Monetra® Payment Software

Secure Implementation Guide
(Covering PCI and PA DSS)

Revision: 4.0
Publication date September 23, 2015

Secure Implementation Guide: (Covering PCI and PA DSS)

Main Street Softworks, Inc.

Revision: 4.0

Publication date September 23, 2015

Legal Notice

Table of Contents

1 Monetra Security Validation (PA-DSS)

Since its inception, Main Street has incorporated leading edge security practices and technologies directly into all software offerings.

The following sections outline the Security Audit used to validate Monetra. They also outline the rolls associated with Monetra users, integrators and resellers when securely implementing Monetra into a PCI compliant production environment, as required by the PA- DSS standard.

Version 4.x of this guide (Monetra Secure Implementation) aligns with PCI-DSS and PA-DSS Version(s) 3.1 dated May 2015 and should be used for all new deployments.

This document has been updated to include all relevant information for customers deploying Monetra Version 7 up to update 14.8 and Monetra version 8.x.y

History Note: The original documentation referenced the Visa PABP document v1.1 as posted on their website date: June 01 2005. Version 1.3 of this guide (Secure Implementation) referenced the Visa PABP document version 1.3 released May 08, 2006. Version 1.4 of this guide (Secure Implementation) references Visa PABP document version 1.4 released January 2007. Version 2.0 transitioned all prior Secure Implementation documentation from PABP to PA-DSS 1.2.

1.1 Do Not Retain Sensitive Data

One of the main goals of PCI/PA-DSS is to prevent the risks associated when full magnetic stripe data, security codes(i.e. CVV2 values) and/or PIN blocks are stored after authorization by payment applications. **Monetra does not store Magnetic Stripe, CVV2 (security codes) or PinBlock(PVV) data post-authorization anywhere within the application.**

PA-DSS references	1.1.x
PCI-DSS alignment	3.2, 3.4

Incoming transaction data: Once Monetra receives sensitive authentication data it is forwarded to the EFT processor and erased. **See Integrator notes below.**

Transaction logs: Monetra does not store sensitive authentication data in transaction logs.

History files: Monetra does not store sensitive authentication data in history files.

Debug logs: Under production settings, Monetra does not output sensitive authentication data in debug logs. **See configuration and integrator notes below**, alongside Section 2.3 of this document, which explains the logging subsystem in more detail.

Audit logs: Monetra does not store sensitive authentication data in audit logs.

Database schema's and tables: Monetra does not store sensitive authentication data inside the database.

1.1.4 When properly configured for a production environment, Monetra does not store sensitive data. Note: If you use 'drop files' or have operated Monetra outside of a standard configuration (i.e. a monetra.log setting in a non-pci mode) while sending live transactions, then it is the integrator and/or users responsibility to securely delete the historical data. **See Integrator notes below.**

1.1.5 While Main Street utilities like the Monetra installer will handle secure upgrades of cryptographic materials (i.e. Keys), it is the integrators responsibility to ensure cryptographic material is properly removed when no longer in use. **See Integrator notes below.**

1.1.5.c Main Streets policy is to collect as little information as possible, only that which is required to help solve any particular support problem. In most cases sensitive data is not required when troubleshooting. In any case where sensitive data is required and forwarded to our company for support purposes, it is handled in accordance with PCI/PA-DSS. It will be stored in a specific location with limited access, encrypted and securely removed when no longer required.

CONFIGURATION NOTES

The monetra.log (debug) output level is configurable. Please reference Section 2.3.1 for information regarding external log settings and PCI-DSS compliance.

INTEGRATOR NOTES

- The integrator is responsible for securely removing historical data elements.
- The integrator is responsible for maintenance of cryptographic materials. Please reference Section 2.8 for information on cryptographic key management.
- If you are instructed to forward a sensitive log file to Main Street, it is required that the communication be encrypted, and that it is securely deleted immediately after use. If you, as an integrator, troubleshoot remote Monetra systems and have sensitive data transferred then you must adhere to PA-DSS 1.1.5.c
- The integrator is responsible for securely passing all sensitive authentication data (i.e. magstripe/CV/pin block) to the Monetra application and receiving and destroying any sensitive information returned (i.e. account numbers, exp. dates) by the Monetra application.

When integrating an application with Monetra, there are several communication methods that can be used with both secure (over public transport) and non-secure (over private transport) to choose from. Currently Monetra provides both Key/Value pair and XML based protocols that can be communicated to the Monetra engine via TCP/IP, SSL, HTTP/HTTPS and Drop- File.

When you are communicating with Monetra over a public network (i.e. the Internet), you should always enable and use a secured connection either through our built in SSL or HTTPS facilities or a secure link such as VPN.

NOTE REGARDING DROP FILE INTEGRATIONS: The use of drop file communications with the Monetra Engine should be considered the least secure method. While drop files provide an easy way for quick administration, legacy application integration and system testing, the use of this method should be limited to testing and should only be used in a production environment in which you cannot use one of the more secure methods. Developers should highly consider upgrading any drop file integration to an IP based method, and all new integrations should be considering either IP or HTTP (dependent on protocol/toolkits used).

Note: Please review Section 2.10 regarding Alternate file system security.

If you use drop files, the following requirements will apply:

1. When a file is written into the designated trans directory, the Monetra application will remove it within a configurable amount of time. When the response file is written back, it is the application's responsibility to destroy or encrypt that file.

2. Care should be taken to apply proper security via permissions on any shared folder/ directory. For example, only give the Monetra User and the Integrated Application read/ write permissions on the folder.

1.2 Stored Data Protection

PA-DSS references	2.1
PCI-DSS alignment	3.1

Keep cardholder data storage to a minimum:

PCI Notice: Develop a data retention and disposal policy. Limit storage amount and retention time to that which is required for business, legal, and/or regulatory purposes, as documented in the data retention policy (i.e. purge data that is no longer required for business purposes). Please see Section 2.3 of this guide for details on data location and removal procedures.

PA-DSS references	2.2
PCI-DSS alignment	3.3

Mask displayed account numbers: Monetra provides the ability to mask transaction report data based on the RBAC security level of the application user. For example, the account manager might need access to pre-settlement data, whereas a day-to-day clerk would not.

PCI Notice: Since Monetra is a true client/server application all data is returned via the protocol. All graphical clients provided by Main Street use this protocol. When the Monetra account is initially set up the administrative (base) account will have the ability to retrieve un-obscured data. It is recommended you only use the administrative account to administer sub-users and when you add a new sub-user to the system, you should include the OBSCURE=YES (key value pair) with the request and thus no un-obscured data would ever be returned for that account.

PA-DSS references	2.3
PCI-DSS alignment	3.4

Encrypt stored sensitive data: Monetra's encryption policies take effect at the application level and are user-configurable. Both the encryption algorithm (cipher) used and a key length(strength) may be specified. Currently, Monetra can utilize Blowfish, AES, RC4, RC5, and CAST5 ciphers. The allowable size (strength) of the encryption key varies depending on the capabilities of the cipher itself, ranging between 8 bits and 2048 bits. The recommended minimum key length for all algorithms is 128 bits, the recommended length is 256 bits. The default algorithm is AES, with a default key length of 256 bits.

Note: Monetra's encryption sub-system can be configured to operate either local (in app via OpenSSL) or remote (via the use of an HSM). Please reference Section 2.6 on HSM's for more information regarding Monetra's remote cryptographic features.

PA-DSS references	2.4
PCI-DSS alignment	3.5

Protect encryption keys: One of the most important aspects of any cryptographic system is the creation, usage, maintenance and support of encryption keys within applications. As of Monetra v7 update 10.0, support for both standard and enhanced key security is available. When Monetra starts up, a series of events are triggered for secure cryptographic key support.

Standard mode:

1. Information is retrieved from the Monetra configuration file (main.conf) for an encryption cipher, key length and key file location.

2. The key file is read into memory and decrypted using the private key, which is stored within the Monetra executable, which itself is encrypted by a symmetric key generated internally.

3. Once the database encryption key itself is decrypted, a key length is checked to ensure it matches the length specified in the configuration.

4. Finally, Monetra attempts to decrypt a single known-value in the database, utilizing the expected good key in order to verify its accuracy prior to startup.

Enhanced mode (optional):

Enhanced mode allows the use of one or more administrator-entered pass-phrases to protect the real encryption key. The implementation uses a K/N pass-phrase scheme, where N is the total number of pass-phrases that protect the key, and K is the total number of pass-phrases required to be entered to unlock the key. Pass-phrases are SHA-256 hashed, and are used as input into AES-256 to protect their assigned payload.

Note: Monetra's encryption sub-system can be configured to operate either local (in app via OpenSSL) or remote (via the use of an HSM). Please reference Section 2.6 on HSM's for more information regarding Monetra's remote cryptographic features.

PCI Notice: Encryption keys used to secure cardholder data should be stored securely in the fewest possible locations, and access to keys must be restricted to the fewest possible custodians.

PA-DSS references	2.5
PCI-DSS alignment	3.6

Implement key management processes and procedures: Monetra's encryption policies take effect at the application level and are user-configurable. Both the encryption algorithm (cipher) used, as well as a key length(strength) may be specified. Currently, Monetra can utilize Blowfish, AES, RC4, RC5, and CAST5 ciphers. The allowable key length (strength) is dependent on the Cipher used. Note: If using the Monetra Installer application, cryptographic keys are properly handled for all Upgrades and application Re-Installations.

Note: Monetra's encryption sub-system can be configured to operate either local (in app via OpenSSL) or remote (via the use of an HSM). Please reference Section 2.6 on HSM's for more information regarding Monetra's remote cryptographic features.

Please see Section 2.8 of this guide, for detailed information regarding Encryption keys.

PA-DSS references	2.6
PCI-DSS alignment	3.6

Guidance regarding the removal/deletion of cryptographic material:

PCI Notice:

1. All cryptographic material no longer in use MUST BE REMOVED.
2. Cryptographic material must be securely deleted from the system, in accordance with industry best practices associated with the computing environment in which the application operates.
3. Removal of old cryptographic material is REQUIRED FOR PCI-DSS COMPLIANCE.

Please review Section 2.8 for instructions on key management/replacement.

CONFIGURATION NOTES

Please review the information contained within Section 2.8 and associated documentation for information on creating and maintaining strong encryption keys.

INTEGRATOR NOTES

It is the integrator's responsibility to ensure all initial encryption keys are created and ongoing encryption key maintenance is supported.

1.3 Provide Secure Password Features

PA-DSS references	3.1
PCI-DSS alignment	8.1, 8.2 and 8.5.8 ~ 8.5.15

Require unique/strong usernames and passwords for administrative access: Monetra provides multi-level username and password facilities for both administrative and general application access.

PCI Notice:

- Customers, resellers and integrators are advised against using the default MADMIN account for payment application logins. (e.g., don't use the 'sa' account for payment application access to the database).
- Customers, resellers and integrators are advised upon initial login, Monetra will force you to change/secure the default MADMIN password.
- Customers, resellers and integrators are advised to review chapter 5 (Monetra user subsystem) and assign secure authentication for Monetra and associated systems.
- Customers, resellers and integrators are advised that by default, Monetra will enforce the use of secure authentication, as outlined in PCI-DSS 8.5.8 through 8.5.15.
- WARNING: If you disable secure authentication within Monetra, it is still your responsibility to comply with 8.5.8 – 8.5.15 If you do not comply with 8.5.8 – 8.5.15 your systems will not be compliant.

PA-DSS references	3.2
PCI-DSS alignment	8.1, 8.2

Require a unique username and complex password for access to PC's, Servers, and databases where payment applications reside: Note: Monetra is provided as a single stand-alone software application. Requirement 3.2 should be implemented and enforced at the systems admin level. We strongly advise the use of controlled access, via unique username and PCI compliant secure authentication. Please reference PCI standard 8.1 and 8.2

PA-DSS references	3.3.x
PCI-DSS alignment	8.4

Encrypt application passwords: Monetra uses strong encryption to store application passwords. Also, by default Monetra exposes the use of SSL/TLS or HTTPS as a communication method that must be deployed in a production environment, or the use of an alternate secure method (i.e. VPN etc) must be in place if using TCP/IP.

CONFIGURATION NOTES

Monetra stores passwords using strong cryptographic techniques. The default storage mechanism leverages the same encryption algorithm and data encryption key as is used to encrypt and store any sensitive cardholder data, such as PANs. Care is taken to ensure all password comparisons are not susceptible to side-channel attacks such as timing attacks. Use of the same encryption key ensures that the overall security of the system is segmented into one subsystem which reduces the overall vulnerability footprint. This default password protection mechanism is configured in `main.conf` via `password_protection=Encrypt`.

Alternatively, integrators may choose to add in additional security features for password storage to try to comply with guidelines imposed by third parties. If it is required that passwords be stored using one-way hashing, then Monetra has the ability to use `PBKDF2 SHA2-HMAC` with a configurable number of rounds. Due to the inherent security risks of brute-force attacks on hashed passwords, especially with the use of modern `GPUs`, `FPGAs` and even custom `ASICs`, Monetra also encrypts this result with the data encryption key. `PBKDF2` may be enabled in Monetra's `main.conf` file by setting `password_protection=EncryptPBKDF2`. The number of rounds can be configured via `password_pbkdf2_iterations` and defaults to 1000.

Note: If `PBKDF2` is enabled, the Monetra POST Protocol cannot be used. In addition, due to the additional hashing overhead, it may impact the overall performance of your system, and with a high number of rounds may introduce a security vulnerability known as a Denial Of Service attack vector. It is not believed that there is any additional security benefit to the use of `PBKDF2` over the standard encrypted passwords, especially since the goal of an attacker would be the cardholder data on file, not the passwords themselves.

INTEGRATOR NOTES

Monetra was designed from the start as a 'stateless' (non persistent/near real-time) application. In short, all requests (functions) into and out of Monetra must be verified (via approved security policy) and should never be considered to act in a persistent (sessioned) state.

1.4 Log Application Activity

PA-DSS references	4.1
PCI-DSS alignment	10.1

Log access by individual users: Monetra provides extensive logging for security audits at both the internal and external level.

PCI Notice: On a default install, Monetra logging is enabled by default. To remain in compliance with PCI-DSS, Security Logging MUST BE ENABLED while in a production environment.

PA-DSS references	4.2
PCI-DSS alignment	10.2

Implement an automated audit trail: Monetra provides extensive logging both at the connection and transaction level. PCI complaint Security logging is set to ON by default.

PA-DSS references	4.3
PCI-DSS alignment	10.3

Implement detailed output for audit: Monetra provides extensive logging both at the connection and transaction level. At a minimum Monetra logs UserID, Event Type, Date/ Time-stamp, response (success/fail), origin of request, identity/name of affected system/ component.

PA-DSS references	4.4
PCI-DSS alignment	10.5.3

Facilitate centralized logging: Monetra can be configured to log locally (via internal log facilities) or to use Syslog for remote logging.

For more information please see Section 2.3 of this guide for logging details.

CONFIGURATION NOTES

Security logging is a configurable parameter. These features must be enabled for PCI compliance and are set to ON by default. Please reference the Monetra Configuration Guide for more information.

