Monetra® Payment Software

Secure Implementation Guide
(Covering PCI-DSS and PA-DSS requirements)

Revision: 2.0
June, 2008

 www.mainstreetsoftworks.com

Table of Contents

1 Payment Systems Security

1.1 Introduction

The landscape of merchants' electronic payment systems has changed radically over the past ten years. What was once mostly proprietary (closed loop/dial) systems, has become a network of computers connecting to multiple partners and services via public networks such as 'the public Internet', GSM, wireless access points etc.

In the past five years alone, Internet and computing technologies have grown at such a rate that many corporations today must provide substantial resources (both policy and human-powered) devoted solely to systems security including Payment Card Industry (PCI) standards. Keeping information such as customer and market data confidential has rapidly become one of the highest corporate priorities.

Welcome to the digitally global economy!

In April 2000, Visa began a proactive approach to payment security by announcing the Cardholder Information Security Program (CISP) as a standard for securing Visa cardholder data. Effective since June 2001, CISP compliance has been required for all entities that store, process or transmit Visa cardholder data.

Today, the former CISP program has been embraced, expanded upon and combined with other card associations requirements creating what is now called the PCI-DSS standard, defined exclusively to ensure multiple association compliance.

Note: This document (2.0) transitions prior Monetra installation/security guidance from a **voluntary** PABP validation(1.4) to the new **mandated** PA-DSS validation. If you are running a version of Monetra prior to 7.0 and/or are looking for archived PABP information please reference the previous Secure Implementation Guide version 1.5 or below.

1.2 The PCI Data Security Standard

The **P**ayment **C**ard **I**ndustry (PCI) **D**ata **S**ecurity **S**tandard offers a single approach to safeguarding sensitive data for all card brands. This standard is currently maintained by the newly formed PCI Security Standards Council.

The PCI Security Standards Council is an open global forum for the ongoing development, enhancement, storage, dissemination and implementation of security standards for account data protection. The organization was founded by American Express, Discover Financial Services, JCB, MasterCard Worldwide, and Visa International. The PCI Security Standards Council's mission is to enhance payment account data security by fostering broad adoption of the PCI Security Standards.

Note: As of this documents publication date, the most current PCI DSS standard is version 1.1 released September 2006 and may be downloaded from WWW.PCISECURITYSTANDARDS.ORG

1.3 Payment Application Data Security Standard (PA-DSS)

The PA-DSS applies to software vendors and others who develop payment applications that store, process, or transmit cardholder data as part of authorization or settlement. The PA-DSS also applies where these payment applications are sold, distributed, or licensed to third parties.

Traditional PCI Data Security Standard compliance may not apply directly to payment application vendors since most vendors do not store, process, or transmit cardholder data. However, since these payment applications are used by customers to store, process, and transmit cardholder data, and customers are required to be PCI Data Security Standard compliant, payment applications should facilitate, and not prevent, the customers' PCI Data Security Standard compliance.

Beginning in January 2008, the Industry implemented a new series of mandates, including the removal of vulnerable payment applications and the requirement for newly boarded Level 3 and 4 merchants to use PABP/PA-DSS compliant applications, if not PCI DSS compliant.

Note: The latest mandates can be found at our website www.monetra.com

1.4 Payment Security Chain of Command

Although the main job of systems security lies with the corporate CIO and/or the IT department, when implementing a payment system that will store, process, or transmit cardholder data, association rules will apply to the system. Much like the HIPAA security initiatives within the health industry, the payment requirements of CISP/PCI DSS strictly deal with securing sensitive/private data. Below is a list of the formal Payment Security chain of command, with respective roles discussed.

1.4.1 Card Associations

These are the actual associations such as Visa and Mastercard who define and ultimately enforce the rules.

As quoted from Visa's website: " The Visa USA Operating Regulations govern the activities of member financial institutions and, by extension, merchants and service providers as participants in the Visa payment system. The simplified requirements (located at http://www.visa.com/cisp/) should help clarify the intent of the more formal regulations."

