

Monetra® Payment Software

Installation, Configuration, and Secure Implementation Guide

Revision: 9.0.2

Publication date September 30, 2022

Installation, Configuration, and Secure Implementation Guide

Monetra Technologies, LLC

Revision: 9.0.2

Publication date September 30, 2022

Copyright © 2022 Monetra Technologies, LLC

Legal Notice

The information contained herein is provided *As Is* without warranty of any kind, express or implied, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose. There is no warranty that the information or the use thereof does not infringe a patent, trademark, copyright, or trade secret.

Monetra Technologies, LLC. SHALL NOT BE LIABLE FOR ANY DIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, WHETHER RESULTING FROM BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, OR OTHERWISE, EVEN IF MONETRA TECHNOLOGIES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. MONETRA TECHNOLOGIES RESERVES THE RIGHT TO MAKE CHANGES TO THE INFORMATION CONTAINED HEREIN AT ANYTIME WITHOUT NOTICE. NO PART OF THIS DOCUMENT MAY BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, FOR ANY PURPOSE, WITHOUT THE EXPRESS WRITTEN PERMISSION OF Monetra Technologies, LLC.

Table of Contents

1. Revision History	1
2. Monetra Payment Engine	2
2.1. Overview	2
2.2. Architecture	2
2.2.1. Hierarchical Users	3
2.3. System Requirements	5
2.3.1. Hardware	5
2.3.2. Software	5
2.4. Licensing	6
2.5. Versioning	7
2.5.1. Version Scheme	7
2.6. Product Lifecycle And Support	7
3. Installation	9
3.1. Prerequisites	9
3.2. Monetra Installer	9
3.2.1. Windows	10
3.2.2. MacOS	11
3.2.3. Linux/Unix/Other	12
3.3. Starting The Installer	14
3.4. Installation Methods	15
3.4.1. Online Installation (Recommended)	15
3.4.2. Offline Installation (Alternative)	16
3.5. Interacting With The Monetra Installer	19
3.6. Installing Monetra	21
3.7. Installing GUI Helper Utilities	21
3.7.1. Monetra Manager	21
3.7.2. Monetra Administrator	21
3.7.3. Monetra Client	21
3.8. System Tuning	22
3.8.1. Linux Host Tuning Example	22
4. Configuration	24
4.1. Logging	24
4.1.1. File Logging	25
4.1.2. Syslog	26
4.2. Database	26
4.2.1. Driver: MySQL	28
4.2.2. Driver: Oracle	29
4.2.3. Driver: ODBC And DB2	29
4.2.4. Driver: PostgreSQL	30
4.2.5. Driver: SQLite	30
4.3. Key Management For Data At Rest	30
4.3.1. Protection Mechanisms	30
4.3.2. Key Generation	34
4.4. Starting And Stopping Monetra	35
4.4.1. Post-Startup Activation Requirements	35
4.5. Mail Configuration For Notices, Settlement Summaries, MFA, and Receipts	35

4.5.1. SMTP Direct Send	36
4.5.2. External Mailer	36
4.6. SMS configuration for MFA, Password Resets, Receipts	37
4.7. Clustering	37
4.8. Advanced Configuration	37
4.9. Creating Initial Administrator User(s)	38
4.10. Activating Processing Institutions	39
4.11. Configuring Data Retention	40
5. Merchant Configuration	43
5.1. Merchant Boarding Parameters	43
5.2. Processing Institution Enablement	43
5.3. Adding The Merchant Profile	43
5.4. Adding Additional Routes	45
5.5. Setting Up Users	47
5.6. Automating Settlement Tasks	48
6. Upgrading	49
6.1. Evaluating Changes	49
6.2. Backing up And Restoring	50
6.3. Implications Of Data Export/Import	50
6.4. Upgrading	51
6.4.1. Cluster Upgrades	51
7. Secure Implementation And Deployment	53
7.1. Firewalls	53
7.2. Database	54
7.2.1. Deployment	54
7.2.2. Key Rotation	54
7.2.3. Sensitive Data Locations	55
7.3. Data Retention	56
7.4. Inbound Communications Protocols	57
7.5. Cryptographic Subsystem Information	57
7.6. Signed TLS Server Certificates For Public-Facing Installations	57
7.6.1. Generating Signing Request With OpenSSL	58
7.6.2. Installing The Certificate	59
7.7. Multi-factor Authentication (MFA)	59
7.7.1. Time-based One Time Pin (TOTP)	59
7.7.2. Requiring TLS Client (Machine) Certificates	60
7.8. API Keys for Integrated Applications	61
7.9. Self-Service MFA and Password Resets	61
7.10. Token Export	61
7.11. Security-Affecting Configuration Options	62
7.11.1. <code>main.conf</code>	62
7.11.2. <code>prefs.conf</code>	63
7.12. PCI Security And Implementation	64
A. [SAMPLE] Cryptographic Material Custodian Agreement	69