INTEGRATOR NOTES

The integrator is responsible for logging and audit trails outside of the Monetra payment application. Example: A POS developer must log all activity inside the order entry application. Once the POS application sends Monetra a transaction, it is logged from that connection level forward.

1.5 Develop Secure Applications

PA-DSS references	5.1
PCI-DSS alignment	6.3

Develop software applications based on industry Best Practices and include information security throughout the software development life cycle:

Note: Main Street software products provide some of the most advanced security features available on the market. All of our flagship products take full (native) advantage of the latest operating platforms. Examples: Monetra is the only product to run NATIVE (i.e. no Java or special runtime libraries required) on Linux, FreeBSD, IBM AIX, Sun Solaris, SCO Unix, Mac OSX and Microsoft Windows.

PA-DSS references	5.1.1, 5.1.2, 5.1.3

PCI-DSS alignment	6.4.3, 6.4.4, 6.3.1

Live PAN's are not used for testing or development: Monetra is designed and applied in ANSI C as a self-contained, fully multi-threaded application. It is the integrators responsibility to ensure the merchant is not using live PANS in a non-production environment.

Removal of test data and accounts: Monetra is designed and applied in ANSI C as a self-contained, fully multi-threaded application. A default install provides a blank (empty) data structure. It is the integrators responsibility to ensure any test accounts and test data structures are removed prior to the system going into live production.

Removal of custom application data (accounts, passwords etc.): Monetra is designed and applied in ANSI C as a self-contained,fully multi-threaded application. A default install provides a blank (empty) data structure. It is the integrators responsibility to ensure any custom application data structures are removed prior to the system going into a live production environment.

PA-DSS references	5.1.4
PCI-DSS alignment	6.3.2

Custom code review: As per documented coding policy and procedures, all software developed by Main Street complies with industry best practices and standards. Monetra software has been extensively analyzed through the use of multiple forensics tools.

PA-DSS references	5.2.x
PCI-DSS alignment	6.5

Develop software securely: As per documented coding policy and procedures, all software developed by Main Street complies with industry best practices and standards. Monetra software has been extensively analyzed through the use of dynamic software development tools (such as Valgrind) as well as multiple digital forensics tools.

PA-DSS references	5.3, 5.3.1, 5.3.2, 5.3.3, 5.3.4
PCI-DSS alignment	6.4.5

Follow change control procedures: As per documented coding policy and procedures, all software developed by Main Street complies with industry best practices and standards.

Documentation of impact: As per documented coding policy and procedures, all software developed by Main Street complies with industry best practices and standards, including documentation of known impact.

Management signoff: As per documented coding policy and procedures, all software developed by Main Street complies with industry best practices and standards, including management signoff.

Operational testing/QA: As per documented coding policy and procedures, all software developed by Main Street complies with industry best practices and standards, including operational testing of changesets.

Back-out procedures: As per documented coding policy and procedures, all software developed by Main Street complies with industry best practices and standards, including change-set backout procedures.

PA-DSS references	5.4
PCI-DSS alignment	n/a

Software Versioning:

The versioning scheme employed by Monetra is formatted as `X.Y.Z`, where each `X`, `Y`, and `Z` components are numeric-only version indicators separated by a period. Each numeric component may be from one to three digits in length. All software distribution updates will result in at least one of the components being updated.

The `X` component of the version indicates the product major version number. The major version component only changes when there are significant feature changes, or the changes impact any part of a security standard, such as PCI PA-DSS.

The `Y` component of the version indicates a project minor version change. The minor version will change when there are minor feature enhancements that do not impact the part of any security standard such as PCI PA-DSS.

The `Z` component of the version indicates a bug-fix release. Bug-fix releases do not change the overall feature-set or functionality of Monetra, but may include security related fixes such as updates to 3rd party libraries (e.g. cryptographic libraries) distributed with Monetra.

Note: **Wildcard Versioning:**

PCI PA-DSS (v3.1+) allows a specific wildcard versioning definition which corresponds to the release which is being validated for compliance. With this release of Monetra, the official wildcard versioning is `8.Y.Z`. The major (`X`) version number component is fixed at `8`, which as per the versioning definition states there will be no major feature changes or changes which impact the PCI PA-DSS standard (e.g. all changes that do not affect the major version number are classified as "no impact" changes). The minor (`Y`) and bug-fix (`Z`) wildcard components comply with the descriptions in the previous section.

1.6 Protect Wireless Transmissions

PA-DSS references	6.1, 6.2, 6.3
PCI-DSS alignment	1.2.3, 2.1.1 and 4.1.1

Monetra provides several methods for securing communications. Many network topologies, including wireless, may take advantage of the provided features. Please refer to the section on secure remote application access for more details.

PCI Notice: If you install Monetra into a wireless environment, wireless vendor default settings must be changed per PA-DSS Requirement 6.1 and secure encrypted transmissions must be implemented as per PA-DSS Requirement 6.2 and 6.3. You must also adhere to PCI-DSS Requirements 1.2.3, 2.1.1, 4.1.1, and 9.1.3

CONFIGURATION NOTES

The Network Administrator should provide security policy and settings when related to wireless technologies (such as access points, bridges and routers).

INTEGRATOR NOTES

When integrated into a wireless environment, Main Street recommends the use of SSL connectivity both *to* the Monetra application and *out* to the processors. See the Monetra Configuration Guide for more details.

1.7 Test for Application Vulnerabilities

PA-DSS references	7.1
PCI-DSS alignment	6.2

Test for application vulnerabilities: Removal of unnecessary and insecure services, applications and protocols: Main Street has been validated to comply with known development processes including, but not limited to:

1. Monitoring and using information from outside security sources for vulnerability assessment and policy.

2. Testing Monetra against new 'identified' vulnerabilities.

3. The ability to timely develop, test and deploy a patch for the primary application security, within a known chain of trust.

CONFIGURATION NOTES

The system administrator must insure any application (service or software) is implemented into, updated and tested within the target environment as accepted policy dictates.

INTEGRATOR NOTES

The integrator is responsible for testing the associated network applications for vulnerabilities outside of the Monetra payment application. Example: A POS integrator must ensure the operating system from which Monetra is executing has the approved security updates/patches in place.

1.8 Facilitate Secure Network Implementations

PA-DSS references	8.2
PCI-DSS alignment	1, 3, 4, 5 and 6

Facilitate secure network implementations: Monetra provides the following tools to assist in secure network deployment. Note: Monetra does not interfere with devices, applications or configurations required for PCI-DSS compliance.

1. Direct SSL socket connection method.
2. Direct XML HTTPS connection method.
3. Digital certificate validation (per connection).
4. Digital certificate validation (per merchant/user).
5. ~~Internal firewall for defined (extended) application access validation.~~ NOTE: As of Monetra Version 7 Update 11.0 the Internal firewall has been deprecated and replaced with auto blacklisting features.
6. Automatic blacklisting feature for failed attempts at login. (As of v7 update 11.0)
7. Multi-port aware. Monetra can be configured to allow certain requests (like MADMIN) to operate on one port while standard transaction types can be run on another. Great feature for Administrative access restrictions at the firewall level. (As of v7 update 11.0)

CONFIGURATION NOTES

Monetra connection parameters are a configurable option. Please reference the most recent Monetra Configuration Guide for more information.

1.9 Never Store Cardholder Data on the Internet

PA-DSS references	9.1
PCI-DSS alignment	1.3.7

Provide payment applications and data separation facilities: Monetra provides the ability to separate both the application and the database from any particular environment. For example, Monetra does not require the client (requesting POS) application or the database (required for storage and parameters) to be located on the same system/network as the Monetra payment server.

CONFIGURATION NOTES

The Monetra database (used for system logs and parameter storage) is a configurable parameter. Please reference the most recent Monetra Configuration Guide for more information. To remain compliant with PCI standards, the cardholder data must never reside on Internet-accessible systems (DMZ). The installation must be configured such that Monetra and the database reside on secure(firewalled) network segments (inside the DMZ [De-Militarized-Zone]). By default Monetra uses IANA assigned ports 8665 (user requests) and 8666 (administrative requests). Depending on the database used and where it resides on the network will define which ports need to be opened for database access.

INTEGRATOR NOTES

The integrator is responsible for proper Monetra network configuration, including secure communication channels to and from the Monetra application data storage mechanism(s).

1.10 Facilitate Secure Remote Application Access

PA-DSS references	10.1
PCI-DSS alignment	8.3

Provide Secure remote application access: Monetra provides a native SSL socket connection that can also authenticate against a user certificate. Please reference the appropriate documentation regarding certificate procedures.

Note: Additionally, Monetra provides an internal black-list to protect against unauthorized application access attempts.

PCI Notice: When customers access Monetra remotely, it is a requirement to use two factor authentication.

CONFIGURATION NOTES

All Monetra connection methods are configurable. Please reference the most current Monetra Configuration Guide for more details.

INTEGRATOR NOTES

PCI Notice: The integrator is responsible for the secure configuration of any Monetra application, in regards to remote access and must at all times follow PCI DSS guidance such as change default settings in the remote-access software (for example, change default passwords and use unique passwords for each customer), allow connections only from specific (known) IP/MAC addresses, use strong authentication and complex passwords for logins, enable encrypted data transmission according to PA-DSS Requirement 12.1, enable account lockout after a certain number of failed login attempts, establish a VPN connection via a firewall before access is allowed, enable the logging function, restrict access to customer environments to authorized personnel and any future requirement by the PCI Council that may be missing from this document.

Two factor authentication examples:

1. Application-Username/Password, SSL with certificate(s).

2. Application-Username/Password, VPN with certificate(s).

3. User-Radius, tokens.

PA-DSS references	10.2

Remote Access and Updates: Monetra is delivered via a PULL method using the Monetra Installer. Main Street will never ask you to allow remote access into your network as it is not desired nor required.

1.11 Encrypt Traffic Over Public Networks

PA-DSS references	11.1, 11.2
PCI-DSS alignment	4.1, 4.2

Provide Strong encryption for data transmission over public networks: Monetra provides built-in SSL connectivity methods that are approved for use across public networks.

Never communicate sensitive data via unencrypted end user messaging technologies (such as e-mail): By default, Monetra does not communicate any sensitive data via end user messaging technologies.

CONFIGURATION NOTES

All Monetra connection methods are configurable. Please reference the most current Monetra Configuration Guide for more details on PCI compliant settings.

INTEGRATOR NOTES

The integrator is responsible for configuring the Monetra application to communicate securely over a public network. Note: Any logfiles sent to Main Street for support and troubleshooting should not contain sensitive data (i.e. PAN's) unless expressly instructed to do so. Encryption technologies must be used before any data is transferred via email.

1.12 Encrypt All Non-Console Administrative Access

PA-DSS references	12.1, 12.2
PCI-DSS alignment	2.3

Encrypt all non-console administrative access: Monetra allows non-console application access to administrative features via SSL connection methods.

If you are not using a secure console (such as ssh or putty) for remote Administrative access, you must use either the Monetra provided SSL or HTTPS facilities, or an externally configured secure channel (such as VPN) for application access.

CONFIGURATION NOTES

All Monetra connection methods are configurable. Please reference the most current Monetra Configuration Guide for more details on PCI compliant settings.

INTEGRATOR NOTES

The integrator is responsible for configuring the Monetra application to communicate securely while providing administrative access.

1.13 Maintain instructional material for customers and integrators

PA-DSS references	13.1, 13.1.1, 13.1.2

Develop and implement training and communication programs: Main street provides a notifications list for all licensed and registered POC's. Training is available through updated documentation and custom on-site training sessions.

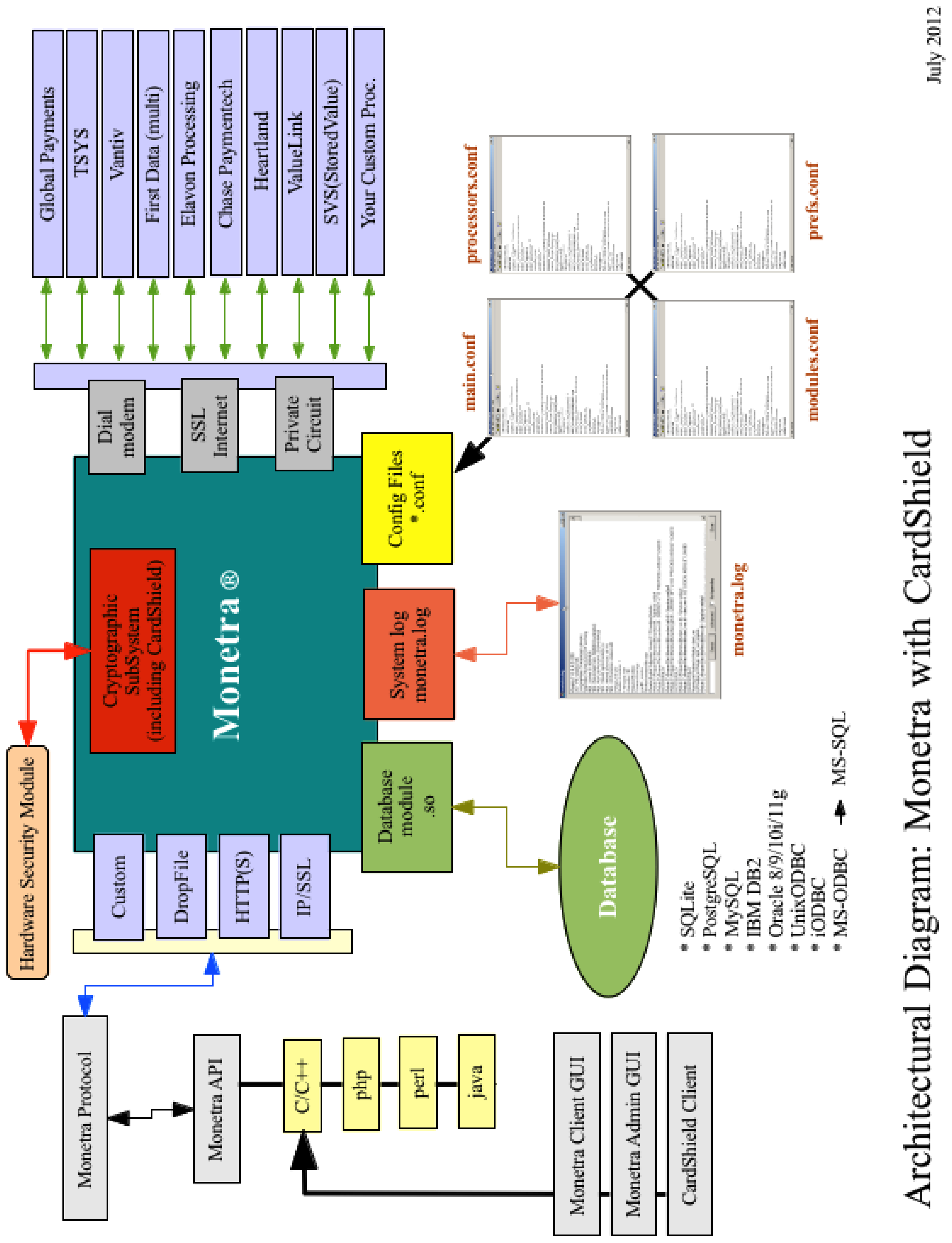
2 Monetra Payment System

2.1 General Overview

The Monetra payment processing software securely links point of sale, e-commerce, and other applications directly to all major transaction processors for performing credit, debit, EBT, gift card and check transactions. Proven effective for thousands of merchants of every type, large and small, in multi-site, corporate-owned/franchisee retail environments, high-volume e-commerce platforms, and more, Monetra delivers quality and performance for every application.