1.4.2 Acquirers

These are the merchant Acquiring banks that underwrite merchant accounts onto the respective payment networks. This is either done in-house or through a network of Merchant Service Providers who act as agents/members for the Acquiring bank.

As quoted from Visa's website: "Members are responsible for ensuring the CISP compliance of their merchants, service providers, and their merchants' service providers. Although there may not be a direct contractual relationship between merchant service providers and acquiring members, all members remain responsible for any liability that may occur as a result of CISP non-compliance. Acquirers must include a CISP compliance provision in all contracts with merchants and non-member agents."

1.4.3 Merchants

The end company receiving payment for goods and/or services. Although the Acquirer or Merchant service provider should have explained (and be actively working you through) PCI DSS, it is ultimately the Merchant's responsibility for verification and compliance.

2 Merchant Requirements for Compliance

Depending on annual transaction volume, PCI-DSS requirements range from completing a self-assessment questionnaire to engaging an independent security assessor for conducting annual on-site security audits.

See www.www.pcisecuritystandards.org and/or contact your bank, processor, or acquirer for more information.

Notes on fines: As defined within the DSS, quoted from Visa's website “If a merchant or service provider does not comply with the security requirements or fails to rectify a security issue, Visa may:

- Fine the acquiring member
- Impose restrictions on the merchant or its agent
- Permanently prohibit the merchant or its agent from participating in Visa programs

Members receive protection from fines for merchants or service providers that have been compromised but found to be PCI-DSS-compliant at the time of the security breach. Members are subject to fines up to $500,000 per incident for any merchant or service provider that is compromised and not PCI-DSS-compliant at the time of the incident.”

Note: The PCI DSS requirements for your payment systems do not change, and must be validated, whether you use an in-house product like Monetra, or an SSC approved online service provider such as Verisign®. Before being validated, you must ask your staff if the entire system is conforming to the requirements or just the service provider themselves.

3 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.

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 transitions all prior Secure Implementation documentation from PABP to PA-DSS.

Versions 2.0 and higher of this guide will deal exclusively with PA-DSS standards. The most current version references PA-DSS v1.1 dated April 2008.

3.1 Do Not Retain Magnetic Stripe, CVVS/CVC2 or PinBlock (PVV) 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, 1.1.1, 1.1.2, 1.1.3, 1.1.4
PCI-DSS references	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 chapter 4 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.

1.1.6 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 requirements and securely removed when no longer needed.

Configuration Notes:

The monetra.log (debug) output level is configurable. Please reference chapter 4 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 5 (Monetra Encryption Keys) 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.
- 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 7 “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.

3.2 Stored Data Protection

PA-DSS reference	2.1
PCI-DSS reference	3.1

GUIDANCE: Keep cardholder data storage to a minimum. : 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 4 of this guide for details on data location and removal procedures.

PADSS reference	2.2
PCI-DSS reference	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.

PA-DSS reference	2.3
PCI-DSS reference	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: As of version 5.3.1, Monetra has the ability to run all cryptographic modules and procedures in a FIPS-140-2 compliant manner. See http://csrc.nist.gov/cryptval/ for more information regarding FIPS 140 (Security Requirements for Cryptographic Modules)

PA-DSS reference	2.4
PCI-DSS reference	3.4.1

Disk Encryption: Monetra performs file level encryption, not disk level.

PA-DSS reference	2.5
PCI-DSS reference	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. When Monetra starts up, a series of events are triggered for secure cryptographic key support.

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.

PA-DSS reference	2.6
PCI-DSS reference	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.

Please see chapter 6 of this guide, for detailed information regarding Encryption keys.

PA-DSS reference	2.7.a
PCI-DSS reference	3.6.5

Guidance regarding the removal/deletion of cryptographic material:

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.
4. Please review section 6.1.2 for instructions on key management/replacement.

