IMPLEMENTATION GUIDE
VISTA PAYMENT SERVER

5.3.0.0

Vista Entertainment Solutions
2024-07-10

for the of cinema

Copyright notice

Vista Entertainment Solutions Limited

PO Box 90551, Victoria Street West, Auckland 1124, New Zealand

P: +64 9 984 4570 **F**: + 64 9 379 0685 www.vista.co

Document version history

Version	Date changed	Author	Notes
1.0	2023-07-17	Stephen Moir, Sam Butchart, Chris Lovie-Tyler	First version
1.1	2024-04-04	Sam Butchart	Clarified logging behaviour when client logging unavailable.
1.2	2024-07-10	Stephen Moir, Sam Butchart	Updated locations of related documents and changelogs. Added Vista Folder Permissions reference. Further clarified logging behaviour when client logging unavailable.

Related documentation

The following Vista documents contain additional information related to some of the concepts and tasks described in this document.

Document	Description	Location
Payment Server Installation Guide	This document details how to install Payment Server.	Vista Services site under **Documents**.
Vista Folder Permissions	The document details the folder permissions required for Vista Cinema products.	Vista Services site under **Documents > Technical / Support Guides**
Data Privacy Feature Guide	This guide describes the areas in Vista software that allow you to anonymise or delete stored personal data so that you can comply with data privacy laws.	Vista Help Centre under **Compliance**

Contents

About Payment Server

Payment Server is the middleware that provides integrated-payments functionality to client applications like Point of Sale (POS). While it is called Payment Server, it does not run on a server; it runs as part of these client applications.

Versioning of Payment Server

Payment Server is versioned independently of other Vista products. The version number is made up of four parts, each separated by a full stop: Major.Minor.Release.Patch. In this schema, changes that impact security require that either the Major or Minor version number is incremented. Increments to the Release and Patch version numbers (wildcard versions) indicate changes without any implications for security.

Major version number

The major version changes when Vista makes changes to Payment Server that would make it incompatible with some currently supported versions of Vista Cinema. Our ability to use one version of Payment Server with many versions of Vista Cinema is something that we value highly, so changes like this would not be undertaken lightly.

Minor version number

The minor version changes when we make a change to the security features of Payment Server. Changes of this type are infrequent, but when they are made, we recommend that you upgrade and use them.

The two most recent examples of this have been 5.2, where we made changes to how card details are released to client applications, and 5.3, where new logging locations were added.

Release version number

The release version changes when a change is made to Payment Server that doesn't impact the security features of the application. This could be a bug fix to existing functionality or a new feature that doesn't impact the security posture of the application.

Patch version number

The patch version changes when a component that ships with Payment Server but is not part of the core Payment Server application changes. This is most commonly because we have added or updated a payment connector.

Document applicability

This document applies to versions of Payment Server with major version number 5 and minor version number 3. If these version numbers match, this document will apply regardless of the release and patch version numbers.

Assumptions

This document assumes that Payment Server is running in an environment with standard security protections, such as anti-malware software, a firewall, and network-level protection systems, in place.

This document also assumes that Payment Server is run by a user with the minimum permission set required to successfully run the software.

Breaking either of these assumptions may compromise the security of Payment Server and impact the PCI validation of your circuit.

Scope of Vista PCI support

Vista is able to provide advice and support to install Payment Server in accordance with these guidelines. However, Vista is not a PCI Qualified Integrator and Reseller (QIR), so is not able to offer advice on aspects of PCI DSS beyond this.

Remote access for support

To offer support for Payment Server, we might need remote access to your environment. For us to safely do this, we require that any remote-access tool uses multi-factor authentication (MFA). If you can't provide this, speak to us and we will have a solution.

Vista-application payment processing

There are several Vista applications that are capable of processing payments, but not all of these use Payment Server. Only the following do:

- POS
- Lumos Kiosk, but not Kiosk (Legacy)

We also have applications that facilitate payments, but they have been designed in such a way that they do not use Payment Server and they never have access to card details. The PCI Secure Software Standard (SSS) does not currently cover applications like these, so it is not possible for these applications to be PCI validated. These applications are:

- Web Payments Module
- Rove
- Serve
- Vista Mobile

It is possible to configure Vista Connect to not use Web Payments Module and to instead capture payment card details directly and send these to Cinema for processing with Vista Sales Server. This configuration is referred to as Cinema payments. Neither Connect nor Vista Sales Server use Payment Server or have any PCI validation.

Kiosk (Legacy) (previously known just as Kiosk) is also able to be configured to accept payments via magnetic stripe reader (MSR), but it does not use Payment Server or have any PCI validation.

Supported platforms

To be PCI compliant, Payment Server must be run on supported systems. Payment Server also requires Vista Cinema to run.