2.2 Architecture

Figure 2.1. Monetra Architecture



2.3 Application Data and Logging

Monetra software provides a modern and robust data subsystem that includes external logging. The system was designed over time to provide transaction auditing, alongside detailed technical transaction tracing facilities for use in troubleshooting, tuning and developer integration.

The system provides two basic types of data. External logging (monetra.log) is provided as a tool to help trace, troubleshoot and tune the system at a transactional level, while the Application (internal) data storage (for reporting) is designed to provide Audit support throughout the transaction life-cycle.

Note: As of Monetra 7 the External logging facility has been enhanced to provide for detailed output in a more structured format. The use of DEBUG= settings have changed and the number format will be deprecated in future versions. Please ensure you have tested any new settings prior to upgrading a production server.

2.3.1 External Logging

The External logging facility (monetra.log) has been designed, and in use for years, as the primary tool for tracing, tuning and troubleshooting a Monetra transaction system. The basic design is one where you can set/adjust certain level(s) of detailed output to be populated into a text file, that can be read (and/or sent to) a technician/developer.

The current external logging system is extremely flexible, and therefore numerous settings can be applied. All parameters are configurable from within the main.conf file and on Unix based systems (including MAC OSX) the logging can be sent to either a defined file by Monetra or handled by the standard SYSLOG process.

LOG SETTINGS: There are several main settings for establishing how the log system works. These not only include a level of output, but also actions and settings for archiving and retention of logged data.

FLAGS AND LEVELS: These settings determine what Monetra outputs during operation. Please take note of the PCI and NON-PCI complaint settings as they pertain to a production and/or test environment.

INIT	PCI	Basic initialization info, such as Monetra version
CONF	PCI	Show configuration and startup details
WARN	PCI	Warnings such as misconfiguration, etc.
INFO	PCI	Un-categorized short information (like stats)
TRAN	PCI	Basic information on when a transaction enters the queue
ERROR	PCI	Any significant error condition
CRIT	PCI	A critical/significant error which must not be ignored
TRAN_DETAIL	PCI	Basic incoming parameters as parsed (with sensitive data sanitized)

CONN	PCI	Log connection details such as IP address, when it is opened/ closed/etc.
PROC	PCI	Log connection info to processors and details when transacting
PROC_DETAIL	PCI	Log more detailed, sanitized, information such as the traces
TRAN_TRACE	Non-PCI	Parsed incoming and outgoing transaction requests un-obscured
TRACE_IN	Non-PCI	Raw trace of data in and out from client connections
TRACE_OUT	Non-PCI	Raw trace of data between Monetra and the processors
SQL	Non-PCI	Raw SQL statements
DEBUG	Non-PCI	Very verbose, un-grouped data
DEV	Non-PCI	Reserved for internal development use only
PROC_TRACE	Non-PCI	Non-sanitized version of PROC_DETAIL

An example of a PCI compliant setting would be as follows.

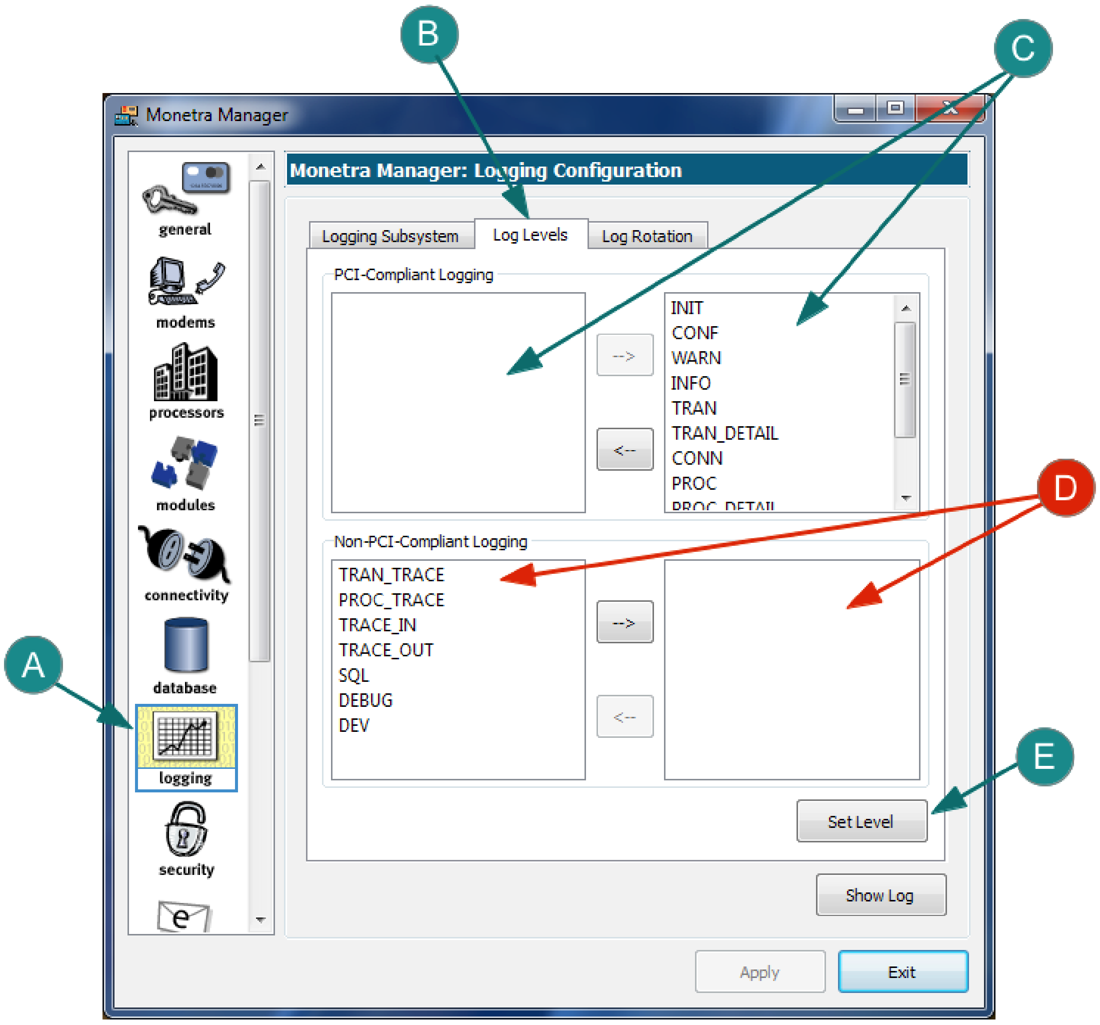
```
debug=INIT|CONF|WARN|INFO|TRAN
```

Note: When you configure a Non-PCI setting for output, upon initial startup the settings WILL NOT BE ACTIVE and Monetra will continue to operate in a PCI-compliant mode.

NOTICE: In order to gain a higher (non PCI compliant) debug level, you must send the 'setlogging' MADMIN command with a 'debug' parameter (i.e list of desired Non-PCI flags) as noted in the table above. Also be aware of the file system type used (NTFS, EXT3 etc.) and the limitations of secure delete associated. If you require the debug output to be elevated in a production environment (very rare) we HIGHLY recommend the use of a ramdisk as discussed in Chapter 8.

The Monetra debug log can be configured via the use of the Monetra Manager (GUI) utility or we provide a handy command-line utility as well.

Set Monetra Log Level



1. Choose the 'logging' section [A] in the left navigation pane of the Manager utility.

2. Choose the 'Log Levels' [B] tab.

3. To turn any PCI-Compliant Level setting ON [C], move it from the left pane to the right pane.

4. To turn any Non-PCI-Compliant Level setting ON [D], move it from the left pane to the right pane. Warning: It is very rare that you should ever need to bump these settings up in a production environment. If you are unsure of what these settings actually do then you should NOT change them.

5. Once all the settings have been moved into the right hand pane, Click on the 'Set Level' [E] button and follow the prompts.

LOG_SYSTEM: These settings determine what process handles logging. Either Monetra (FILE) or the system log daemon(SYSLOG).

```
debugsys=SYSLOG
```

PREPEND: The logging facilities allow for prepending data such as a custom time stamp to each line output. Please reference the most current configuration file (main.conf) for available parameters.

```
debugprepend=%a %D %H:%m:%s.%u %z [%f]:
which produces: Jan 11 09:19:11.426235 -0500 [TRACE_IN]:
```

Note: This feature allows custom structuring of Monetra log data output, which can be helpful when indexing data for future analytics.

FILE_LOCATION: If FILE is the chosen method of logging, then this represents the location of the output directory where the file is written to.

```
debugdir=/usr/local/monetra/
```

NUMBER_ARCHIVES: Monetra provides for logfile rotation, according to a pre-defined schedule. This value represents the number of logfiles for the system to maintain.

```
debugkeep=10
```

ARCHIVE_COMMAND: A method is exposed to call a command (such as bzip2) on a logfile after rotation.

```
debugarchive=bzip2 -f
```

ARCHIVE_EXT: A method is exposed to add an extension to a command (such as bzip2) on a logfile after the archive command is run. For example, if the logfile is bzip2'd, an extension of .bz2 would be necessary for proper rotation.

```
debugarchiveext=.bz2
```

RESTART_ROTATION: A method is exposed to start a new logfile (rotate) when Monetra is started.

```
debugrotaterestart=YES
```

AUTO_ROTATE_DAYS: A method is exposed to allow for log rotation at daily intervals.

```
debugrotatedays=7
```

AUTO_ROTATE_SIZE: A method is exposed to allow for log rotation to be specified once the logfile reaches a specified size. Specified in kilobytes, default is 10240 (10MB).

```
debugrotatesize=10240
```

2.3.2 Application Data Storage

The application storage system (database) provides for additional auditing of the application and detailed reporting on the merchant transactions.

LOCATION: On a default install, Monetra uses a self contained, server-less, zero-configuration SQL database engine called SQLite and its files will reside inside of the ../data/ directory. If you configure an external SQL database (such as Oracle) for use then wherever that database resides is in scope for data storage.

If you use an external SQL database you can securely remove the data by securely deleting the database. WARNING! If you delete an active/production SQL database Monetra will cease to operate!

Described below are the main data stores with notes on their use and tips on how to keep them efficiently operating.

The first set described below is for the Engine Administrative user (MADMIN) and must be called by a privileged user.

errorlog	Transaction Communication Errors	Lists raw communication errors on per-engine basis

This next set describes the User (per merchid/termid) logs that must be called by an Administrative User Request.

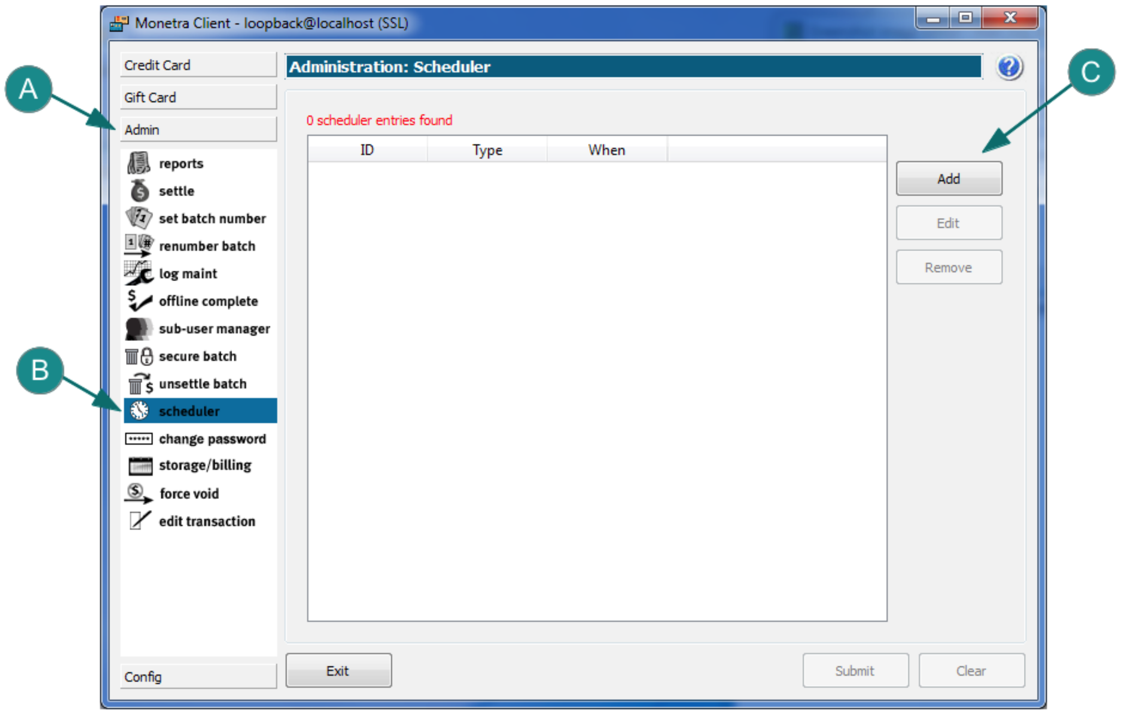
GL	Get Log (settled)	Returns all settled transactions for the requested user (merchant) account. Note: This log can be cleared using the Clear Transaction History (CTH) function. Note2: By default, on successful settlement upload, or forced settlement local, Monetra retains enough data (securely encrypted) to restore a batch to an unsettled state, if batch recovery procedures are required. Hint: You can identify which batches are in an a Reversible state by the associated Y/N flag. Warning: Once you are comfortable the transaction batch(es) are properly settled into your bank, and/or at a standard Business Process timeline (i.e. every 90 days) you should REMOVE or SECURE the Monetra transaction history. To remove the data altogether use the (CTH) function mentioned above. To keep

		the history, but secure the transactions so they MAY NOT BE UNSETTLED, you would issue a Secure Transactions (securetrans) function.
GUT	Get Unsettled Transactions	Returns all unsettled transactions for the requested user (merchant) account. Note: This report is cleared once the batch has been settled. From there, it will populate the history report (GL) mentioned above.
GFT	Get Failed Transactions	Returns all failed transactions for the requested user (merchant) account. Note: This report can be cleared using the Clear Failed History (CFH) request.

Monetra's automated task sub-system (cron) may be used to set daily tasks designed to automatically purge and/or secure Monetra data.

In the example listed below we show how to setup an automated task that will run once every day of the week (except Sunday) and will purge all data older than 90 days.