PA-DSS reference	2.7.b

In accordance with 2.5 above, Monetra never uses a key in the clear, thus the working cryptographic material is considered non recoverable once deleted from the file system.

Configuration Notes:

Please review the information contained within this guide 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.

3.3 Provide Secure Password Features

PA-DSS reference	3.1

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

- 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 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 reference	3.2

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 reference	3.3

Encrypt application passwords: Monetra uses strong encryption to store application passwords. Monetra exposes the use of SSL 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:

Strong password management is configurable and can be handled at the Monetra level, or integrated at the application level. If you disable this feature in Monetra, the developer must provide strong password administration inside the integrated application. Please review the configuration guide for information on creating and maintaining strong password management.

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.

3.4 Log Application Activity

PA-DSS reference	4.1

Log access by individual users: Monetra provides extensive logging for security audits at both the internal and external level. External and internal Security logging is set to ON by default.

PA-DSS reference	4.2

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

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.

3.5 Develop Secure Applications

PA-DSS reference	5.1

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 reference	5.1.1

All changes/patches are tested via QA: Main Street tests all application changes internally via set QA procedures prior to releasing any code into a beta or production release.

1. All input is validated
2. All errors are properly handled
3. All cryptographic material is securely stored
4. Monetra provides secure communications technologies
5. Monetra handles RBAC properly.

PA-DSS reference	5.1.2

Separate Test and Production environments: Monetra is designed and applied in ANSI C as a self-contained, fully multi-threaded application. Our application is distributed TO the end user where the production environment resides.

PA-DSS reference	5.1.3

Separation of duties between development, test and production environments: 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 using a separation of duties between personnel assigned to test and production systems/environments.

PA-DSS reference	5.1.4

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.

PA-DSS reference	5.1.5

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.

PA-DSS reference	5.1.6

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 reference	5.1.7

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 reference	5.3, 5.3.1, 5.3.2, 5.3.3, 5.3.4

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 reference	5.4

Removal of unnecessary **and insecure services, applications and protocols:** Monetra is designed and applied in ANSI C as a self-contained, fully multi-threaded application. Monetra provides a small security footprint and requires minimal external libraries to function properly (see CIR below) . Example: Monetra does not require you to install vulnerable applications like Internet Explorer or Internet Information Server(IIS) to operate on the windows platform. It also does not require any special runtimes such as Java, which can add to the security footprint.

Note: Current Internal Requirements: Monetra only relies on 3 primary components to execute: OpenSSL (http://www.openssl.org), Zlib (http://www.gzip.org/zlib/) and the system-specific C library (including threading and math components) provided by the manufacturer of the OS on which Monetra resides. Both OpenSSL and Zlib are statically linked, so the version residing on the system does not inhibit system security. Other specific add-on modules may make use of other libraries, especially those which require communication with external SQL databases.

Integrator Notes:

The integrator is responsible for developing external applications that adhere to the standards as outlined in PA-DSS ref 5.2. If the integrator first installs Monetra into a test environment and then deploys to a live environment, the integrator will be responsible for the removal of all non-essential application accounts according to PCI standards 6.3.5 and 6.3.6

3.6 Protect Wireless Transmissions

PA-DSS references	6.1, 6.2

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 (PA-DSS ref 11.1, 11.2, 11.3 below) for more details.

Configuration Notes:

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

NOTE: If you install Monetra into a wireless environment, you must reference PCI-DSS sections 1.3.8, 2.1.1 and 4.1.1

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.

3.7 Test for Application Vulnerabilities

PA-DSS reference	7.1

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.

3.8 Facilitate Secure Network Implementations

PA-DSS reference	8.1

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.

Configuration Notes:

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

3.9 Never Store Cardholder Data on a Public-Facing Internet Connection

PA-DSS reference	9.1

Provide payment applications and data separation **facilities:** Monetra provides the ability to be fully separated from both the application and the database. 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 as the 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]).

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).