Vista supports our software running on platforms that still have extended support from Microsoft. Please refer to the Microsoft documentation for when extended support ends for these products. (Search at https://learn.microsoft.com/en-us/lifecycle/products/.)

Payment Server 5.3 has been validated on the following platforms:

Microsoft Windows Client

- Windows 11
- Windows 10 Enterprise
- Windows 10 IoT

Windows Server

- Windows Server 2022
- Windows Server 2019

- Windows Server 2016
- Windows Server 2012 R2

SQL Server

- Azure SQL Database
- SQL Server 2019
- SQL Server 2017
- SQL Server 2016
- SQL Server 2014

Keeping up to date with Payment Server changes

All of the details of Payment Server changes are available in the release notes that are available on the Vista Services site, under Documents > Release Notes (Payment Server) or to logged in users of the Vista Help Centre under Changelogs > Payment Server changelog.

As with all Vista products, if there is a change to Payment Server that has a security impact, we will send out details via the Security Bulletin email. If you would like to receive these emails, please contact your Vista representative.

Keeping Payment Server up to date

Payment Server can be installed independently of Vista Cinema. The installer is available on the Services site. The latest version of Payment Server is supported on all currently supported versions of Vista Cinema. However, we have not intentionally broken backwards compatibility, so Payment Server 5.3 should theoretically work with any version of Vista Cinema from version 5.0.1.18 on.

Payment Server is also shipped with Vista Cinema. When a new version of Payment Server is released, we also update all currently supported versions of Vista Cinema to include the latest version. If you are keeping Vista Cinema up to date, Payment Server will be up to date too.

If your organisation has moved to Vista Cloud, all software versions will be updated regularly, including Payment Server.

See the *Payment Server Installation Guide* for full details of how to install Payment Server.

Determining the installed version of Payment Server

To check the version of Payment Server installed in your on-premises environment, find the row of the tblProduct table in the VISTA database with the Product_strCode value “VISTAPAYMENTSERVER”. The Product_strVersion column will contain the currently installed version of Payment Server.

EMV devices (PIN pads) are recommended

Vista strongly recommends the use of EMV devices (also known as PIN pads) for payments to protect card-holder details. These devices are much more restricted than the general-purpose computers that Vista software runs on and so are much less prone to being compromised. When EMV devices are used, Payment Server only has access to card-holder data that is released by the EMV device. This data is usually restricted to:

- Truncated card number (no more than the first 6 and last 4 digits)
- Card-holder name
- Card expiry data

In Vista Cloud, EMV is the only option for payment-card processing.

How Payment Server protects MSR card data

When an MSR is used for payment, Payment Server needs to have access to the sensitive card details in cleartext to facilitate the payment. Once we have had a response from the payment gateway, these cleartext card details are removed from volatile RAM. When not required for transmitting to a payment provider, we make use of the SecureString class provided by the .NET framework to keep the card details encrypted in RAM.

We have additional checks to ensure that card details are never written to logs. Vista applications use a shared logging component. Within this component, we check all output for numbers that look like they might be card numbers, and we truncate them to no more than the first 6 and last 4 digits (hereafter referred to as “6.4”). As the logging component is shared, this applies to all applications, not just Payment Server. Within Payment Server, we have additional processing to ensure that track data and CVV information being handled by Payment Server is also truncated.

Payment Server only makes the 6.4 of a card number available to other applications. This ensures that when we write any transaction details elsewhere, for example to the database, the most detail about a card we will store is the 6.4.

Because we use 6.4, we do not use hashed card details.

We do not need to store full card details beyond the lifetime of a transaction, so we do not use encryption for long-term storage. As we don’t use encryption, we don’t need to worry about key management.

Whenever a primary account number (PAN) is displayed, it’s only ever 6.4 (because that’s all Vista software has stored).

When transmitting card data, we enforce the usage of TLS 1.2 and encourage the use of industry-recommended cipher suites.

See Appendix A for a table of controls on all sensitive data.

Power-failure handling

To handle a scenario where there is a power failure at the POS workstation after transaction details have been sent to the payment gateway, but before a response has been received and processed, we keep the obfuscated middle digits of a card number on POS until the Vista transaction has been completed (via the payment screen button, with the Vista logo on it).

Capturing card details before payment (checks)

If you need to capture card details well before you will process payments (common in food service), we support this. We don’t ever capture full card details, though. We use a variety of techniques, depending on which payment gateway is being used, but this always results in us storing a card token/card proxy which can be used for later payment. This data is encrypted before being written to the database.