1 Revision History

Version	Date	Changes
v9.0.2	2022-09-30	<ul style="list-style-type: none">Remove section regarding wildcard versioning, add information about PCI SSF listing.
v9.0.1	2022-07-29	<ul style="list-style-type: none">Note that must upgrade to 8.20.1 before going to 9.0.1+
v9.0.0	2022-07-11	<ul style="list-style-type: none">Remove mention of configuration options that were long deprecated and no longer part of Monetra v9.Removal of <code>hsmfull</code> mode of operation.Document automatic data encryption key rotation configuration.Update to reference PCI SSF instead of PA-DSS.Update list of supported operating systems.Note Monetra v8 supported until July 1, 2023.Add sections better explaining the cryptographic subsystemAdd sections explaining password resets, MFA, and API Keys.
v8.13.1	2020-02-24	<ul style="list-style-type: none">Re-order database key protection mechanism sections and add clarifications for passphrase key protection.Clarifications for CA certificate signing requests.Add clarification regarding running mixed Monetra versions in a cluster during upgrade.Add clarification regarding use of PBKDF2 password protection limitations.Add clarifications for SQLite secure deployments.Add system hardening installation prerequisite.Clarifications for PA-DSS data retention requirements.Clarifications for split-knowledge/dual-control key protections.Clarify that logging is required to be enabled for PCI DSS.Update guidance for wireless networks by duplicating information from the PCI DSS standard.Document password length/strength requirements that are enforced by Monetra.Add licensing section for additional clarification on product updates.
v8.13.0	2019-12-09	<ul style="list-style-type: none">First Release of redesigned documentation.

2 Monetra Payment Engine

2.1. Overview	2
2.2. Architecture	2
2.2.1. Hierarchical Users	3
2.3. System Requirements	5
2.3.1. Hardware	5
2.3.2. Software	5
2.4. Licensing	6
2.5. Versioning	7
2.5.1. Version Scheme	7
2.6. Product Lifecycle And Support	7