Set Automated Data Purge

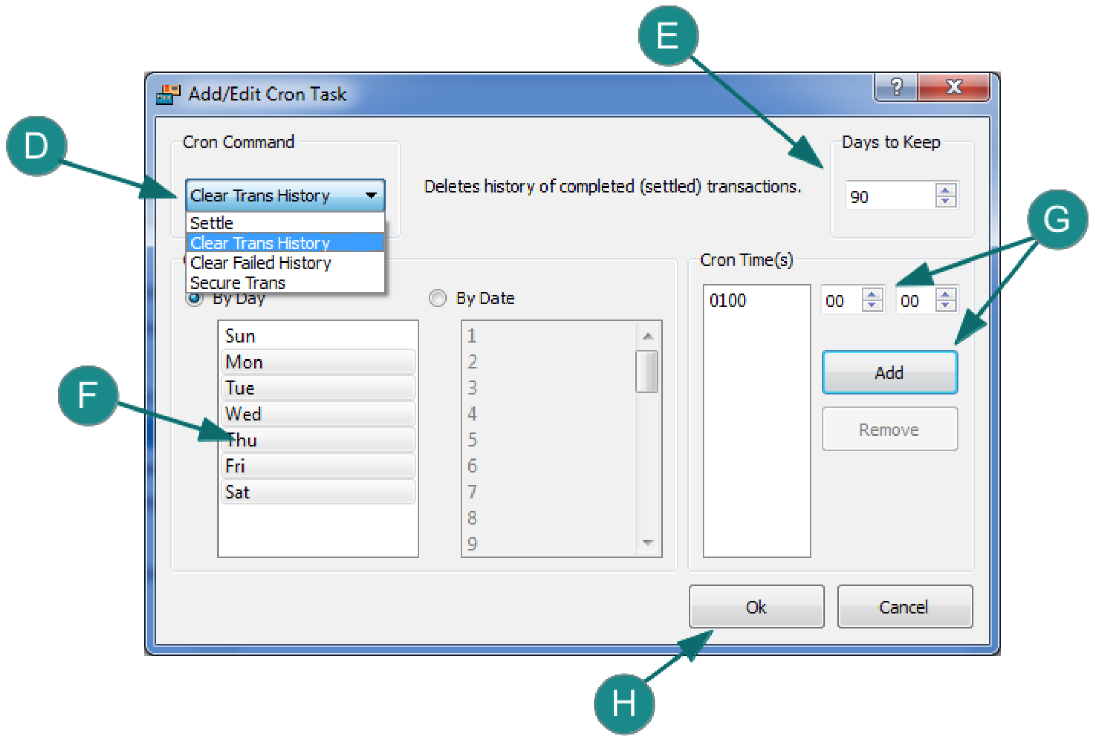


1. Choose the 'Admin' tool-bar [A] in the left navigation pane of the Manager Client.

2. Choose the 'scheduler' [B] section on the left.

3. Click the 'Add' [C] button.

Set Automated Data Purge (continued)



Deletes history of completed (settled) transactions.

Days to Keep: 90

Cron Time(s): 0100 | 00 | 00

1. Choose 'Clear Trans History' from the Cron Command [D] drop-down.

2. Select the number of days worth of data to KEEP by editing the 'Days to Keep' [E] text-box.

3. Select the days you wish the task to run from the 'By Day' [F] control.

4. Choose the time of day you would like the task to run by entering the 'Cron Time(s)' and Clicking the 'Add' button [G]

5. Once all of the settings have been verified Click the 'Ok' [H] button.

2.4 SNMP Monitoring

Simple Network Management Protocol (SNMP) is an "Internet-standard protocol for managing devices on IP networks." At its most basic level it is simply a protocol for collecting and organizing information. It is technically implemented as a protocol within RFC 3411 (and variants) http://tools.ietf.org/html/rfc3411

Why use SNMP?

SNMP provides Monetra administrators with the information they need to pro-actively monitor the state of a Monetra Server. With this data, preemptive actions may be taken (via SNMP reporting) to avoid downtime and minimize the impact of any issues, should they arise. Note: The use of SNMP can also provide excellent statistics for performance tuning your installation.

2.4.1 SNMP Monetra Architecture

Monetra implements support for SNMP v2 via the use of the AgentX protocol as defined in http://www.ietf.org/rfc/rfc2741.txt as follows:

- The agent responsible for collecting Monetra statistics must reside on the same machine as Monetra (localhost).
- Monetra will connect TO the local agent and supply statistics upon request.
- If statistics are required to be collected remotely, the administrator must set up a private connection between the local agent and the remote collector.

Note: SNMP is NOT currently supported for Monetra deployments on Microsoft Windows.

2.4.2 SNMP Data Elements

Monetra currently exposes the following data elements for statistical consumption, archiving and reporting.

Table 2.1. Standard Information

SNMP ELEMENT	DESCRIPTION
`mInfoRunning`	Is the server running? This will be 1 if the Monetra is running.
`mInfoVersion`	Current Monetra Version
`mInfoStartTime`	Date and time the Monetra server was last started, as unix timestamp.
`mInfoUpTime`	Time elapsed since the Monetra server was last started.
`mInfoRequestProcessed`	Number of transactions processed since the Monetra server was last started.
`mInfoRequestPending`	Number of transactions queued for processing.

Table 2.2. Connectivity Stats

SNMP ELEMENT	DESCRIPTION
`mConnectionsActive`	Number of active connections to the Monetra server.
`mConnections`	Number of connections created since the Monetra server was last started.
`mConnectionsFailed`	Number of failed connections since the Monetra server was last started.

Table 2.3. License Related

SNMP ELEMENT	DESCRIPTION
`mLicenseTrans`	Number of transactions the Monetra server license permits.

`mLicenseTransUsed`	Number of transactions processed since yesterday that counted toward license limit.
`mLicenseUsers`	Number of user profiles the Monetra server license permits.
`mLicenseUsersUsed`	Number of user profiles currently configured.

Table 2.4. Monetra Database

SNMP ELEMENT	DESCRIPTION
`mDbQuery`	Number of database queries performed since the Monetra server was last started.
`mDbQueryTime`	Amount of time in milliseconds used in performing database queries since the Monetra server was last started.
`mDbQueryTimeOver05s`	Number of database queries that required more than 0.5 seconds to complete since the Monetra server was last started.
`mDbQueryTimeOver1s`	Number of database queries that required more than 1 second to complete since the Monetra server was last started.
`mDbQueryTimeOver2s`	Number of database queries that required more than 2 seconds to complete since the Monetra server was last started.
`mDbQueryTimeOver5s`	Number of database queries that required more than 5 seconds to complete since the Monetra server was last started.
`mDbQueryTimeOver10s`	Number of database queries that required more than 10 seconds to complete since the Monetra server was last started.
`mDbError`	Number of database errors detected since the Monetra server was last started.
`mDbDisconnect`	Number of database disconnections detected since the Monetra server was last started.

Table 2.5. Cryptographic Operations

SNMP ELEMENT	DESCRIPTION
`mCryptDbEnc`	Number of db encryptions performed since the Monetra server was last started.
`mCryptDbDec`	Number of db decryptions performed since the Monetra server was last started.
`mCryptDbEncDec`	Number of db encryptions and decryptions performed since the Monetra server was last started.
`mCryptDbTime`	Amount of time in milliseconds used in performing db cryptographic operations since the Monetra server was last started.

`mCryptDbError`	Number of cryptographic errors since the Monetra server was last started.
`mCryptDukptLocalDec`	Number of local CardShield (DUKPT) decryptions performed since the Monetra server was last started.
`mCryptDukptLocalDecError`	Number of local CardShield (DUKPT) decryptions errors (failed decryptions) since the Monetra server was last started.
`mCryptDukptLocalDecTime`	Amount of time in milliseconds used in performing local CardShield (DUKPT) decryptions since the Monetra server was last started.
`mCryptDukptHsmDec`	Number of external (HSM) CardShield (DUKPT) decryptions performed since the Monetra server was last started.
`mCryptDukptHsmDecError`	Number of external (HSM) CardShield (DUKPT) decryption errors (failed decryptions) since the Monetra server was last started.
`mCryptDukptHsmDecTime`	Amount of time in milliseconds used in performing external (HSM) CardShield? (DUKPT) decryptions since the server was started.

2.4.3 Installing SNMP

Installing SNMP on Linux

```
$ sudo apt-get install snmpd
```

For additional tools (not required)

```
$ sudo apt-get install snmp
```

Installing SNMP on Mac OS X

Net-SNMP ships by default with OS X. No additional components need to be installed.

2.4.4 Configuring SNMP

2.4.4.1 Configuring SNMP on Linux

Ensure it is setup to accept connections from remote machines.

/etc/default/snmpd

```
SNMPDRUN=yes
SNMPDOPTS='-Lsd -Lf /dev/null -u snmp -I -smux -p /var/run/snmpd.pid'
```

2.4.4.2 Configuring SNMP on Mac OS X

Edit /etc/hostconfig and add:

```
SNMPSERVER=-YES-
```

Edit /System/Library/LaunchDaemons/org.net-snmp.snmpd.plist and change the following value:

```
<dict>
        <key>Disabled</key>
        <true/>
```

To this.

```
<dict>
        <key>Disabled</key>
        <false/>
```

Either restart the machine to have snmpd load or run:

```
sudo launchctl load /System/Library/LaunchDaemons/org.net-snmp.snmpd.plist
```

Configure snmpd

/etc/snmp/snmpd.conf

```
syslocation "Monetra Server"
syscontact admin@domain.com
agentuser  nobody
agentgroup snmp
rocommunity monetra 127.0.0.1 1.3.6.1.4.1.39735
master agentx
agentxsocket tcp:localhost:705
```

This is a minimal configuration using SNMP v2. It allows read only access to the Monetra OID. Change 127.0.0.1 to the address of the machine that will be monitoring Monetra. The monetra community string is used for referencing the information.

Testing:

With Monetra's MIB installed.

```
$ snmpwalk -m MONETRA-SERVER-MIB -v2c -c monetra localhost -Os 1.3.6.1.4.1.39735.100
$ snmptable -m MONETRA-SERVER-MIB -v2c -c monetra localhost -Os 1.3.6.1.4.1.39735.100.1.6
```

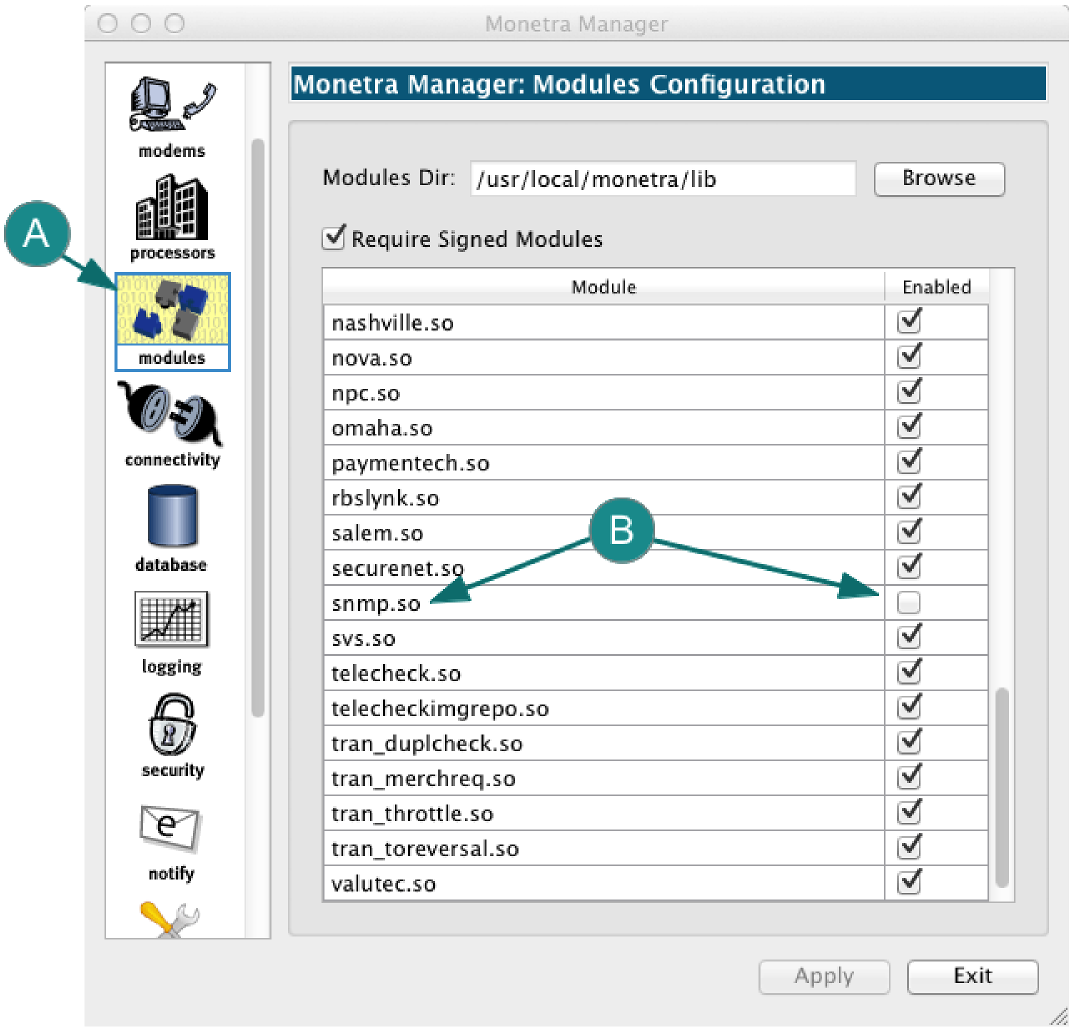
Without Monetra's MIB installed.

```
$ snmpwalk -v2c -c monetra localhost -Os 1.3.6.1.4.1.39735.100
$ snmptable -v2c -c monetra localhost -Os 1.3.6.1.4.1.39735.100.1.6
```

2.4.5 Configure Monetra for SNMP

Monetra implements support for SNMP via a module that needs to be loaded at runtime. As with all other modules these may be configured to load via the Monetra Manager utility or by editing the modules.conf file (loadmodule=snmp.so) and restarting Monetra.

Load Monetra SNMP module



Modules Dir: /usr/local/monetra/lib

✓ Require Signed Modules

Module	Enabled
nashville.so	✓
nova.so	✓
npc.so	✓
omaha.so	✓
paymentech.so	✓
rbslynk.so	✓
salem.so	✓
securenet.so	✓
snmp.so	☐
svs.so	✓
telecheck.so	✓
telecheckimgrepo.so	✓
tran_duplcheck.so	✓
tran_merchreq.so	✓
tran_throttle.so	✓
tran_toreversal.so	✓
valutec.so	✓

1. Choose the 'modules' section [A] in the left navigation pane of the Manager utility.
2. Find the snmp module (snmp.so) [B] and place a check-mark in the 'Enabled' check-box.
3. Click on the Apply button at the bottom and restart Monetra if prompted.

2.5 Monetra Virtual User SubSystem

The Monetra virtual subsystem has been deployed for years to administer, secure and report within the payment application. The base system works as follows.

Note: On a properly licensed Monetra server, an unlimited number of MONETRA USER and SUBUSER accounts may be installed and supported, dependent on production hardware platform and available resources.