Third-party dependencies for protection

When protecting MSR card data, we rely on protection mechanisms not developed by Vista. These include:

- The SecureString class, which is used to protect data when it is held in system RAM. This is provided by Microsoft’s .NET framework.
- The Data Protection API (DPAPI), which is used to encrypt files that Payment Server manages for internal functionality. This is provided by Microsoft’s Windows operating system.
- TLS 1.2 and configured cipher suites, which are used to protect information sent to third-party payment gateways. These are provided by Microsoft’s Windows operating system, and the available cipher suites will depend on your version and configuration of Windows.
    - TLS 1.2 or above must be used with approved cryptographic algorithms. Approved cryptographic algorithms and methods are those recognised by industry-accepted standards bodies (for example: NIST, ANSI, ISO, and EMVCo). Weak ciphers, SSL/early TLS, or cryptographic algorithms or parameters that are known to be vulnerable must not be used.

  - We also recommend keeping up to date with Windows updates, as this will ensure that your Root Certificate Authority store is up to date. For the transfer of sensitive data, the certificates provided by Windows are sufficient.

- We recommend a file integrity management (FIM) and security information and event management (SIEM) system to ensure the integrity of critical files and records of security related events that take place in Payment Server. Vista does not recommend any specific vendor for these systems but does offer advice on their configuration in this document. Specifically, in the “File protection” and “Integrating Payment Server with an SIEM service” sections.

Enabling protection of MSR card data

If your installation of Windows is up to date, no further action is required to enable the data protections detailed in this document. They will all be available immediately upon installation and be applied to all sensitive data provided to Payment Server.

Outdated versions of Windows may not support sufficient TLS cipher suites to communicate with payment gateways and may result in this communication failing.

Database configuration

Vista applications are unable to use Windows authentication with SQL Server, so when installing SQL Server you will need to ensure that SQL authentication is still enabled. Users with Windows authentication can be used when doing admin tasks such as installing the software and doing updates.

The SQL Server user used by Vista applications must not be the sa (systems administrator) user.

PCI DSS 4.0 requires passwords that have a minimum length of 12 characters and contain both numeric and alphabetic characters.

We recommend configuring the server to only support TLS1.2 or later.

If the version of SQL Server supports transparent data encryption (TDE), we recommend enabling this to protect data at rest.

Payment Server access

Payment Server is accessible to applications running on the workstation that it is installed on. As such, we rely on Windows Authentication to ensure that only permitted users have access to the workstation. We recommend that workstations with Payment Server installed are secured with passwords that comply with PCI-DSS requirements for passwords and password policies. You can find these in PCI-DSS 4.0 requirements 8.3.X.

Pre-configured Vista Cinema users

Two pre-configured users (sysadmin and vistasupport) are created during Vista Cinema installation to allow initial configuration of the software and later support by Vista Services representatives. These users both have SysAdmin permissions. Only the sysadmin account is configured to allow immediate access to the software following installation, and the default password must be reset upon first login to Back Office. Vista recommends setting a strong password for this user and disabling both preconfigured users after the initial configuration of the software, only enabling them at the request of Vista Services representatives. If left enabled, these pre-configured users can be used to access POS and set a PIN for the user, allowing elevated access to POS functionality.

File protection

Ensure that users of Payment Server have the minimum file permissions necessary to be able to run the application. Payment Server requires the same permissions as the client application along with some Payment Server–specific permissions.

Write permissions are required for Payment Server to write logs and cache data in the event of losing access to the database server or losing power before a transaction is authorised.

We recommend that a file integrity management (FIM) system be used to monitor all directories that contain application files used by Payment Server and the client application. Ensure FIM monitoring is not required for the locations where logs and temporary files are written to.

The following tables show the minimum permissions required on a workstation running Payment Server and whether we recommend that FIM be used to monitor that location. Some permissions are included for client applications where this directly impacts Payment Server. A high level of permissions is required when installing or updating the client application. The Vista Folder Permissions technical guide contains more details on the permissions required for Vista Cinema applications.

On-premises

Vista applications are usually installed in C:\Vista, but this location can be set on first installation. All paths are relative to this installation directory.

Path	Permissions	FIM Recommended
\PaymentsModule\	Read	Yes
\PaymentsModule\VistaPaymentRecovery.txt	Read, Write	No
\PaymentsModule\CachedDatasets\	Read, Write	No
\PaymentsModule\Log\	Read, Write	No
\[Client Application]\ (for example, \VistaPOS\)	Read and execute	Yes
\Log\ (default logging directory for client applications)	Write	No

Vista Cloud

This Vista installation directory is C:\Program Files (x86)\Vista

Path	Permissions	FIM Recommended
C:\Program Files (x86)\Vista\[Client Application]\ (for example, \VistaPOS\)	Read and execute	Yes
C:\Program Files (x86)\Vista\PaymentsModule\	Read	Yes
C:\ProgramData\Vista\	Read, Write	No

Required open server ports

The following ports are those that are required for the server that Payment Server is installed on. Client applications may require additional ports to be open.

UDP ports:

- 1434 – SQL Server Browser service

TCP/IP Ports:

- 1433 – SQL Server
- 445 – Windows File sharing
- 135 – Windows File sharing

Ensure that all ports that are *not* required for the operation of Payment Server or other client applications are disabled.