3.10 Facilitate Secure Remote Software Updates

PA-DSS reference	10.1

Provide Secure software updates: Monetra supports 'live update' features via a “subscribe/pull” method, where instead of Main Street needing access into a customer system, the customer simply downloads a new release from a known Main Street Server via a provided update utility.

Example: Very much like the anti-virus software update utilities in use today, Main Street provides an installer/updater utility for simplified (push button) software updates.

Integrator Notes:

- The integrator is responsible for securely implementing the required system update and maintenance policy regarding the Monetra software within a validated system. If the production system uses VPN and/or broad band “always-on” connections, then it is recommended you use a firewall product as per PCI ref 1.2

3.11 Facilitate Secure Remote Application Access

PA-DSS reference	11.1, 11.2, 11.3

Secure remote administration may be required from time to time by the merchant. SSL or HTTPS communications should be used.

Note: Main Street Softworks makes it a policy to NEVER connect into remote systems for any reason (installation, configuration upgrades etc.).

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 IP ruleset (firewall) to protect against unauthorized application access.

Configuration Notes:

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

Integrator Notes:

The integrator is responsible for the secure configuration of any Monetra application, in regards to remote access. When customers access Monetra remotely, it is a requirement to use two factor authentication.

Two factor authentication examples:

1. Application-Username/Password, SSL with certificate(s).
2. Application-Username/Password, VPN with certificate(s).
3. User-Radius, tokens.

3.12 Encrypt Traffic Over Public Networks

PA-DSS reference	12.1, 12.2

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 e-mail: By default, Monetra does not communicate any sensitive data via e-mail.

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.

3.13 Encrypt All Non-Console Administrative Access

PA-DSS reference	13.1

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.

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

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.

3.14 Maintain instructional material for customers and integrators

PA-DSS reference	14.1, 14.1.1, 14.1.2

A new requirement, as of PA-DSS is for us to develop, maintain, and disseminate a Secure Implementation Guide(s) for customers, resellers, and integrators.

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.

4 Monetra Application Data and Logging

4.1 Introduction

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 Internal data storage (for reporting) is designed to provide Audit support throughout the transaction life-cycle.

Note: As of Monetra 7.x 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.

4.1.1 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.

Note: As of Monetra 7.0.x the External logging facility has been enhanced to provide for detailed output in a structured format. If you are running a version prior to 7.0.x please ensure your 'debug=' setting is at 2 or lower, to remain compliant with security standards.

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	Uncategorized short information (like stats)
TRAN	PCI	Basic information on when a transaction enters the queue
TRAN_DETAIL	PCI	Basic incoming parameters as parsed (with sensitive data sanitized)
TRAN_TRACE	NonPCI	Parsed incoming and outgoing transaction requests unobscured.
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.
PROC_TRACE	NonPCI	Non-sanitized version of PROC_DETAIL
TRACE_IN	NonPCI	Raw trace of data in and out from client connections
TRACE_OUT	NonPCI	Raw trace of data between Monetra and the processors
SQL	NonPCI	SQL statements
ERROR	PCI	Any significant error condition
CRIT	PCI	A critical/significant error which must not be ignored
DEBUG	NonPCI	Very verbose, ungrouped data.
DEV	NonPCI	Reserved for internal development use only.

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

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

NOTE-1: When you configure a NonPCI 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 NonPCI 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.**

NOTE-2: A handy command-line utility is provided to help change the output level.

```
./monetra_setdebug -?
Usage: ./monetra_setdebug [-P madminpassword] [-m method] [-h hostname] [-p port] [-d
debugflags]
madminpassword  : defaults to none (entered interactively)
hostname        : defaults to 'localhost'
port            : defaults to 8333
method          : defaults to IP (SSL also available)
debugflags      : defaults to 'INIT|CONF|WARN|INFO|TRAN|TRAN_DETAIL|CONN|PROC|PROC_DETAIL|ERROR|
CRIT'
```

```
Ex: ./monetra_setdebug -P test123 -h 192.168.1.99 -p 8333 -d 'INIT|CONF|WARN|INFO|TRAN|
TRAN_DETAIL|CONN|PROC|PROC_DETAIL|ERROR|CRIT'

All flags:

INIT|CONF|WARN|INFO|TRAN|TRAN_DETAIL|TRAN_TRACE|CONN|PROC|PROC_DETAIL|PROC_TRACE|TRACE_IN|
TRACE_OUT|SQL|ERROR|CRIT|DEBUG|DEV
```