2.1 Overview

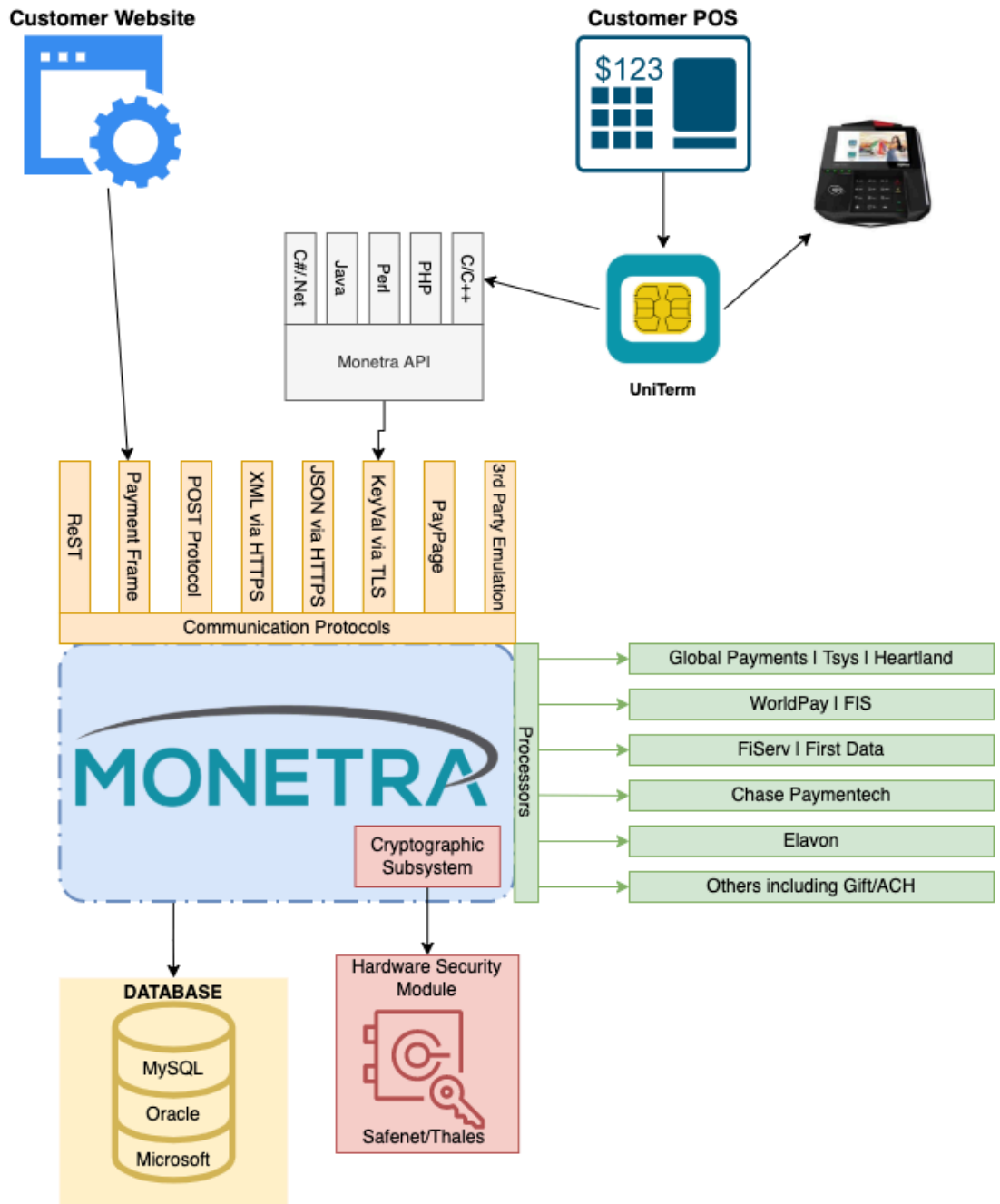
Monetra is a fast, efficient, and secure payment application. It's designed to scale from small, custom, embedded devices to fully-redundant payment servers processing thousands of transactions per minute. Trusted for over 20 years by thousands of merchants throughout North America, Monetra is the premier product of its type.

Monetra supports critical payment features such as:

- Extensive EMV Processor certifications across the US and Canada
- Robust Tokenized Card Storage and Recurring Billing
- P2PE Card Encryption proven to comply with PCI P2PE standards, including HSM support
- Flexible, developer-friendly integration options, including ReSTful APIs and iFrame integrations
- Clustering support for redundancy and load balancing.

2.2 Architecture

Monetra is written in 100% ANSI C89 to be compatible across all major and embedded operating systems, making it one of the most efficient, scalable, and portable payment engines available. It is built on a small payment core that houses all security and routing details, with all other features implemented via our flexible modular subsystem, which acts as an abstraction layer so that a change to the core does not impact a module and vice-versa. Inbound communications protocols, databases, processing institutions, etc. are all designed as separate modules to a common interface.



