sMessage>
```

40.37.6 commitVoucher-commitVoucherAck



The following example illustrates the construction of a `commitVoucher` request from an edge-server and a `commitVoucherAck` response from the host.

```
<s2s:s2sMessage
    xmlns:s2s = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/"
    xmlns:pvc = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/pvc"
    >
    <s2s:s2sHeader
        s2s:toSystem = "https://voucherHost"
        s2s:fromSystem = "https://voucherEdgeServer"
        s2s:messageId = "23581326"
        s2s:dateTimeSent = "2013-12-31T15:15:30.000-05:00"
        />
    <s2s:s2sBody>
        <pvc:playerVoucher
            s2s:propertyId = "099"
            s2s:dateTime = "2013-12-31T15:15:30.000-05:00"
            s2s:commandId = "23581326"
            s2s:sessionType = "request"
            s2s:sessionId = "6"
            >
            <pvc:commitVoucher
                pvc:clientType = "S2S_system"
                pvc:clientId = "ABC_voucherEdgeServer"
                pvc:endClientType = "S2S_egm"
                pvc:endClientId = "ABC_123"
                pvc:transactionId = "97532"
                pvc:validationId = "09912345678912345"
                pvc:voucherAmt = "12500000"
                pvc:creditType = "S2S_cashable"
                pvc:voucherSource = "S2S_endClient"
                pvc:largeWin = "false"
                pvc:voucherSequence = "123"
                pvc:expireCredits = "false"
                pvc:expireDateTime = "2000-01-01T00:00:00.000-00:00"
                pvc:transferAmt = "12500000"
                pvc:transferDateTime = "2013-12-31T15:15:30.000-05:00"
                pvc:endClientAction = "S2S_redeemed"
                pvc:endClientException = "0"
                />
        </pvc:playerVoucher>
    </s2s:s2sBody>
</s2s:s2sMessage>

<s2s:s2sMessage
    xmlns:s2s = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/"
    xmlns:pvc = "http://www.gamingstandards.com/s2s/schemas/v1.2.6/pvc"
    >
    <s2s:s2sHeader
        s2s:toSystem = "https://voucherEdgeServer"
        s2s:fromSystem = "https://voucherHost"
        s2s:messageId = "11235818"
        s2s:dateTimeSent = "2013-12-31T15:15:31.000-05:00"
        />
    <s2s:s2sBody>
        <pvc:playerVoucher
            s2s:propertyId = "099"
            s2s:dateTime = "2013-12-31T15:15:31.000-05:00"
            s2s:commandId = "11235818"
            s2s:sessionType = "response"
            s2s:sessionId = "6"
            >
            <pvc:commitVoucherAck
                pvc:clientType = "S2S_system"
                pvc:clientId = "ABC_voucherEdgeServer"
                pvc:endClientType = "S2S_egm"
                pvc:endClientId = "ABC_123"
                pvc:transactionId = "97532"
                />
        </pvc:playerVoucher>
    </s2s:s2sBody>
```

```
</s2s:s2sMessage>
```