Use of wireless technology

Payment Server has no functionality that explicitly requires or prohibits the usage of wireless technology. If Payment Server is accessing resources via a wireless technology, please follow all PCI recommendations.

Payment connectors

Payment Server uses an add-in architecture to enable integration with different payment providers. Payment connectors are available in the _Install_Customisation folder of the Payment Server installation package. Each payment connector includes documentation on how to configure Vista software to use the connector.

Payment-connector configuration

The primary place where payment connectors are configured is in the database table tblPaymentModuleConfig.

Which payment-connector configuration is used by Payment Server is determined by finding the most specific row in tblPaymentModuleConfig based on client ID (Pay_strClientId), client application (Pay_strApplicationId) and the type of card that is used for payment (Pay_strCardType). If any of these fields contain a hyphen (-), this is treated as a wildcard and will match to any value.

Connectors will be chosen in the following order. If there is more than one match for any of these items, the last defined item will be used.

1. Matching card type, application, and client ID
2. Matching card type and application or client ID
3. Matching card type
4. Matching application and client ID
5. Matching application or client ID
6. All wildcards

The client ID and the application name are always provided by the client application to Payment Server.

The card type is determined in a few ways:

- If the client application specifies a payment gateway ID in the PAYGATEWAY_ID parameter, this is used to match on the card type before any of the following options. If this parameter is provided, it will not match on a wildcard card type.
- If EMV devices (PIN pads) are being used for payment, the client application needs to specify the card type when calling Payment Server via the CARD_TYPE parameter.

- If a mag stripe reader has been used, Payment Server first attempts to determine the type of card based on the card number. The most specific entry in tblCardDefinition that matches the card number and is marked as a payment card (Card_strPaymentCard = 'Y') is used for the card type. If there is no match, the CARD_TYPE parameter is used.

The payment gateway ID and card type are both configurable for POS as part of the Payment Type configuration.

COM vs .NET binding

Historically, payment connectors for Payment Server were created as COM components and required registration to work. COM is still required for some legacy connectors, but .NET binding is our preferred way of doing things now. For Payment Server to load a payment connector with .NET binding, the connector DLL and all of its required files must be in the Vista Installation Directory\PaymentsModule folder.

Card-release policies

All MSR card swipes for applications that use Payment Server are processed by Payment Server. Client applications can request that Payment Server release full card details for cards that aren't payment cards. This is often used for Loyalty member cards. Payment Server will use a card-release policy to examine the card details and determine whether the card is a payment card. It looks at both the card number and the format of the data in the card track. If it determines that the card is not a payment card, the full card number is returned. Otherwise, nothing is returned. The same policy is also used to filter key presses but without the additional context of the track data that is present in a card swipe.

We have found that the default policy that is used by Payment Server is appropriate for most situations. However, there are cases where non-payment cards will be indistinguishable from payment cards. In this situation, Payment Server will not return those card details, and the client application will not work as intended. If this happens, please contact Vista to see if we are able to create a custom card-release policy for you. While we take great care with development of these policies, they will not have been tested by our PCI assessors and are considered custom development. Please consult with your PCI QSA to discuss the implications for your PCI assessment.

If you are creating cards that you intend to process via an MSR, to avoid having to create a custom card-release policy, have the card details vary from the format used by payment cards.

For example:

- Use a card number that is longer than what is allowed for payment cards (20 digits or more).
- Avoid known payment-card BIN ranges.
- Exclude track data that is required to be present for payment cards (for example, expiry date).

Payments Service

As part of installing Payment Server, a Service Framework web service called Vista Payments Service is installed by default but can be excluded if not required. This service is a read-only service used by some clients to retrieve their payment configuration rather than directly from the database. The only clients that use this at present are Serve and Rove. If you are not using these applications, you do not need to install this service.

When browsing from a server that has Vista Payments Service installed, API documentation for the service can be found at the following URL:

http://127.0.0.1:18890/PaymentsService/{installed_version}/swagger/ui/index

At the time of writing, the latest version to use in this URL is "1.1.2.0".

Accessing logs for Payment Server

Payment Server makes log entries available in several ways. It has its own specific logging location, and it also makes log entries available to the client application.

Before any message is written to a log file, card numbers, track data, and CVVs are truncated and are not written to disk in full even at the most detailed logging levels.

Client-application logs

Payment Server makes log entries available to client applications so they can be included in their logs. While Payment Server is logging-system agnostic, the standard logging provider used by Vista applications is Nlog, and logging is usually controlled by the nlog.config file. Payment Server uses the logger name “visPaymentModule” and provides the following information at these logging levels:

- Error: Errors with trace information
- Debug: Bank failures with trace information
- Trace: All requests with trace information

For more information on configuring Nlog logging, please refer to the Nlog documentation.