2.2.1 Hierarchical Users

Monetra is designed as a hierarchical user system; this means that depending on where a user or profile exists within the hierarchy, controls their view of the rest of the system. Each group can have an unlimited number of users, merchant profiles, and other groups attached. There is, however, a maximum group depth of 8 enforced by the system. The top-level group is auto-created and reserved for global or sysadmin users. No merchant profiles are allowed to be

attached to the top-level group, only other groups and users. Every individual merchant should be isolated into their own group with their own users assigned for access.

An example group hierarchy might look like the below image:



Members of the hierarchy:

- Group - A node in the hierarchical tree.

Each new group which is chained to another group creates a new level of the hierarchy. There are 8 total allowed levels of groups (one group chained to another group), however there is no limit on the number of groups allowed to be associated with a single group. A group may be assigned flags and permissions. Flags can be used to send push notifications or configure tokens for being shared across any members below it in the hierarchy. Each group has a permission set to ensure no user below that point in the hierarchy can contain greater permissions than the group itself.

A group may have a User, Merchant Profile, or another group attached to them as a child. The top-level group, however, cannot be associated with any merchant profiles, you must create a child group.

All users in a group can access any merchant profile in the same group as well as any chained groups below them.

- User - A real person with need to access the system.

A User may be assigned to any group in the hierarchy. Where they are placed in the hierarchy controls their "view". A user may also be assigned fine-grained permissions for what actions they are allowed to perform.

- Merchant Profile - A single physical store, phone center, or website.

A merchant profile is used to group payments for one location. A profile may have multiple routes for defining different payment channels.

- Route - A payment channel such as credit cards, gift cards, or how they are routed for authorization vs settlement.

There are one or more routes associated with each merchant profile. Each route will differ by card type or authorization vs settlement. Each route is going to be bound to a processing institution.

2.3 System Requirements

2.3.1 Hardware

Monetra may be deployed on a back-office server supporting a single merchant location, a centralized server (e.g. cloud) cluster supporting tens of thousands of merchant locations, or a dedicated payment device. Each deployment option has different requirements for available system resources.

Please see the recommended system requirements below for each deployment scenario. The numbers listed are for recommended available resources after taking into account the requirements for the Operating System and any other software.

- Back Office: 1GB RAM - 1x1GHz CPU - 256MB Disk + 768 bytes per transaction
- Cloud: 4GB RAM - 4x2GHz CPU - 256MB Disk + 1024 bytes per transaction
- Dedicated Device: 256MB RAM - 1x400MHz CPU - 256MB Disk + 768 bytes per transaction

2.3.2 Software

Monetra requires only a base installation of a supported Operating System to function. All components are either implemented directly within Monetra or provided by the Monetra installation. There is no dependency on external components, such as Java, .Net, IE, or IIS. However, most installations will need a database deployment, but that is most typically running on another node on the network and not co-installed.

2.3.2.1 Operating Systems

Though Monetra will run on many operating systems, not all builds have gone through the stringent PCI Software Security Framework (SSF) validation process due to necessary time constraints and cost considerations. For these non-validated environments, it is the responsibility of the installer to ensure that the integration is validated as part of the merchant's

or service provider's PCI DSS validation. The same code base is used for all operating systems, and since there are no external dependencies, there are no anticipated behavior or security-related differences from environment to environment that would impact the ability to receive such an external validation.

Supported and validated operating systems:

- RHEL 8 x64
- RockyLinux 8 x64
- AlmaLinux 8 x64
- Ubuntu 20.04LTS x64
- Debian 11 x64
- Windows Server 2019 x64
- Windows Server 2022 x64

Other operating systems that Monetra is supported on but have not undergone PCI SSF validation:

- Windows 10 or higher 64bit (x64), other than listed systems
- Linux with glibc 2.17 or higher on x64, other than listed systems
- Custom embedded systems



PCI Notice: Any operating system Monetra is deployed on must be supported by the respective vendor and have all relevant security patches applied and be kept up-to-date and patched.