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

EXAMPLE:

```
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.

EXAMPLE:

```
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 file.

EXAMPLE:

```
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.

EXAMPLE:

```
debugkeep=10
```

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

EXAMPLE:

```
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.

EXAMPLE:

```
debugarchiveext=.bz2
```

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

EXAMPLE:

```
debugrotaterestart=YES
```

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

EXAMPLE:

```
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).

EXAMPLE:

```
debugrotatesize=10240
```

4.1.2 Internal Data Storage

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

connlog	Connection Details Logging	lists raw connection data on a per-engine basis. Note: this log can be cleared using the clear conn log function (clearconnlog).
tranlog	Transaction Details Logging	lists auditable transaction details with connection identifier matching on a per- engine basis. Note: this log can be cleared using the clear transaction log function (cleartranlog)
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.

5 Monetra Virtual User SubSystem

5.1 Introduction

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.

5.1.1 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'.

5.1.2 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.

5.1.3 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.

6 Monetra Encryption Keys

6.1 Introduction to 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.

6.1.1 Encryption Key Creation

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 keylen outfile [egd pool]

Example:

```
$ /usr/local/monetra/bin/monetra_keygen 256 /usr/local/monetra/my_monetra.key
```

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

6.1.2 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 datafile 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 proepr 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.

7 Monetra Client Certificates

7.1 Introduction to Client Certificates

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.

7.1.1 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,

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 Cert
./CA/certs
./CA/crl
./CA/index.txt
./CA/newcerts
./CA/private
./CA/private/cakey.pem  <-- CA private key
./CA/serial
```