While the exact format of log entries is dependent on the nlog.config file, the body of log entries provided by Payment Server will be similar to the format provided below.

Database logs

Transaction summary information is written to tblPaymentLog. Information is written to this table several times during a transaction. A list of the fields available in this table and their descriptions can be found in Appendix H.

Payment Server logs

Prior to making logs available to client applications, Payment Server managed its own set of logs that were written to at the end of the transaction. These logs are now disabled by default but can be re-enabled if required. For on-premises installations, these logs are written to the PaymentModule\Log directory in the Vista installation directory. This is usually C:\Vista\PaymentModule\Log\. For a Vista Cloud installation, these log files are written to the program data directory, usually C:\ProgramData\Vista\PaymentModule\Log\.

Payment Server uses two log files in this directory, ErrorlogA.txt and ErrorlogB.txt. Payment Server will first write to one of these files until it has reached 50kb in size, and then it will switch to the other file. If the other file already exists, it will be deleted and overwritten. In the rare case where there is a problem accessing one of these files, you may see files with an ErrorLogBusy_ prefix.

The amount of information that is written to these logs is controlled by the LOGGING_LEVEL control setting (see Pay_strControlSettings1 in Appendix B). All parameters passed to Payment Server and returned by the payment connector are included in the logs along with Payment Server configuration settings. If client application logging is not available to Payment Server and either LOGGING_LEVEL is not set or is set to ‘Off’, then it will use the logging level ‘2T’.

- Off: No logs are written.
- 0: Only log when there is an error.
- 1: Log when there are errors and bank failures.
- 2: Log all requests.
- 2T: Log all requests with trace information.

Log entries in ErrorLog.txt files have the following format:

Note: VPM stands for Vista Payment Module.

```
=================LOGGING DEBUG INFO================
[Date and time of log entry]
[Code version information]
[Client ID]
[Client application type]
[Request type]

INPUT PARAMS PASSED BY CLIENT
[A list of input parameters provided by the client]
-----------------added by VPM-----
[Further input parameters generated by Payment Server as part of the request]

OUTPUT PARAMS - Generated by ServiceModule and/or VPM
[A list of output parameters, generated by the payment connector or Payment Server]
```

Integrating Payment Server with a SIEM service

Importing logs into a security information and event management (SIEM) system can be accomplished through adding Nlog targets in the nlog.config files of client applications. Several SIEM providers or third parties, such as Splunk and Datadog, offer these Nlog targets. Nlog maintains a list of available integrations on their website.

Additional configuration for Payment Server

Other places where Payment Server can be configured are tblConfigure and tblPaymentModuleSettings. See Appendices E and F for more details.

Appendices

Appendix A: Sensitive data processed by Payment Server

Data processed	Why we need it	Where we keep it	How we protect it	Length of time we keep it for
Truncated PAN (6.4)	Reprinting receipts Matching bookings to a card number Managing transactions	In RAM In Payment Server log files In VISTA database Returned to client applications, and may appear in log files	No additional protections	Client applications determine how long this is kept in the database. Please see the *Data Privacy Feature Guide* for details of how to purge data.
Full PAN/Track 1/Track 2/CVV	Facilitating payments when MSR is used	In RAM	Card details are held encrypted at rest in RAM through .NET's SecureString class, which uses DPAPI encryption. In the clear in RAM when required for transmitting to payment gateway TLS 1.2 is used for all communication to payment gateways that require full PAN.	Cleartext card details are removed from RAM after transmission to the payment gateway. Encrypted card details are held until a new transaction is started.
Expiry date/card name	Matching bookings Managing transactions	In RAM Returned to client applications	No additional protections	Client applications determine how long this is kept in the database. Please see the *Data Privacy Feature Guide* for details of how to purge data.
Middle PAN digits	Recovering from power failure before payment authorisation completes	On disk: $DataPath\\ tblTrans_Temp.txt	DPAPI encrypted	Until overall transaction is completed, or power failure rollback is done
Payment card proxies/tokens	Processing payments for checks when card is presented	In DB tblSecureStore	AES CBC encrypted	When check is completed.

Data processed	Why we need it	Where we keep it	How we protect it	Length of time we keep it for
	at the start of the order			
Payment configuration	Stored locally on the client workstation so Payment Server can operate when disconnected from the database server on startup	On Disk: $DataPath \CachedDatasets	DPAPI encrypted	Overwritten on application start
SQL Insert statements for the following tables: tblPaymentHistory tblPaymentLog tblPaymentReceipt tblVista_Log	When operating without a connection to the database (in offline mode), SQL Insert statements are stored for inserting into the database when connectivity is restored.	On Disk: $DataPath\ QueuedDbWrites	DPAPI encrypted	Removed once SQL statements have been applied to the database

Appendix B: tblPaymentModuleConfig fields

Field	Description
Pay_strClientId	The client ID this configuration applies to. Set to "-" to apply to all clients.
Pay_strApplicationId	The application this configuration applies to. Set to "-" to apply to all applications.
Pay_strCardType	The card type this configuration applies to. Set to "-" to apply to all card types.
Pay_strBankService	If using .NET binding, the .dll extension is added to this name to find the assembly to load. If using COM binding, .PMServiceInterfaceV2 is added to this name to find the class to create. This can be set to "NOTACCEPTED" to specify that a specific card type is not accepted, as opposed to not being configured.
Pay_strServiceSettings1/2/3	A delimited collection of settings that are specific to the payment connector. It's not important which of these fields a setting is in, as they are all combined and read as one set of settings.