A Virtual User is defined by a unique identifier (username) and password combination. When setting user passwords, please use the following guidelines:

1. Passwords should be at least 7 alpha-numeric characters long (and include both characters (such as the *) and letters).
2. Passwords should be changed every 90 days.
3. Passwords should not be repeated/re-used in up to 4 changes.

2.5.1 Administrative User(s)

The main application administrative account has master privileges for administering any and all user/subuser accounts within the system. This user is often referred to as MADMIN.

EXAMPLE:

1. Upon a fresh install, the system administrator will log into the Monetra server using username=MADMIN and password=password.
2. Once connected, the Administrator should change the MADMIN password to something strong(see note above).
3. At this point the first system user (merchant profile) can be added. Lets say we add an account for Jane that connects to vital and we call this user 'JaneVital' and give her a password of 'test-123'.

2.5.2 System User(s)

The Monetra Administrative account has the power to add a user account. (i.e. This represents any single MID/TID/TERMINAL combination routing to any number of processors, on a per industry profile). Each user account is responsible for managing its own subusers.

EXAMPLE:

1. The master user account for Jane was created by MADMIN and is called 'JaneVital'.
2. JaneVital has the administrative role for the JaneVital account, and all subusers.

2.5.3 System SubUser(s)

These are administered via the master user account to create a non-privileged sub user account.

EXAMPLE:

1. JaneVital logs in and creates a sub-user called 'Jim' that only has access to the 'Sale' function for the JaneVital master account. Jim gets a password of 'test-9876'

2. When Jim logs into monetra with [username=JaneVital:Jim password=test-9876] he can only run a sale transaction.

2.6 Hardware Security Modules (HSM)

Hardware Security Module(s) (HSM) are a type of secure cryptoprocessor targeted at managing digital keys, accelerating cryptoprocesses in terms of digital signings/second and for providing strong authentication to access critical keys for sensitive software applications. Many Government agencies around the world use HSM's to secure their most valuable assets and all major HSM providers have taken their products through both a FIPS 140 [http://en.wikipedia.org/wiki/FIPS_140] and a Common Criteria [http://en.wikipedia.org/wiki/Common_Criteria] evaluation to assure the security levels of their products.

Why use an HSM?

Pro's	Con's
1. Required for Point to Point encryption systems that want to reduce the scope of PCI. 2. Provides hardened physical security for cryptographic material (keys etc.). 3. Provides tools and procedures to ease the management of cryptographic material.	1. Hardware based offering that can be prohibitively expensive for smaller deployments. 2. Can be technologically challenging to install and configure.

See the PCI website [https://www.pcisecuritystandards.org/] for more details regarding PCI and P2P systems.

2.6.1 HSM Monetra Architecture

Monetra's support for HSM devices was designed to be transparent for Monetra administrators. For example there are no new HSM specific API calls or procedures. A generate Key request to Monetra will happen via the HSM if it is configured for use. There are however several different ways you can deploy and use an HSM with Monetra.

1. **HSM Full:** This mechanism is used to pass all encryption and decryption operations for the Monetra database through the HSM. Due to the number of cryptographic operations required for things like large reports and settlements, there may be a performance penalty for its use.

2. **HSM Key Load:** This mechanism is used to protect the database encryption key using an HSM-protected key as its key-encrypting-key. Upon startup, Monetra will request the HSM decrypt the database key which will remain in volatile memory until Monetra is closed. This is our recommended mode of operation to maintain high levels of performance. .

3. **CardShield:** All key material and cryptographic operations (related to CardShield P2P system) happen within the HSM.

Note: 1. CardShield mode can be configured in conjunction with either 'HSM Full' OR 'HSM Key Load' to also protect data stored within Monetra's database.

2. Separate HSM devices can be used for each mode (i.e. CardShield crypto can be on one HSM device while the DataBase crypto can reside on another).

3. Initial support for HSM devices is limited to Monetra builds that execute on Linux and/or Microsoft Windows based platforms. If you require the use of an HSM from an alternate platform (such as Solaris or IBM AIX) then please contact sales@monetra.com

4. PCI requires HSMs be configured in FIPS-140-2 Level 3 mode to comply with P2PE requirements

2.6.2 Supported HSM Devices

To take advantage of HSM support, you must be running Monetra version 7 update 9.0 (or higher), or 7 update 10.0 for SafeNet ProtectServer support.

Fully Supported:

- Thales nShield Connect (network attached) and nShield Solo (PCI Card). See Thales website [http://www.thales-esecurity.com/Products/Hardware%20Security%20Modules.aspx]

- SafeNet ProtectServer and ProtectServer External (network attached). See SafeNet website [http://www.safenet-inc.com/products/data-protection/hardware-security-modules-hsms/]

Note: SafeNet uses PKCS11 which requires a passphrase (aka PIN) be entered to authenticate with the HSM in order to log into a session to use the keys it protects. The Monetra Manager provides prompting when necessary, but when using headless/gui-less environments, you must use the provided monetra_scc utility to authenticate with the HSM before Monetra is ready to use

Note: 1. To benefit from PCI scope reduction the HSM must be configured to operate under FIPS 140-2 level 3

2.6.3 Monetra HSM Configuration

As with most other aspects of Monetra, HSM support can be configured via a command-line/text editor or via the Monetra Manager utility which provides a simple user interface.

We have included instructional procedures below that will show how to configure Monetra for HSM support using both the Monetra Manager and the classic command-line methods.

The procedure described below in general terms describes how to configure Monetra to use an HSM device. The settings are configured to offload both Monetra's CardShield sub-system and 'Key Load' mode for DB encryption key protection.

Before you start this procedure, ensure you have a Thales nShield HSM device or SafeNet ProtectServer device (as outlined in the Supported Devices section) that has been initially configured and is ready to go.

1. **Initialize:** Initialize the HSM with its security world or a User Token.

2. **Bump to FIPS 140-2 level 3 mode:** Ensure the device is running in Level 3 mode.

3. **Load HSM drivers:** Ensure you load the HSM drivers as provided by the manufacturer on the local machine the Monetra Server executes from.

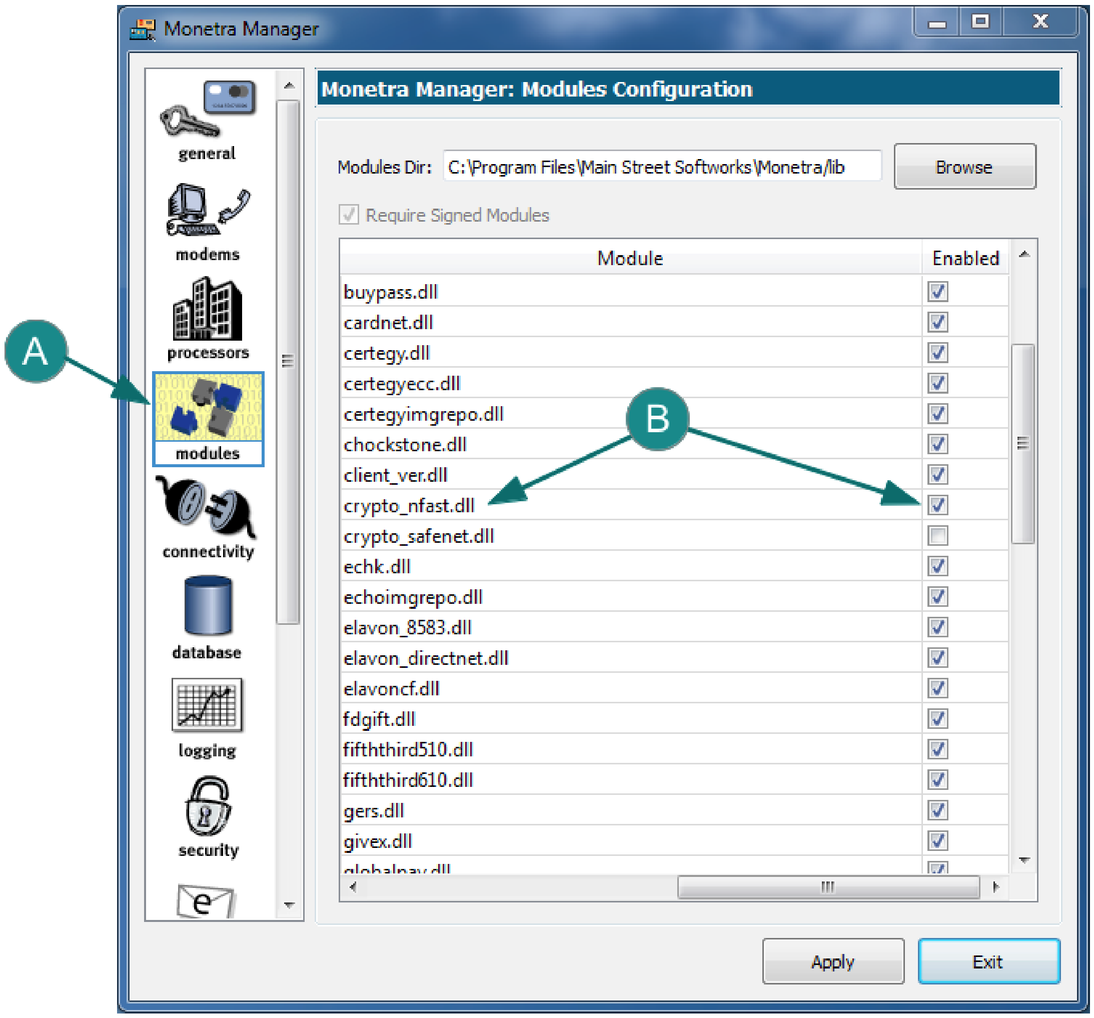
4. **Thales only: Add Monetra to nFast group:** Ensure you add Monetra to the nFast group so it is allowed to use the HSM(permissions).

5. **SafeNet only: Load the CardShield Functionality Module:** Import the Monetra CardShield Module's signing certificate into the HSM then load the functionality module.

2.6.3.1 Configure via GUI Manager

Configuring Monetra to support an HSM device

Figure 2.2. Load HSM Modules



Monetra Manager: Modules Configuration

Modules Dir: C:\Program Files\Main Street Softworks\Monetra/lib

[✓] Require Signed Modules

Module	Enabled
buypass.dll	✓
cardnet.dll	✓
certegy.dll	✓
certegyecc.dll	✓
certegyimgrepo.dll	✓
chockstone.dll	✓
client_ver.dll	✓
crypto_nfast.dll	✓
crypto_safenet.dll	
echk.dll	✓
echoimgrepo.dll	✓
elavon_8583.dll	✓
elavon_directnet.dll	✓
elavoncf.dll	✓
fdgift.dll	✓
fifththird510.dll	✓
fifththird610.dll	✓
gers.dll	✓
givex.dll	✓

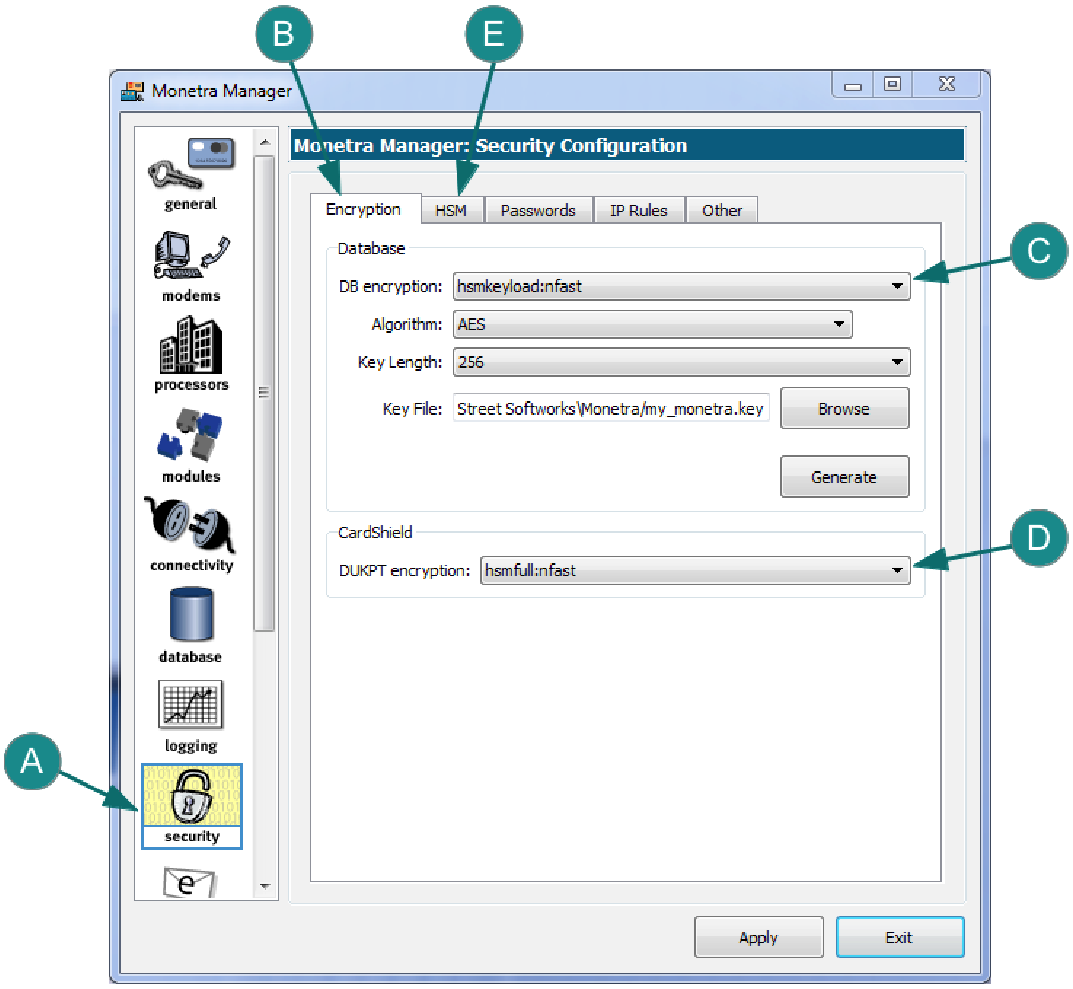
1. Choose the 'modules' section [A] in the left navigation pane of the Manager utility.

2. Find the HSM module for your HSM. For Thales, the module is crypto_nfast, for SafeNet, the module is crypto_pkcs11 [B] and place a check-mark in the 'Enabled' check-box.