cacert.pem should look like this:

```
-----BEGIN CERTIFICATE-----
MIIDYzCCAsygAwIBAgIJAKcV7BfxXP68MA0GCSqGSIb3DQEBBAUAMH8xCzAJBgNVBAYTAlVTMRAwDgYDVQQIEwdGbG9yaWRh
MRQwEgYDVQQHEwtHYWluZXN2aWxsZTEkMCIGA1UEChMbTWFpbiBTdHJlZXQgU29mdHdvcmtzLCBJbmMuMSIwIAYJKoZIhvcN
AQkBFhNzdXBwb3J0QG1vbmV0cmEuY29tMB4XDTA1MDYwOTEzNTMyMVoXDTA2MDYwOTEzNTMyMVowfzELMAkGA1UEBhMCVVMx
EDAOBgNVBAgTB0Zsb3JpZGExFDASBgNVBAcTC0dhaW5lc3ZpbGxlMSQwIgYDVQQKExtNYWluIFN0cmVldCBTb2Z0d29ya3Ms
EluYy4xIjAgBgkqhkiG9w0BCQEWE3N1cHBvcnRAbW9uZXRyYS5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAL1xN
VuiUpYu4lRxWVdcAv/Fd1JVujLhHAgMo1BA1w8br2QJ7+
+eK7BXEU3mCN8jmuUyn2R/nE2U87X+RdY0KH9FQ7Abfj1b2vOYcmVHBvgm8FGBeueew8900M3xi/Gwz8AINOFCEg/
+/0TuLkCUDg4RFuJDOyZQaGHzMSeGreZrAgMBAAGjgeYwgeMwHQYDVR0OBBYEFCGABWx3AnXYZysdlhtL/5qBZxDbMIGzKIj
blieJRLILJDlivliEJflIDlfildIFJLIdfhqBZxDboYGEpIGBMH8xCzAJBgNVBAYTAlVTMRAwDgYDVQQIEwdGbG9yaWRhMRQ
wEgYDVQQHEwtHYWluZXN2aWxsZTEkMCIGA1UEChMbTWFpbiBTdHJlZXQgU29mdHdvcmtzLCBJbmMuMSIwIAYJKoZIhvcNAQk
BFhNzdXBwb3J0QG1vbmV0cmEuY29tggkApxXsF/Fc/rwwDAYDVR0TBAUwAwEB/zANBgkqhkiG9w0BAQQFAAOBgQBVwrhdYFn
koISAwivmKbs6Mf7OgmvTm5Cg+hjG7VBwPbJf991tsxqirbEd9W3tqGJ58RKJchnstrcihpfdMptFfKDaNDawG6xWiHUYvmw
zkaAI+ciXBm/DpvDZvm9A09SCOHz+aJNILazyhz9q38MAtK04vwT7o5Cac/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/r
kEQJDUGGnRihoi5ZyrHi68fJOrO4G1xvCf37IMSb65LGcNKWtaEFo2YV5SnJfYtJU1vAUh4LV4Zt5ZT6CWGTFcxVzmtEWaCj
pDEvKliclbjLDIFj4idfldDO1xAoAaesVsD1V+KsxrSdqwek5T1TzB9478TtbodfRRzRlsE6bFVNDkdQFv3im/ZgM4rFhAHi
hRffQy7mW6XGkg19T1JbcZfIHXGUrzOi3iOb6mBJVANZcjZdwnbOsrbWcUPBEXgHcQUrsce7qhzx5LaEQUowEJHAhERQ5GxG
RAmeHz8MeyJklolnH5chVkmEAKBRrWMi4EsPHdfgXVyEG9uJ2StbHDuh7N6fkmCGXwrblbkpf1M6/j/i31e58FVOB616Oufa
1EaSDzeaCQzm/uWIIiCz0RsxXY/hoGOZ9miyQjjZru0OiVFf8QNsF2MpD2uVraWSyYTcGgF0x8hKXvlcnsi3aNlzzMT+iGxO
OEWdIbFsIfD9YqQ35h9BAkn7weJQfDAlD2K6noktgBmfcYJlUqGDt46/a990nKek6jQH02rPngZMRNaO/97VgNbgiZ1zkBq2
p83FY397IhuSVRLqq5wW7S4dXNyu7J3+tIqN5LiUdePGqyix7ltOovFqmV5cWnKVquK02Vc/Y8KNZNFBREC4WT2m8663DThC
2ocQrmleHcLP7YoGCWp4EAmBpxHrU0FQYCTX7tIXO5KKLRPjenrPQ==
  -----END RSA PRIVATE KEY-----
```

7.1.2 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:

```
enableSSL=yes
SSLCertRequired=yes
SSLCAFile=/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:

```
SSLCAFile: /etc/monetra/mycafile.pem
SSLCertRequired: 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().

7.1.3 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
```

8 Alternate file system security

8.1 Notes on File System Security, as applied to PA-DSS

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.

8.2 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://www.microsoft.com/technet/prodtechnol/winxppro/deploy/cryptfs.mspx**

8.3 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.

Examples of FREE ramdisk utilities include.

Windows:

http://www.mydigitallife.info/2007/05/27/free-ramdisk-for-windows-vista-xp-2000-and-2003-server/

Note: The Linux kernel version 2.4 has built-in support for ramdisks.

Nice HOWTO: http://www.vanemery.com/Linux/Ramdisk/ramdisk.html

For Mac OSX users:

```
hdid -nomount ram://128000
```

This will create a virtual disk that is 64MB (the size you specify is 512byte blocks). It will output the path to the virtual disk that was created /dev/disk2...

Next, you would format the disk.

```
newfs_hfs /dev/disk2
```

Once the disk is formatted, create a mount point.