Field	Description
	While a pipe (|) is the traditional delimiter used, *any* first character in this field will be treated as the delimiter. Setting names and values are separated by an equals sign (=). Example: |Name1=Value1|Name2=Value2| See connector documentation for available settings.
Pay_strControlSettings1	A delimited collection of settings for Payment Server. While a pipe (|) is the traditional delimiter used, *any* first character in this field will be treated as the delimiter. Setting names and values are separated by an equals sign (=). Example: |Name1=Value1|Name2=Value2| See Appendix C for available settings.
Pay_strRemoteServerName	If using a COM connector, the name of the server where the object will be created. If empty, the local computer is used.
Pay_strIsOffline	Should not be edited manually; controlled by Payment Server. This is set to Y if the payment gateway is offline.
Pay_dtmOfflineStateChanged	Should not be edited manually, controlled by Payment Server. The time when Pay_strIsOffline was updated
Pay_strSettlementSettings	A collection of settings that are specific to settlement. Unlike other fields in this table, the delimiter for these settings is always pipe (|). Setting names and values are separated by an equals sign (=). Setting names are not case-sensitive. Example: |Name1=Value1|Name2=Value2| Pay_strApplicationId must be set to “VISTA_SETTLE”, and Pay_strClientId must be set to “-“. See Appendix D for all available settings.
Cinema_strCode	(Veezi only) The cinema code that this payment configuration applies to.

Appendix C: Payment control settings

This is the list of possible delimited settings that are available for Pay_strControlSettings1 in Appendix B.

Setting	Data type	Default	Description
ADMINABLE	Y/N	N	If Y, indicates that this payment connector supports admin functions. When asked to access admin functions, Payment Server will look for all connectors with admin functions. If there is only one connector, that will be selected. If there is more than one connector but only one with ADMIN_DISPLAY_NAME set, that connector will be selected. If there are no connectors with ADMIN_DISPLAY_NAME set, the first one will be selected. If more than one connector has ADMIN_DISPLAY_NAME set, a form with all connectors with ADMINABLE set will be displayed.
ADMIN_DISPLAY_NAME	String		Ensure this is set for all connectors with ADMINABLE set to Y. This is the name shown in the form that allows users to select which connector they want to access admin functionality for. If this is not set and the form is shown, the value from Pay_strBankService will be used instead.
ALWAYS_PRINT	Y/N	N	If Y, a print stream will be generated even if there was an error in the transaction.
ALWAYSGENTRANSNO	Y/N	N	If Y, a new transaction number will be generated for every request.
APPLYSURCHARGE	Y/N	N	If Y, Payment Server will check whether a surcharge applies to the card used in this transaction. If there is a surcharge and it is less than the amount of the transaction, the surcharge will be added to the transaction. If N, no surcharge will be applied even if one is configured for the card used.
BIND.NET	Y/N	N	If Y, the payment connector will be loaded as a .NET assembly. Otherwise, the payment connector will be loaded using COM. Unless the connector only supports COM, set this to Y.
CLEARREPRINTSTREAM	Y/N	N	If Y and a receipt reprint is requested, the print stream will be cleared so you don’t get a receipt.

Setting	Data type	Default	Description
ENABLE_AUTOMATIC_GRATUITY	Y/N	N	If Y and the transaction is being made by POS with a recommended tip, and Payment Server and POS are online, an additional call will be made to the payment connector to set the tip amount.
ENABLE_MULTIPLE_INTERMEDIATE_PRINT	Y/N	N	Ensure this is set to Y for all payment connectors that can return more than one customer verification/signature receipt.
IGNORE_EXPIRY	Y/N	N	If Y, ignore the expiry date on the supplied card.
IGNORE_RETURNED_CARDTYPE	Y/N	N	If Y, Payment Server will ignore the CARD_TYPE value returned by the payment connector.
INPROGRESSVOIDABLE	Y/N	N	Set to Y if this connector is capable of processing a void of a financial transaction that is in progress.
LOGGING_LEVEL	String	OFF	Controls what is written to the Payment Server log files. This has no impact on Payment Server–related information that is written to host-application log files. Logs are written on the following events according to these log levels: • Off: No logs are written • 0: Only log when there is an error. • 1: Log when there are errors and bank failures. • 2: Log all requests. • 2T: Log all requests with trace information.
LUHN10CHECK	Y/N	Y	If Y and the card category used is CREDIT, a LUHN10 check will be performed on card numbers.
MAX_HOURS_OFFLINE	Integer	72	After this number of hours being in offline mode, Payment Server will refuse all future payment requests.
POWER_FAIL_RECOVERY	Y/N	N	Set to Y when the payment connector supports Vista power-fail recovery. When POS restarts after it has stopped unexpectedly during a transaction that has yet to be completed, it will send a request to Payment Server to perform a power recovery. Payment Server also records the last transaction number used. When POS requests a power-fail recovery, Payment Server requests that the connector determine the status of the last transaction number. If the last transaction was a successful financial transaction, the connector will void it.