2.3.2.2 Databases

Monetra supports all major relational SQL database systems on the market. Most installation environments will already have a production-grade database in place on the network, and it is recommended to use this existing resource rather than running a dedicated system just for Monetra (as long as the existing system is deployed following all PCI DSS rules). While all data is encrypted using strong field-level encryption prior to entering the SQL abstraction layer, PCI still considers encrypted cardholder data at rest as in-scope.

Supported databases:

- SQLite - Simple file-based database. Bundled with Monetra. Only suitable for dedicated payment devices.
- Microsoft SQL Server 2012 or higher
- MySQL 5.7 or higher (including derivatives such as MariaDB or Percona XtraDB)
- Oracle 11.2 or higher
- PostgreSQL 10.2 or higher
- IBM DB2/UDB 9.7 or higher

2.4 Licensing

Monetra is licensed using a subscription model, typically with an initial, front-loaded fee followed by a lower, annual subscription fee that includes support and maintenance, such

as product updates. Product updates include access to new features, bug fixes, interchange-compliance updates, and security patches. Some new features may come with added fees.

Maintaining an active support and maintenance agreement is required in order to receive security patches. PCI-DSS rules state that all merchants must deploy vendor-supplied security patches. An active agreement is thus a requirement to remain in compliance and continue using Monetra. Monetra will not automatically disable itself when a support agreement has expired; it is the responsibility of the merchant to ensure an active support agreement is in place.

Once a support agreement has expired, the ability to download the software for any version (including the version originally installed) is disabled. This is to ensure that merchants do not attempt to deploy software with known security issues, which would violate PCI-DSS rules.

Please see [Chapter 6: Upgrading](#) for more information.

2.5 Versioning

2.5.1 Version Scheme

The versioning scheme employed by Monetra is formatted as $x.y.z$, where x , y , and z 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 approved and listed software version will appear as "Monetra 9.0.0" on the PCI SSC List of Validated Payment Software.

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 SSF.

The y component of the version indicates a product 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 SSF.

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 they may include security-related fixes, such as updates to 3rd party libraries (e.g. cryptographic libraries) distributed with Monetra.

2.6 Product Lifecycle And Support

Monetra may have multiple version streams that are simultaneously supported. Only the latest version of any active stream qualifies for technical support and will continue to receive updates.

A new stream is created only under a couple of scenarios:

- A database schema break has occurred
- A major integration compatibility break has occurred

In both scenarios, due to the time required to schedule maintenance windows, and possibly additional development and QA efforts, Monetra will receive varying levels of support.

For database schema breaks, the previous Monetra release stream will be supported for one year from the point in time a new release is made with a new database schema version. Only significant bug fixes and security fixes will be made available for this release stream. No features or processor certification updates will be provided.

For major integration compatibility breakage, the previous Monetra release stream will be supported for two years from the point in time a new release stream is made with the integration break. The release stream will not receive any feature enhancements, but will qualify for bug fixes, security fixes, and any card brand interchange requirements that may be introduced and enforced during the given time frame the stream is supported.

If both scenarios occur for the same release, then the longer release stream rules will apply.



Note: As of July 2022, there are two streams currently supported. The recently released Monetra v9 and the prior generation Monetra v8. The Monetra v8 series will be supported until July 1, 2023 as it is considered a database schema break only due to the v8 emulation layer contained within v9.

3 Installation

3.1. Prerequisites	9
3.2. Monetra Installer	9
3.2.1. Windows	10
3.2.2. MacOS	11
3.2.3. Linux/Unix/Other	12
3.3. Starting The Installer	14
3.4. Installation Methods	15
3.4.1. Online Installation (Recommended)	15
3.4.2. Offline Installation (Alternative)	16
3.5. Interacting With The Monetra Installer	19
3.6. Installing Monetra	21
3.7. Installing GUI Helper Utilities	21
3.7.1. Monetra Manager	21
3.7.2. Monetra Administrator	21
3.7.3. Monetra Client	21
3.8. System Tuning	22
3.8.1. Linux Host Tuning Example	22

3.1 Prerequisites

Customers and integrators must ensure that underlying software and systems prevent inadvertent capture and retention of cardholder data.