3. Click on the 'Apply' button at the bottom and restart Monetra when prompted.

Configuring Monetra to support an HSM device (continued)

Figure 2.3. Set Encryption Options



Monetra Manager: Security Configuration

Database

DB encryption: hsmkeyload:nfast

Algorithm: AES

Key Length: 256

Key File: Street Softworks\Monetra/my_monetra.key

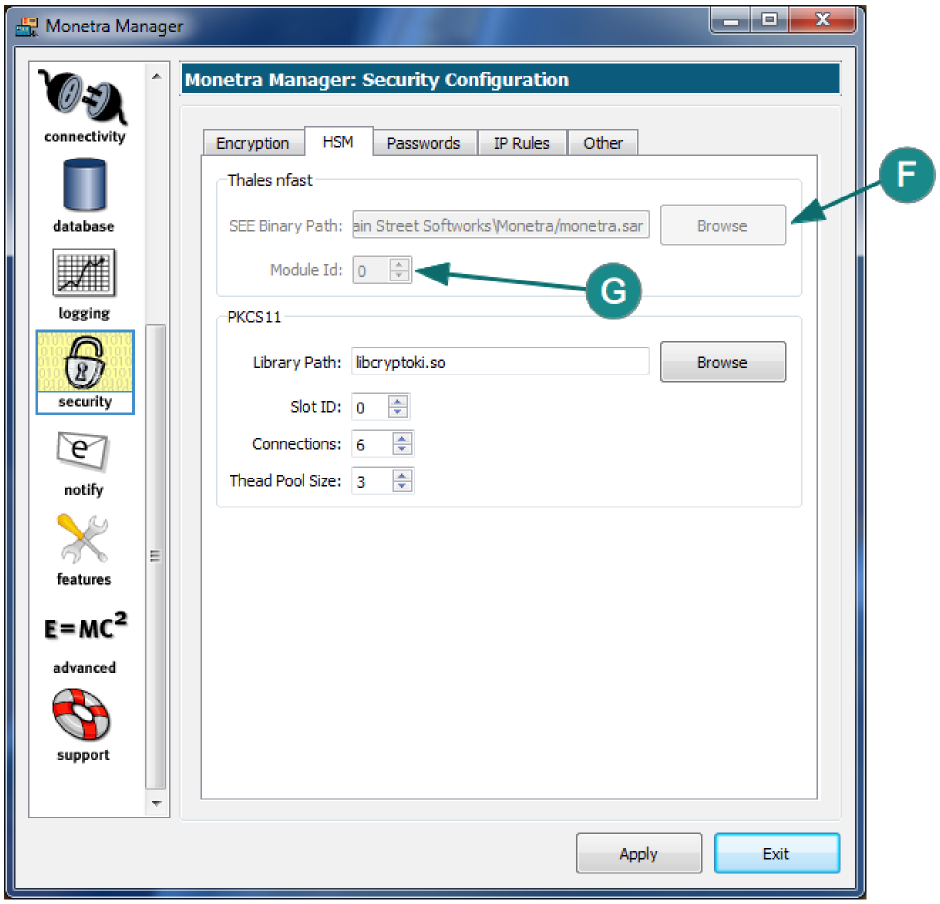
CardShield

DUKPT encryption: hsmfull:nfast

1. Choose the 'security' section [A] in the left navigation pane of the Manager utility.
2. Select the Encryption [B] tab at the top.
3. Choose 'hsmkeyload' from the DB Encryption drop-down [C]. Note: Upon initial set it will pop up another dialog asking for the device type (choose 'nfast' or 'pkcs11' depending on your HSM).
4. Choose 'hsmfull' from the CardShield DUKPT Encryption drop-down [D]. Note: Upon initial set it will pop up another dialog asking for the device type (choose 'nfast' or 'pkcs11' depending on your HSM).
5. Select the HSM [E] tab at the top.

Configuring Monetra to support a HSM device (continued)

Figure 2.4. Set HSM Options



Monetra Manager: Security Configuration

Thales nfast

SEE Binary Path: ain Street Softworks\Monetra/monetra.sar

Module Id: 0

PKCS11

Library Path: libcryptoki.so

Slot ID: 0

Connections: 6

Thead Pool Size: 3

1. Configure a Thales HSM
   a. Verify the location of the Monetra SEE Binary path [F].
   b. Verify the Module ID# [G]. 0 = automatically assign the first available HSM
2. Configure a SafeNet HSM
   a. Configure the appropriate pkcs11 slot ID. The first user token is usually 0
   b. Configure the number of connections to use, this is how many simultaneous operations can be performed on the HSM. Recommended: 6
   c. Configure the number of threads to use. This is the number of helper threads to spawn in order to parallelize usually non-parallel cryptographic tasks such as bulk encryption and decryptions for reports. This number must be less than or equal to the number of connections.
   d. Configure the location of the pkcs11 driver provided by the HSM vendor
3. Important: Click on the 'Apply' button at the bottom and DO NOT restart Monetra if prompted.
4. Continue on to key generation, using the Monetra Manager utility, as per Section 2.8.2.

2.6.3.2 Configure via terminal/console

Configuring an HSM device by hand

HSM settings are configured like all other aspects of the system such as modems and log-files. Depending on the Operating System, your configuration files (*.conf) may reside in a location other than the ones shown in the example below.

1. Open /etc/monetra/main.conf with a text editor, set:

```
dbencrypt=hsmkeyload:nfast   # or hsmkeyload:pkcs11
enckeyfile=/usr/local/monetra/my_monetra.key   # DB encryption key
dukptencrypt=hsmfull:nfast   # or hsmfull:pkcs11
```

2. Open /etc/monetra/modules.conf with a text editor, and un-comment:

```
loadmodule=crypto_nfast.so
```

or

```
loadmodule=crypto_pkcs11.so
```

3. Open /etc/monetra/prefs.conf with a text editor, and set:

```
nfast_moduleid=0
nfast_seebinary_path=/usr/local/monetra/monetra.sar
```

or

```
pkcs11_slotid=0
pkcs11_connections=6
pkcs11_threadpool_size=3
pkcs11_library_path=libcryptoki.so
```

4. Generate the encryption key as per the next section.

2.7 Network Segmentation via Multi-PORT

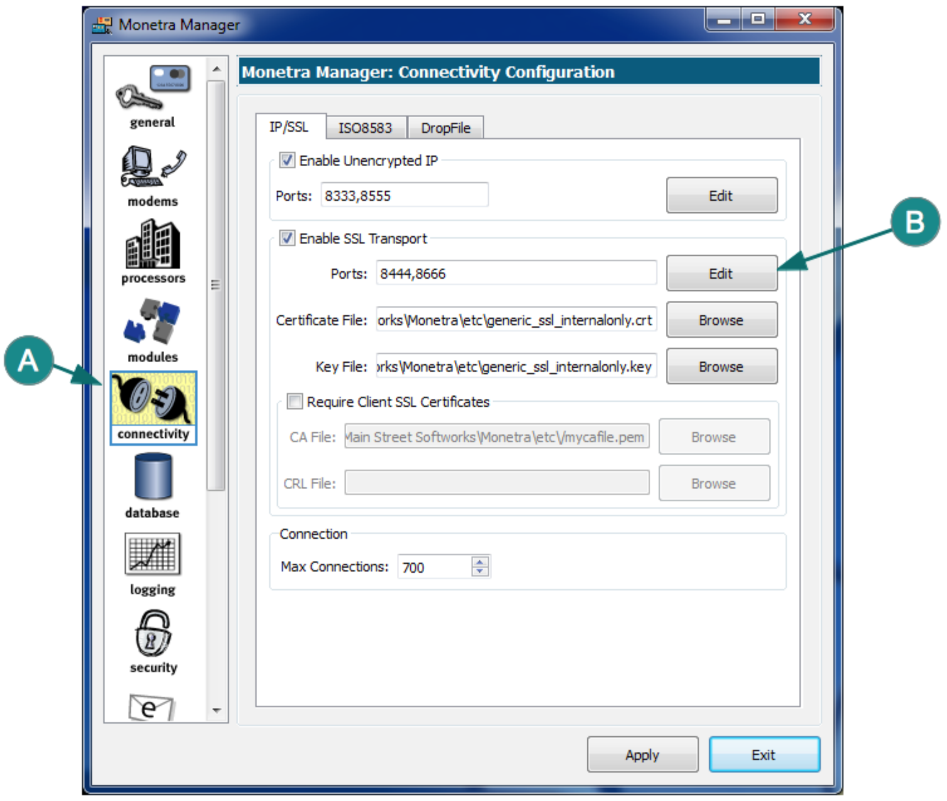
As of Monetra version 7 update 11.0, Monetra has been enhanced to support multi-PORT features.

Each configured PORT has a variety of options that can be set, including PROTOCOLS (such as monetra, monetra-xml, VISA2 etc), raw communication methods (Stream, HTTP etc.) and permissions (ADMIN, USER etc.)

In the example below, we will restrict all access coming in on port 8666 to standard users with the OBSCURED permission set.

Restrict Port

Figure 2.5. Port Management



Monetra Manager: Connectivity Configuration

[✓] Enable Unencrypted IP
Ports: 8333,8555

[✓] Enable SSL Transport
Ports: 8444,8666
Certificate File: orks\Monetra\etc\generic_ssl_internalonly.crt
Key File: rks\Monetra\etc\generic_ssl_internalonly.key

[ ] Require Client SSL Certificates
CA File: Main Street Softworks\Monetra\etc\/mycafile.pem
CRL File:

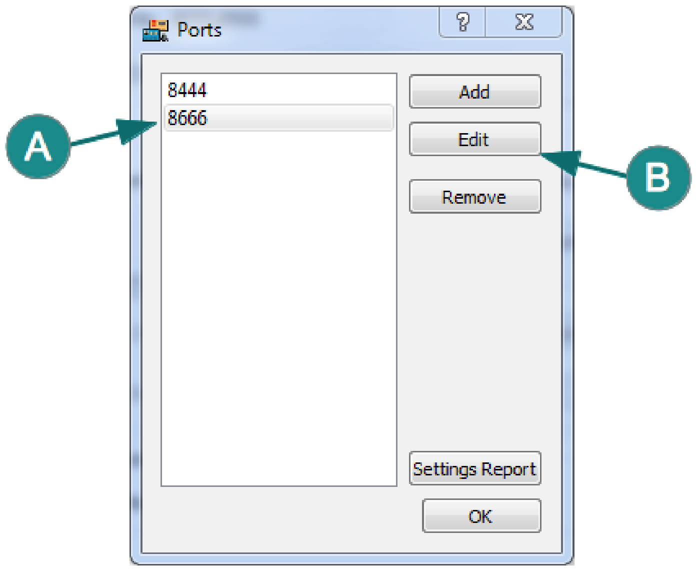
Connection
Max Connections: 700

1. Choose the 'connectivity' section [A] in the left navigation pane of the Manager utility.

2. Select the 'Edit' [B] button from within the SSL Transport section.

Restrict Port (continued)

Figure 2.6. Ports Selection

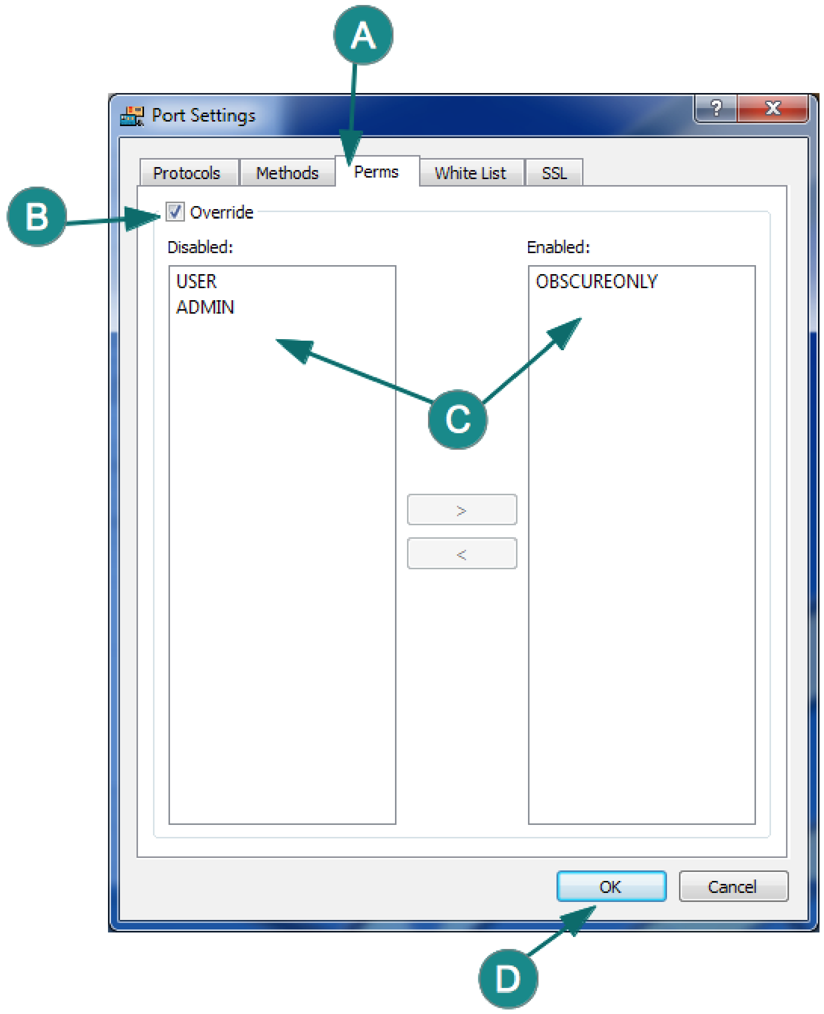


1. Highlight the Port you would like to configure [A].

2. Select the 'Edit' [B] button.

Restrict Port (continued)

Figure 2.7. Permissions Settings



1. Select the 'Perms' tab [A].

2. Enable 'Override' [B] check-box.

3. Enable only the permissions for the port you want by moving them from 'Enabled' to 'Disabled' window [C].

4. When all changes are complete, choose the 'OK' [D] button.

2.8 Encryption Keys

As required by PA-DSS Monetra provides facilities for strong application level encryption. One of the most important aspects of this system is the creation, distribution and maintenance of the encryption keying infrastructure.

While the following reference outlines the basic steps for key administration, we recommend you refer to the most current Monetra Configuration Guide.

2.8.1 Encryption Key Creation from a console

Depending on the base operating platform from which Monetra is deployed, the steps may vary. Please reference the most recent Installation and Configuration guides for more details on how to use Monetra key generation tools.

On the Linux operating system, an encryption key might be created/generated for use with Monetra as follows.