```
mkdir /tmp/mymount
```

Now you can mount the disk.

```
mount -t hfs /dev/disk2 /tmp/mymount
```

When you are ready to un-mount the disk you would use this command.

```
umount -f /dev/disk2
hdiutil detach /dev/disk2
```

Here's a Mac OS X helper script:

```
----------------
#!/bin/sh
die() {
        echo "$*"
        exit 1
}

if [ "$#" != "2" ] ; then
        die "Usage: $0 size_in_MB mount_point"
fi

if [ ! -d "$2" ] ; then
        die "$2 doesn't exist as a directory, create it"
fi

size=`expr $1 '*' 1024 '*' 2`
echo "Creating ramdisk of ${size} sectors..."
mydev=`hdid -nomount ram://${size} | tr -d '[:space:]'`
if [ ! -e "$mydev" ] ; then
        die "hdid failed ($mydev)..."
fi
newfs_hfs $mydev || die "newfs failed"
mount -t hfs $mydev "$2" || die "mount $mydev to $2 failed"
echo "$mydev mounted on $2"
```

9 Operating System Information

9.1 PCI/PA-DSS compliance statement

The current version of Monetra is developed and deployed to support Multiple Operating Systems including Linux, Unix, Mac OSX, Microsoft Windows and others.

According to current PCI requirements, a validated application must execute from a System that is supported by the manufacturer, to include up-to-date security related patches and enhancements. Reference PCI 6.1.

If you are unsure of which operating systems are valid, please reference the current Operating System Manufacturers support page.

Example for Microsoft:

http://www.microsoft.com/windows/lifecycle/default.mspx

Examples of unsupported Operating Systems':

- Windows NT 4.XX
- Windows 98
- Windows 98-SE
- Windows ME
- RedHat Linux 6,7, 8, 9
- IBM AIX 4.3

As per vendor notices:

http://www.microsoft.com/windows/support/endofsupport.mspx

http://www.microsoft.com/windows/lifecycle/servicepacks.mspx

http://www.redhat.com/security/updates/eol/

10 Resources on the World Wide Web

10.1 Card Association Links

PCI Security Standards Council, LLC.

http:www.pcisecuritystandards.org

Visa: http://www.visa.com/cisp

Master Card: http://www.mastercard.com

http://www.mastercardmerchant.com/datasecurity/data_protection.html

American Express:

http://home.americanexpress.com/homepage/merchant_ne.shtml

10.2 Security Organizations

CERT: http://www.cert.org/

Security Authorities: Bureau of Industry and Security:

http://www.bis.doc.gov/

FBI CyberSecurity: http://www.fbi.gov/cyberinvest/cyberhome.htm

U.S. Secret Service: http://www.secretservice.gov/financial_crimes.shtml

INTERPOL: http://www.interpol.int/Public/TechnologyCrime/default.asp

10.3 Operating Systems

Linux: http://www.kernel.org/ http://www.nsa.gov/selinux/

FreeBSD: http://www.freebsd.org/security/

Apple Mac OS X: http://www.apple.com/macosx/features/security/

IBM AIX: http://www-1.ibm.com/servers/aix/overview/security.html

Sun Microsystems Solaris: http://www.sun.com/software/security/

SCO Unix: http://www.sco.com/support/security/

Microsoft Windows: http://www.microsoft.com/security/default.mspx

10.4 Monetra Related Technologies

SSL: http://www.openssl.org/

ZLIB: http://www.zlib.net/

GNU C: http://www.gnu.org/software/libc/libc.html

11 References, Acknowledgments, License

11.1 References

This document references the following publications.

1. PCI-DSS version 1.1 (September 2006)
2. PA-DSS version 1.1 (April 2008)
3. Monetra Installation and Configuration Guides

11.2 Acknowledgments

Main Street's software products actively use, support and promote the Open Source SSL toolkit located at http://www.openssl.org .