Setting	Data type	Default	Description
			If the status of the last transaction cannot be determined, POS will ask the operator if a void should be attempted.
PRINT_RECEIPT	Y/N	Y	If N, the receipt is still generated, but the VPMCMD_PRINTCUSTRECEIPT output parameter is set to N to indicate to client applications not to print a receipt automatically.
RECORD_TRACE_TIMING	Y/N	N	If Y and the payment connector has returned trace timing information, this trace information will be recorded in tblPaymentLog
RECORD_ZIP	Y/N	N	If Y and a ZIP code is provided, it will be added to the payment details in tblPaymentHistory
SETTLEMENTREQUIRED	Y/N	N	Ensure that this is set to Y for any payment connector that requires manual settlement
SKIP_PAYMENT_LIMIT	Y/N	N	If Y, all payment-limit checks will be skipped for this connector.
SMALLTICKET_LIMIT	Integer	0	Amount in cents up to which value (inclusive) the sale is considered "small ticket".
SMALLTICKET_PRINTCUSTRECEIPT	Y/N	Y	If N and this is a “small ticket” transaction (see SMALLTICKET_LIMIT), and the card details for the transaction have come from an MSR card swipe, VPMCMD_PRINTCUSTRECEIPT will be set to N to indicate to client applications that they should not print the receipt automatically.
STF_ACTIVATE_WHEN_DELAY_EXCEEDS	Integer	90	The number of seconds of processing a payment-connector request after which the gateway is considered to be offline and store and forward may be used
STF_FAILOVER_ENABLED	Y/N	N	If Y and the payment connector returns the GATEWAY_STATUS parameter with a value of UNAVAILABLE, or if the response time from the gateway exceeds STF_ACTIVATE_WHEN_DELAY_EXCEEDS, future transactions will use store and forward
STF_MAXAMOUNT	Integer	0	Max amount in cents (inclusive) that will be accepted when using store and forward.
STORE_POS_RECEIPT_DATA	Y/N	N	If Y, receipts for POS transactions that aren’t bookings will be recorded in tblPaymentReceipt. This setting has no effect on other Payment Server clients, and all receipts for those clients will be written.

Setting	Data type	Default	Description
USE_RETURNED_CARD_NUMBER	Y/N	N	If Y, and a CARD_NO parameter was included, store the card number returned by the payment connector in tblPaymentHistory. If N, store the card number from the input parameters in tblPaymentHistory.
USECARDCATPARM	Y/N	N	If Y, checks the specified Card Category matches the Card Category supplied by the PAN-range definitions in tblCardDefinition for the discovered Card Number. Used with USEEXTERNALSWIPE for when returned card numbers aren't in any predictable range.
USEINTERMEDIARY	Y/N	N	If Y, the visPayLink payment connector will be used instead of the one specified in Pay_strBankService
VALIDATE_PROMOTION_CARDS	Y/N	N	If Y, the truncated card number returned from the payment connector will be validated against the truncated card number passed *to* the payment connector. If they do not match, the transaction will fail.
VOID_PASS_THRU	Y/N	N	When the POS operator uses the DELETE button with a payment line highlighted, POS will request Payments Module void the payment. When Payments Module is asked to void the payment and this control setting is N, the connector will be asked to do a REFUND. When the setting is Y, the connector will be asked to do a VOID. (The request will be “passed through”.)

Appendix D: Settlement settings

This is the list of possible delimited settings that are available for Pay_strSettlementSettings in Appendix B.