To generate a key, use the following shell command:

monetra_keygen

EXAMPLE:

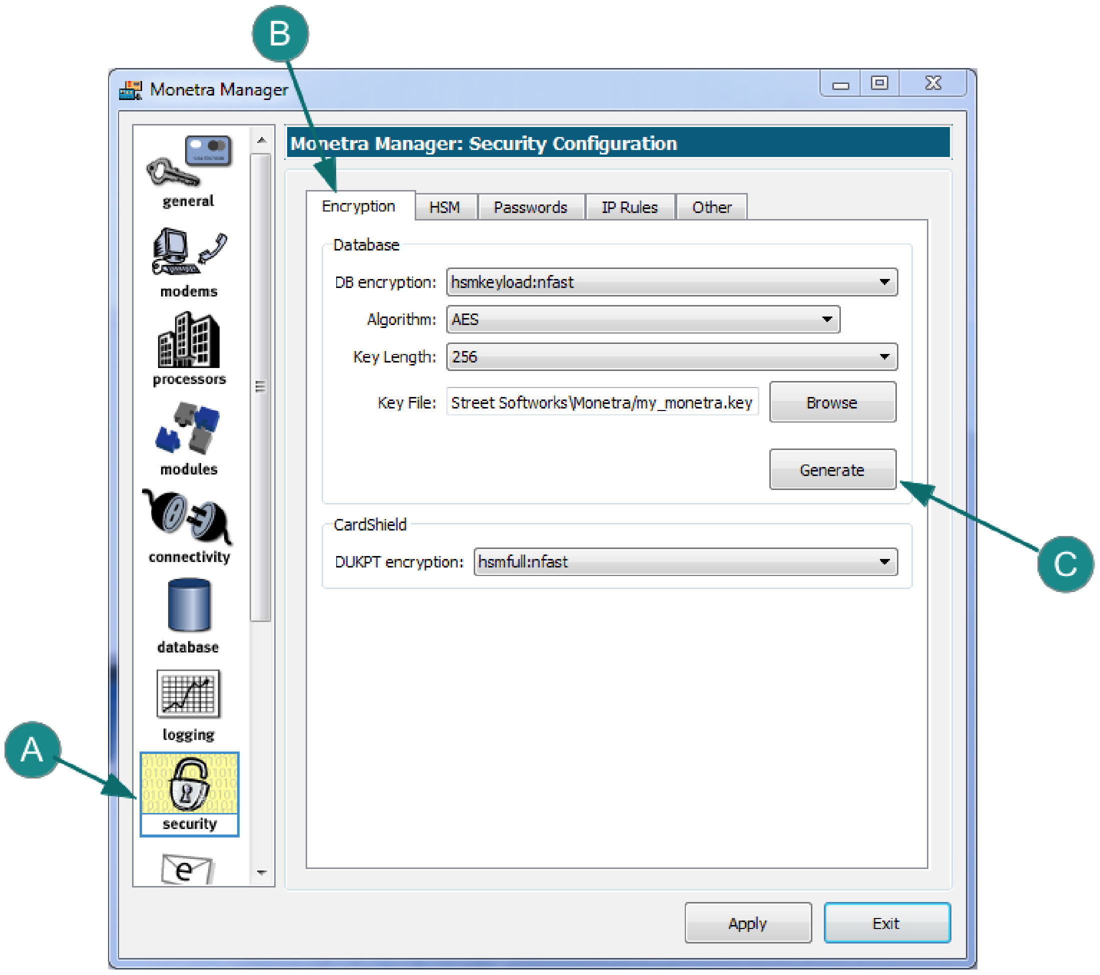
```
$ /usr/local/monetra/bin/monetra_keygen
```

Note: On Microsoft Windows or Mac OS X operating systems, it is recommended to use the Monetra Manager utility to build all encryption keys (see image below).

2.8.2 Encryption Key Creation from the Monetra Manager

Generate Encryption Key

Figure 2.8. Generate Encryption Key



Monetra Manager: Security Configuration

Database

DB encryption: hsmkeyload:nfast

Algorithm: AES

Key Length: 256

Key File: Street Softworks\Monetra/my_monetra.key

CardShield

DUKPT encryption: hsmfull:nfast

1. Choose the 'security' section [A] in the left navigation pane of the Manager utility.
2. Select the Encryption [B] tab at the top.
3. Click the 'Generate' [C] button and follow the prompts.
4. Click on the 'Apply' button at the bottom and restart Monetra when prompted.

2.8.3 Encryption Key Creation (with pass phrase)

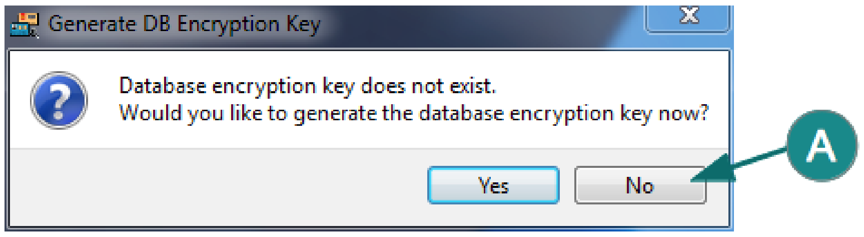
If not using an HSM, you can configure the database encryption key to be protected by a series of passphrases that must be entered on generation and startup

On the Windows operating system, an encryption key might be created/generated for use with Monetra as follows.

Note: In the example provided below, the side channel features are used to manage the pass phrases.

Configuring Monetra Encryption Key with Pass Phrase

Figure 2.9. Run Monetra Manager Application


Generate DB Encryption Key

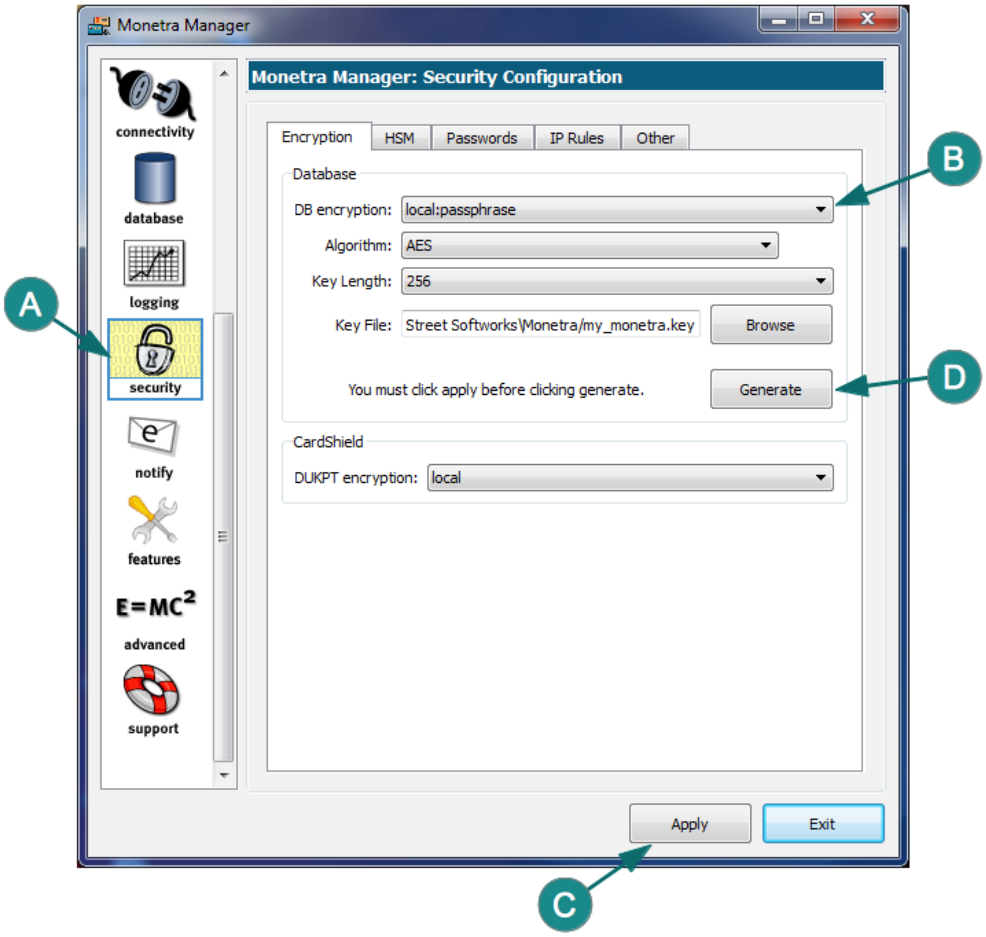
Database encryption key does not exist.
Would you like to generate the database encryption key now?

1. When you first run the Monetra Manager utility, if a key does not exist you will be prompted to create one.

2. If you are configuring the system to use a pass phrase then choose 'No' [A]

Configuring with Pass Phrase (continued)

Figure 2.10. Configure DB Encryption Method



Monetra Manager: Security Configuration

Database

DB encryption: local:passphrase

Algorithm: AES

Key Length: 256

Key File: Street Softworks\Monetra/my_monetra.key

You must click apply before clicking generate.

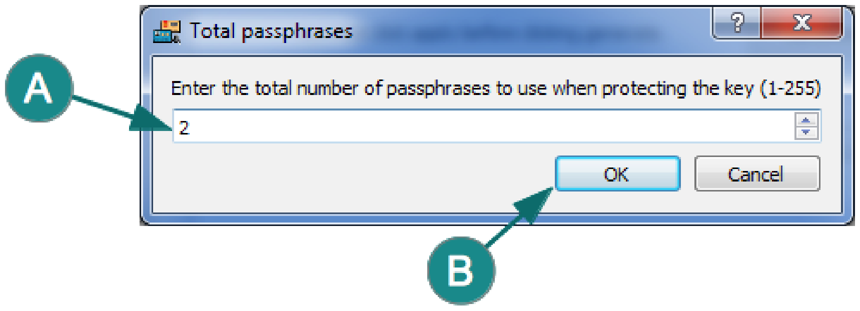
CardShield

DUKPT encryption: local

1. Choose the 'Security' section [A] in the left navigation pane of the Manager utility
2. Set the DB encryption type [B] to "local:passphrase".
3. Click the 'Apply' [C] button.
4. Click the 'Generate' [D] button.

Configuring with Pass Phrase (continued)

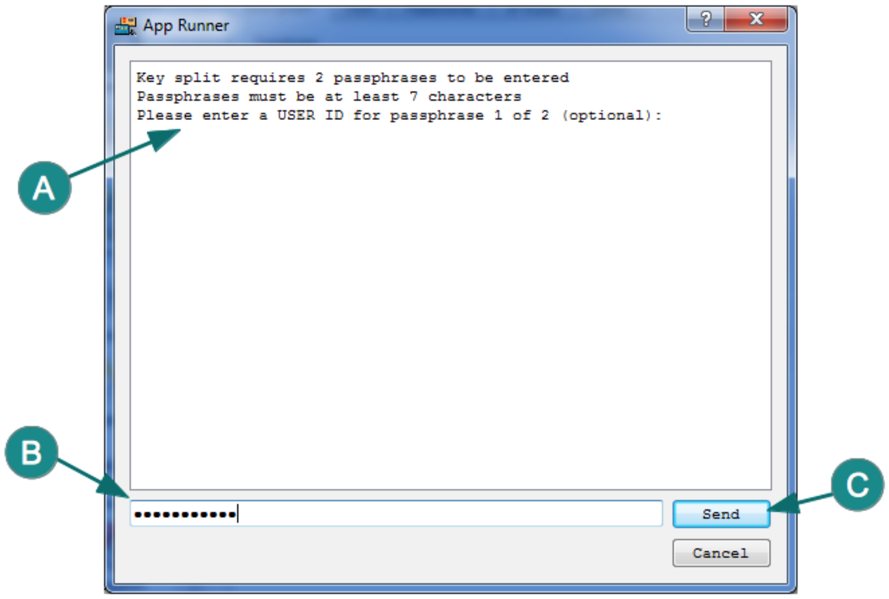
Figure 2.11. Configure Number of Pass Phrases


Total passphrases
Enter the total number of passphrases to use when protecting the key (1-255)
2
OK
Cancel
A
B

1. Enter the number of phrases to use [A].
2. Click the 'OK' [B] button.

Configuring with Pass Phrase (continued)

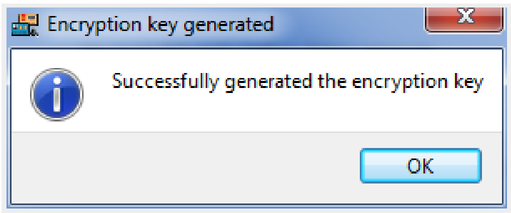
Figure 2.12. Applying the Pass Phrases



```
Key split requires 2 passphrases to be entered
Passphrases must be at least 7 characters
Please enter a USER ID for passphrase 1 of 2 (optional):
```

- At this point you will be using the side-channel utility to enter your phrases. Pay attention to the prompts [A] and enter all required information into the text-box [B] and then Click on the 'Send' [C] button.

Note:



Successfully generated the encryption key

Upon successful Key Generation, you should see a similar dialog to the one above.

When starting Monetra, the required number of passphrases must be entered before Monetra is ready to use. This can be entered via the Monetra Manager which will automatically detect when passphrase entry is required. If using a headless/gui-less environment, we provide a tool called monetra_scc which can be used for passphrase entry.

2.8.4 Encryption Key Maintenance

As with most security systems and policies, it is required to change out encryption keys from time to time or as often as needed. Please review the most recent Installation and Configuration guides for more details.

Below is an example of the steps required to replace an encryption key on the Linux Operating system:

1. Settle all transactions.
2. Export your current Monetra data-file encrypted.
3. Stop Monetra.
4. Securely remove the contents of your data directory.
5. Build a new Monetra encryption key.
6. Re-start Monetra.
7. Import your data.
8. Validate the import and proper Monetra operation, then securely remove any unused cryptographic materials.

Note: When you upgrade or re-install any Monetra engine via the Monetra Installer Utility and it performs an export/import, the encryption key will be replaced as part of the process.

2.9 Client Certificates (SSL)

In order to address issues of secure client verification, Monetra has the ability to only allow secure SSL and XML_HTTPS connections from authorized clients. By utilizing a Certificate Authority and client-side SSL certificates, only clients who present an SSL certificate signed by a CA that the administrator has configured Monetra to recognize, are allowed to connect. This document describes the actions required to set up a simple Certificate Authority using

OpenSSL and the steps required to integrate the Certificate Authority's signing certificate and client-side SSL certificate restrictions into Monetra.

Due to the large number of possible configurations and implementations, this guide only describes the minimal steps required to enable client-side certificates. It is up to the administrator to ensure that file permissions, locations, security, etc. are implemented and controlled according to local security guidelines.