For Windows, methods include [examples]:

- Disabling System Restore
- Encrypting the `Pagefile.sys`
- Clearing the `Pagefile` at shutdown

For Linux, it is recommended to disable or encrypt the system swap.

For systems and databases, backup processes should be reviewed, to ensure all locations of stored cardholder data are documented and align with the customer's data retention policy. Consideration should also be given to the use of other technologies in use such as system snapshots.

3.2 Monetra Installer

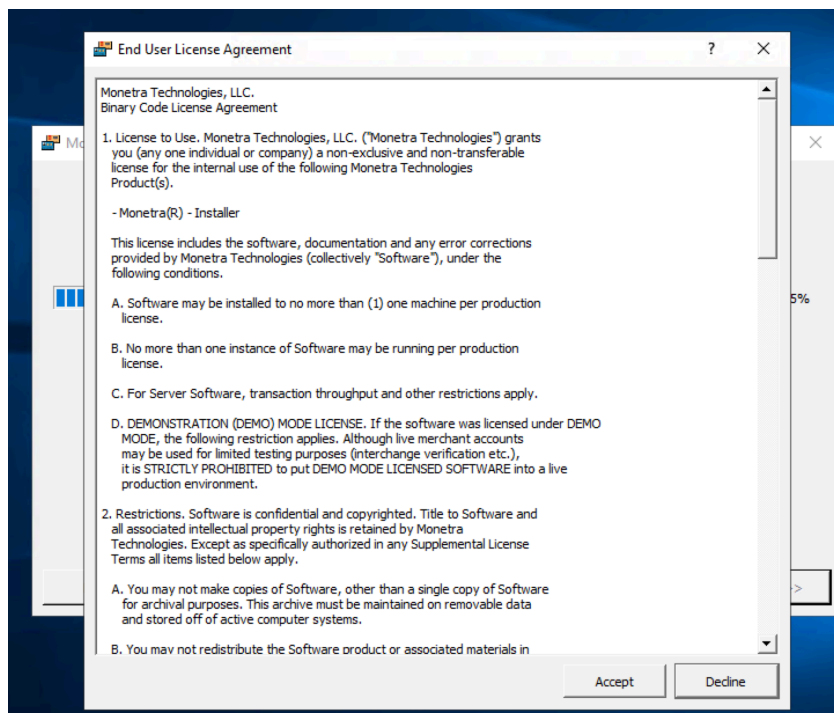
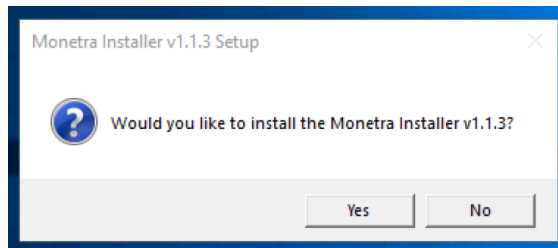
The Monetra Installer is our custom utility for installing and upgrading the Monetra payment engine as well as any helper utilities. The features and functionality are the same across all the operating systems Monetra will run on, and this is the only supported method for installing and upgrading Monetra.