Setting	Data type	Default	Description
SETTLEMENTMODE	String	F	Determines how the settlement process will run: • N (None): The settlement process will not run. • F (Full): All unsettled transactions will be settled. • D (Full by day): All unsettled transactions within a day of the oldest unsettled transaction will be settled. • M (Full partial by day): All unsettled transactions within a day of the oldest unsettled transaction will be settled, in batches determined by BATCHSIZE. • P (Partial): A single batch of unsettled transactions will be settled, determined by BATCHSIZE.
ALLOWRETRYONFAIL	Y/N	Y	If Y, failed settlements will be attempted again.
WAITMINUTESBEFORERETRY	Integer	20	If ALLOWRETRYONFAIL is set to Y, this is the minimum number of minutes that have to pass until a retry is attempted.
MAXDAYSALLOWRETRY	Integer	5	If ALLOWRETRYONFAIL is set to Y, this the maximum number of days after the original failure before no more attempts will occur.
BATCHSIZE	Integer	200	The number of settlements that will be processed on each run of the process. Note: Doesn't apply in full, or full by day, settlement mode.
CHOOSETERMINALTOSETTLE	Y/N	N	If Y, settlement can be made against a specific configured workstation. Otherwise, settlement will be made against all terminals.
DISPLAYNAME	String	Connector name	The display name of the payment connector as shown in the Settlement Utility.
DISPLAYTIPMESSAGE	Y/N	N	If Y and using the Settlement Utility a warning will be shown when attempting to settle that will remind the user that tips will no longer be able to be added to existing transactions once settlement is complete.

Appendix E: tblConfigure settings impacting Payment Server

Setting	Data type	Description
CURRENCYSCALE	Integer	The number of decimal places for currency values

Setting	Data type	Description
LOWESTCASHUNIT	Decimal	Lowest cash rounding unit. Example: In NZ, 10 cents is the minimum coinage.
PRINTUSEANSITOOEM	Y/N	If Y, translate special ANSI characters to the printer's OEM character set. Set to N if running in a country which uses a double-byte character set (DBCS).

Appendix F: tblPaymentModuleSetting

Setting	Description
TRANSNUMBERPREALLOCATIONSTART	The transaction number to begin the allocation of transaction numbers for offline transactions at.
TRANSNUMBERPREALLOCATIONSIZE	The number of transaction numbers available per workstation for offline transactions.
TRANSNUMBERPREALLOCATIONMINFREE	The number of transaction numbers available for offline transactions before a workstation will request a new allocation.

Appendix G: Sensitive resources

A sensitive resource is an external resource upon which software relies to provide security features or process sensitive data. Sensitive resources are often provided by or shared with the underlying platform, operating environment, or other applications that coexist within or outside the software's operating environment.

Payment Server depends on the following sensitive resources:

Resource	Usage
Windows Authentication	Providing access to the software
Windows Data Protection API (DPAPI)	Encrypting data written to disk
Microsoft Authenticode	Validating the authenticity of Vista components
.NET SecureString class	Protecting sensitive data that is held in RAM

Appendix H: tblPaymentLog fields

Field	Description
PayLog_intTransNo	An incrementing identifier for this row.
PayLog_dtmTransDateTime	The date and time of the logged action, in the time zone of the site that the action occurred at.
timestamp	An automatically generated, unique, binary number for the row.
PayLog_strTransType	The type of request made to Payment Server.
PayLog_strCardNo	The 6.4 truncated card number.

Field	Description
PayLog_strSwiped	If Y, the card data was captured via a swipe.
PayLog_intAmount	The monetary amount tied to the request, in cents.
TransC_lgnNumber	A reference to the corresponding row in tblTrans_Cash, or NULL if no such row exists. This transaction number is provided by the client application.
PayLog_strVistaPaymentTransNumber	An identifier for the transaction generated by Payment Server, or NULL if the action does not require a transaction number.
PayLog_strConnectorUsed	The name of the payment connector that was used to complete this action.
PayLog_strClientId	The identifier of the client that made this action.
PayLog_strApplicationId	The identifier of the type of application that made this action.
PayLog_strUserId	If the action was made by a distinct user of the application, such as a POS operator, the identifier of that user.
PayLog_strBankTransNumber	An identifier for the transaction provided by the payment connector, or empty string if this row represents an intermediate log entry.
PayLog_strAccepted	A Y/N value indicating whether the action was accepted by the payment connector, or NULL if this row represents an intermediate log entry.
PayLog_strReturnCode	The result code returned by the payment connector, or NULL if this row represents an intermediate log entry.
PayLog_strReturnText	The result text returned by the payment connector, or NULL if this row represents an intermediate log entry.
PayLog_strFinished	If N, this row represents an intermediate log entry.
Cinema_strCode	The code of the cinema that the action was made at.
PayLog_strTraceTime	An optional output of payment connectors to provide the times taken to perform actions. See RECORD_TRACE_TIMING in Appendix C. NULL if the RECORD_TRACE_TIMING control setting is disabled, the connector doesn’t provide this information, or this row represents an intermediate log entry.
PayLog_decResponseTime	The amount of time taken by the payment connector to carry out the request in seconds, or NULL if this row represents an intermediate log entry.

Field	Description
PayLog_strClientReference	A reference code provided by the client application when making the request, or NULL if no reference was provided.