2.9.1 Certificate Authority Setup and Use

Create the CA base directory:

```
$ mkdir ~/myCA
```

Copy CA.pl from your OpenSSL distribution into the CA base directory:

```
$ cd ~/myCA
$ cp .../CA.pl .
```

Edit the CA.pl script, ensuring the basic settings are correct:

- verify number of days certificate is valid ($DAYS)
- set CA base directory ($CATOP) relative to the location of CA.pl
- fix dirmode as appropriate ($DIRMODE), 0777 is *not* safe

Edit the system openssl.cnf and change 'dir' to match $CATOP

Create the CA layout. Enter a CA private key pass phrase when prompted and fill in the relevant information for your use:

```
$ ./CA.pl -newca
A certificate filename (or enter to create)

[press ENTER here]

Making CA certificate ...

Generating a 1024 bit RSA private key

...............++++++

..++++++

writing new private key to 'CA/private/cakey.pem'

Enter PEM pass phrase:

Verifying - Enter PEM pass phrase:

-----
```

You are about to be asked to enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields, but you can leave some blank. For some fields there will be a default value and if you enter '.', the field will be left blank.

```
Country Name (2 letter code) [AU]:US

State or Province Name (full name) [Some-State]:Florida

Locality Name (ie. city) []:Gainesville

Organization Name (eg, company) [Internet Widgits Pty Ltd]:Main Street Softworks, Inc.

Organizational Unit Name (ie. section) []:

Common Name (ie. YOUR name) []:

Email Address []:support@monetra.com
```

Fix permissions on the generated files as required (files are generated using the user's umask):

```
$ chmod -R o-rx .
```

Verify that the Certificate Authority layout includes both the Certificate Authority Cert and private key:

```
$ find . | sort
.

./CA

./CA.pl

./CA/cacert.pem

./CA/certs

./CA/crl

./CA/index.txt

./CA/newcerts

./CA/private

./CA/private/cakey.pem

./CA/serial
```

cacert.pem should look like this:

```
-----BEGIN CERTIFICATE-----

MIIDYzCCAsygAwIBAgIJAKcV7BfxXP68MA0GCSqGSIb3DQEBBAUAMH8xCzAJBgNVBAYTAlVTMRAwDgYDVQQIEwd
GbG9yaWRhMRQwEgYDVQQHEwtHYWluZXN2aWxsZTEkMCIGA1UEChMbTWFpbiBTdHJlZXQgU29mdHdvcmtzLCBJbm
MuMSIwIAYJKoZIhvcNAQkBFhNzdXBwb3J0QG1vbmV0cmEuY29tMB4XDTA1MDYwOTEzNTMyMVoXDTA2MDYwOTEzN
TMyMVowfzELMAkGA1UEBhMCVVMxEDAOBgNVBAgTB0Zsb3JpZGExFDASBgNVBAcTC0dhaW5lc3ZpbGxlMSQwIgYD
VQQKExtNYWluIFN0cmVldCBTb2Z0d29ya3MsEluYy4xIjAgBgkqhkiG9w0BCQEWE3N1cHBvcnRAbW9uZXRyYS5j
b20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAL1xNVuiUpYu4lRxWVdcAv/Fd1JVujLhHAgMo1BA1w8br2Q
J7++eK7BXEU3mCN8jmuUyn2R/nE2U87X+RdY0KH9FQ7Abfj1b2vOYcmVHBvgm8FGBeueew8900M3xi/Gwz8AINO
```

```
FCEg/+/0TuLkCUDg4RFuJDOyZQaGHzMSeGreZrAgMBAAGjgeYwgeMwHQYDVR0OBBYEFCGABWx3AnXYZysdlhtL/
5qBZxDbMIGzKIjblieJRLILJDlivliEJflIDlfildIFJLIdfhqBZxDboYGEpIGBMH8xCzAJBgNVBAYTAlVTMRAw
DgYDVQQIEwdGbG9yaWRhMRQwEgYDVQQHEwtHYWluZXN2aWxsZTEkMCIGA1UEChMbTWFpbiBTdHJlZXQgU29mdHd
vcmtzLCBJbmMuMSIwIAYJKoZIhvcNAQkBFhNzdXBwb3J0QG1vbmV0cmEuY29tggkApxXsF/Fc/rwwDAYDVR0TBA
UwAwEB/zANBgkqhkiG9w0BAQQFAAOBgQBVwrhdYFnkoISAwivmKbs6Mf7OgmvTm5Cg+hjG7VBwPbJf991tsxqir
bEd9W3tqGJ58RKJchnstrcihpfdMptFfKDaNDawG6xWiHUYvmwzkaAI+ciXBm/DpvDZvm9A09SCOHz+aJNILazy
hz9q38MAtK04vwT7o5Cac/LCh5Yr6w==

-----END CERTIFICATE-----
```

and private/cakey.pem should look like this:

```
-----BEGIN RSA PRIVATE KEY-----

Proc-Type: 4,ENCRYPTED

DEK-Info: DES-EDE3-CBC,2505C24E04098DEE

z1eWv6x/NNmVibFRUvi3G0bbLwv+RfbEPLAqNDzeHNS4Zm5ksTwkxaNNfg+X5wVjhchXuq4T0yt5USWk3fCU2Dx
+2vGm4j/rkEQJDUGGnRihoi5ZyrHi68fJOrO4G1xvCf37IMSb65LGcNKWtaEFo2YV5SnJfYtJU1vAUh4LV4Zt5Z
T6CWGTFcxVzmtEWaCjpDEvKliclbjLDIFj4idfldDO1xAoAaesVsD1V+KsxrSdqwek5T1TzB9478TtbodfRRzRl
sE6bFVNDkdQFv3im/ZgM4rFhAHihRffQy7mW6XGkg19T1JbcZfIHXGUrzOi3iOb6mBJVANZcjZdwnbOsrbWcUPB
EXgHcQUrsce7qhzx5LaEQUowEJHAhERQ5GxGRAmeHz8MeyJklolnH5chVkmEAKBRrWMi4EsPHdfgXVyEG9uJ2St
bHDuh7N6fkmCGXwrblbkpf1M6/j/i31e58FVOB616Oufa1EaSDzeaCQzm/uWIIiCz0RsxXY/hoGOZ9miyQjjZru
0OiVFf8QNsF2MpD2uVraWSyYTcGgF0x8hKXvlcnsi3aNlzzMT+iGxOOEWdIbFsIfD9YqQ35h9BAkn7weJQfDAlD
2K6noktgBmfcYJlUqGDt46/a990nKek6jQH02rPngZMRNaO/97VgNbgiZ1zkBq2p83FY397IhuSVRLqq5wW7S4d
XNyu7J3+tIqN5LiUdePGqyix7ltOovFqmV5cWnKVquK02Vc/Y8KNZNFBREC4WT2m8663DThC2ocQrmleHcLP7Yo
GCWp4EAmBpxHrU0FQYCTX7tIXO5KKLRPjenrPQ==

-----END RSA PRIVATE KEY-----
```

2.9.2 Restricting SSL Connections with Certificates

Copy your CA's certificate (cacert.pem) to a location that Monetra can access:

```
$ cp CA/cacert.pem /etc/monetra/mycafile.pem
```

Edit prefs.conf, set:

```
COMM_ssl=yes
COMM_ssl_cert_required=yes
COMM_ssl_cafile=/etc/monetra/mycafile.pem
```

and restart Monetra.

Note: You will need to ensure you have a server-side SSL cert/key to allow SSL connections. This may be a certificate generated with the newly-created CA or an existing cert/key combination.

You can verify that these settings are being used by checking monetra.log and locating these lines:

```
COMM_ssl_cafile: /etc/monetra/mycafile.pem
COMM_ssl_cert_required: yes
```

At this point, only clients with valid SSL certificates signed by this certificate authority key are granted further access.

To create a signed client certificate, you must first create a certificate request. Using OpenSSL:

```
$ openssl req -new -nodes -out newreq.pem

Generating a 1024 bit RSA private key

........................................++++++

.........++++++

writing new private key to 'privkey.pem'

-----

You are about to be asked to enter information that will be incorporated
into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

-----

Country Name (2 letter code) [AU]:US

State or Province Name (full name) [Some-State]:Florida

Locality Name (eg, city) []:Gainesville

Organization Name (eg, company) [Internet Widgits Pty Ltd]:Main Street Softworks, Inc.

Organizational Unit Name (eg, section) []:Testing

Common Name (eg, YOUR name) []:testbox.monetra.com

Email Address []:support@monetra.com

Please enter the following 'extra' attributes to be sent with your certificate request:

A challenge password []:

An optional company name []:
```

At this point, the Certificate Authority must sign the certificate:

```
$ ./CA.pl -sign
```

Using configuration from /etc/ssl/openssl.cnf

Enter pass phrase for ./CA/private/cakey.pem:

```
DEBUG[load_index]: unique_subject = "yes"
```

Check that the request matches the signature

```
Signature ok
```

```
Certificate Details:
        Serial Number:
            a7:15:ec:17:f1:5c:fe:bf
        Validity
            Not Before: Jun 10 15:37:32 2005 GMT
            Not After : Jun 10 15:37:32 2006 GMT
  Subject:
   countryName    = US
   stateOrProvinceName  = Florida
      localityName   = Gainesville
      organizationName  = Main Street Softworks, Inc.
      organizationalUnitName = Testing
      commonName    = testbox.monetra.com
      emailAddress   = support@monetra.com

  X509v3 extensions:
      X509v3 Basic Constraints:
          CA:FALSE
      Netscape Comment:
       OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                AB:72:09:4E:22:65:8E:6F:79:CA:9A:AD:3E:C4:20:05:4C:8E:99:B0
            X509v3 Authority Key Identifier:
                keyid:21:80:05:6C:77:02:75:D8:67:2B:1D:96:1B:4B:FF:9A:81:67:10:DB
                DirName:/C=US/ST=Florida/L=Gainesville/O=Main Street Softworks,
Inc./emailAddress=support@monetra.com
                serial:A7:15:EC:17:F1:5C:FE:BC

  Certificate is to be certified until Jun 10 15:37:32 2006 GMT (365 days)

  Sign the certificate? [y/n]:y

  1 out of 1 certificate requests certified, commit? [y/n]y

  Write out database with 1 new entry

  Data Base Updated

  Signed certificate is in newcert.pem
```

The signed client certificate is now located in newcert.pem, and the private key is in privkey.pem.

On the client side, applications developed using the libmonetra API can use the M_SetSSL_CAfile() and M_SetSSL_Files() functions to load the CA's certificate along with the client private key and signed client certificate into the client application after calling M_SetSSL() but before calling M_Connect().

2.9.3 Restricting User Access with Certificates

The administrator can add per-user restrictions on which client certificate(s) are allowed to execute transactions. Monetra uses a cryptographically-secure digest (hash) of the client certificate to identify individual client certificates.

The client certificate digest can be generated using the M_SSLCert_gen_hash(), giving the filename as the first argument. Alternatively, the digest can be generated using the X509_digest() digest in OpenSSL with the SHA1 digest algorithm or can be generated using the openssl(1) application:

```
$ openssl x509 -sha1 -in newcert.pem -noout -fingerprint

fingerprint: 96:f8:ac:6b:76:8b:d5:f3:5f:bb:2d:0c:4e:9d:19:c4:b4:49:ad:36
```

To add a client certificate restriction to Monetra, a transaction such as this would be used:

```
username=madmin
password=password
action=admin
admin=restriction
restriction=add
restriction_user=loopr
restriction_type=ssl_cert
restriction_data=96:f8:ac:6b:76:8b:d5:f3:5f:bb:2d:0c:4e:9d:19:c4:b4:49:ad:36
```

To retrieve the client restrictions for a particular user, a transaction such as this would be used:

```
username=madmin
password=password
action=admin
admin=restriction
restriction=list
restriction_user=loopr
```

To remove a client restriction:

```
username=madmin
password=password
action=admin
admin=restriction
restriction=remove
restriction_num=1
```

2.10 Alternate File System Security

Since the earliest days of Unix, DOS and most versions of Windows, payment applications have relied heavily on an inter-application communication via textual based files that are written and then read from a computer's hard disk. While simple and easy to implement, these legacy integrations (drop-files) have higher security implications.

The first risk identified is the ability of an attacker to gain root control over a machine and scan the shared directory for incoming and outgoing files. This risk must be mitigated via operating system and file system security measures.

The second risk identified is when a physical disk is examined outside of the operating system. Examples would be removing a hard drive and performing a forensic analysis, or booting the computer with a CDROM that contains a base OS and forensic tools. In theory an attacker or even a technical ebay shopper could peruse old files that were intentionally deleted on the physical disk.

The third risk identified is when the Operating System uses a journaled file system thus creating an unwanted phenomenon called 'data remanence'. WARNING: Due to the numerous journaled file systems in production today, and the lack of 'Secure Delete' tools for all types, if you must operate Monetra's output logs in a non-pci mode then we recommend the use of an alternate RAM-Disk, exclusively for those trouble-shooting scenarios.

Since payment applications can communicate sensitive data such as card numbers and CVV2 values, it becomes imperative to look at if, how and why any 'disk based' communication happen. If disk based files are written and contain sensitive data, then we highly recommend you look at alternate security measures that help improve the security posture of your disk based communication systems.

Some of the more modern alternatives would be to deploy an additional layer of security around your /trans directory such as an encrypted file system or implementing a temporary/ memory-resident file system.

2.10.1 Encrypted File Systems

The advantage of an encrypted file system is that when a file is written to disk, no matter the file system or operating system used, it should be considered safe form of protection since the process does not depend on the integrity of the operating system after the encryption takes place.

Note: If you use disk encryption, PCI-DSS requirement 3.4.1 will apply to your configuration.

For more information on encrypted file systems on linux, please visit.http:// www.linuxjournal.com/article/6481

For more information on encrypted file systems for Microsoft Windows, please visit.http:// technet.microsoft.com/en-us/library/bb457065.aspx

2.10.1.1 Temporary File Systems

The advantages of a temporary file system (or RAM disk) is that while it mimics a physical disk drive, it maintains all the processes in 'volatile' system memory (RAM).

If the computer shuts down, the memory is reset, and thus erased.

Note: A nice side effect of using a temp/RAM disk is noticeable performance improvements across all operating platforms.

For more information on temporary file systems, please visit:http://www.wikipedia.org/wiki/ TMPFS

PA-DSS GUIDANCE: If you are operating Monetra in a production environment, and need to elevate the monetra.log output, with a non PCI compliant setting, then we highly recommend you test and deploy a modern RAM-DISK exclusively for securing the logfile activities (i.e. Storage) while trouble-shooting the production system.

A [SAMPLE] Cryptographic Material Custodian Agreement

[Insert Company Name] requires the use of strong cryptographic systems throughout our business operations and process'. These systems are operated as per the current company policy [insert cryptographic policy reference# etc.].

Under normal operating conditions within our technology infrastructure, there may be times where cryptographic material needs to be transferred/stored outside of the direct sub-system. If cryptographic material is ever exported outside of the secure system, as per policy, it must be assigned to an approved custodian who shall be responsible for safeguarding the integrity of the cryptographic material.

You [Insert employee name] have been assigned the duty of Cryptographic Material Custodian for the following materials.

Note: Material must be tracked from issuance until it has been returned, replaced or has expired.

Material ID	Issued	Expires
________________	____/__/__	____/__/__

Material may be exported in several different technical formats. Indicate the type of material being disclosed to custodian.

[ ] Administrative Integrated Computer Chip (Smart Card) for HSM support.
[ ] Base Cryptographic Key (encrypted) for backup and archive.
[ ] Base Cryptographic Key (plain-text) for backup and archive.

By signing below the parties indicate acceptance of the terms and conditions of this Agreement and understand that the terms and conditions of this Agreement are a binding contract upon the parties.

IN WITNESS WHEREOF, the duly authorized representatives of both parties have caused this Agreement to be executed in duplicate effective this ______ day of ________________ in the Year __________.

Employee	[Insert Company Name]
printed name: _____________________	printed name: _____________________
title: _____________________	title: _____________________
____________________________________ signature	____________________________________ signature

References

PCI Security Standards Council. PA-DSS v3.1: Payment Application Data Security Standard, May 2015.

PCI Security Standards Council. PCI-DSS v3.1: Payment Card Industry Data Security Standard, April 2015.

PCI Security Standards Council. PCI-HSM v2.0: Payment Card Industry Hardware Security Module, May 2012.

Main Street Softworks, Inc. Monetra Insallation Guide v7.6: September 2011.

Main Street Softworks, Inc. Monetra Configuration Guide v5.5: September 2006.