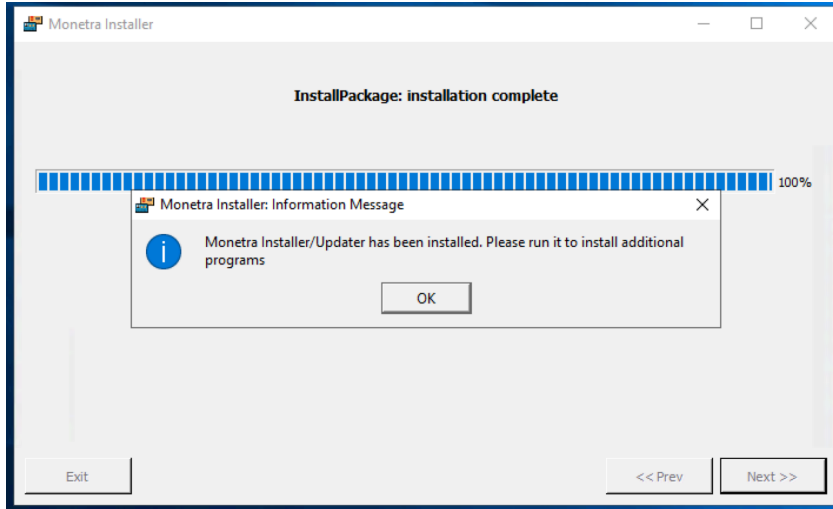
The Monetra Installer can be downloaded from <https://www.monetra.com/downloads/>. This package is used to install the Installer on a new system, but it can also be used to override or

upgrade an existing Installer installation. The installation process can vary from platform to platform and is outlined in the following sections.

3.2.1 Windows

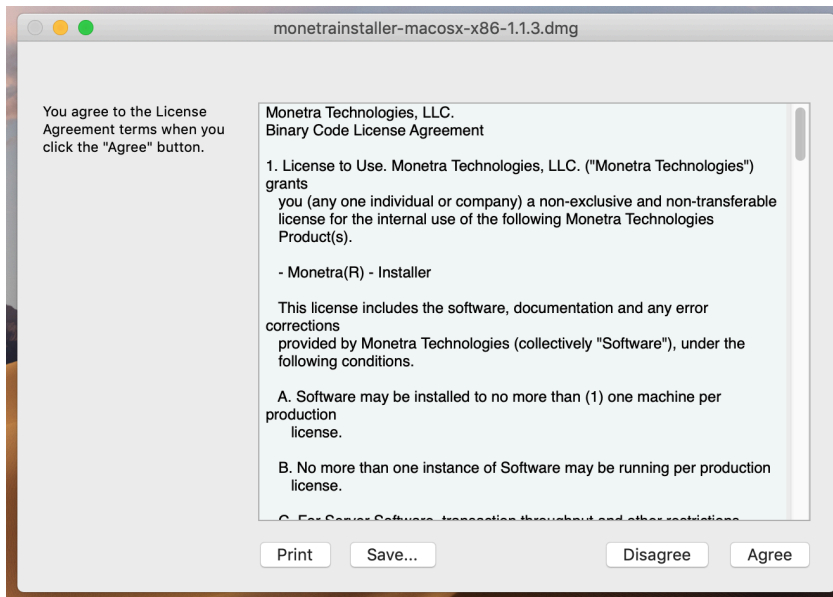
The Installer on Windows is downloaded as a signed executable file. Simply double-clicking on the downloaded Installer will start the installation process. Follow the on-screen prompts. The below screen shots are representative of the process.

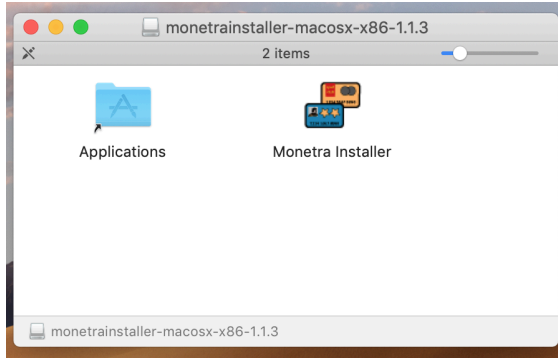




3.2.2 MacOS

The Installer on MacOS is a Disk Image (DMG). Double-clicking on it will mount it as a virtual drive, inside of which will be the Monetra Installer application and a link to the Applications folder. All that is needed to be done for installation is to drag the application to the Applications link. The below screen shots are representative of the process.





3.2.3 Linux/Unix/Other

The Installer on other systems is a self-extracting tarball with a `.run` extension. In order to run the installation, first make the package executable via

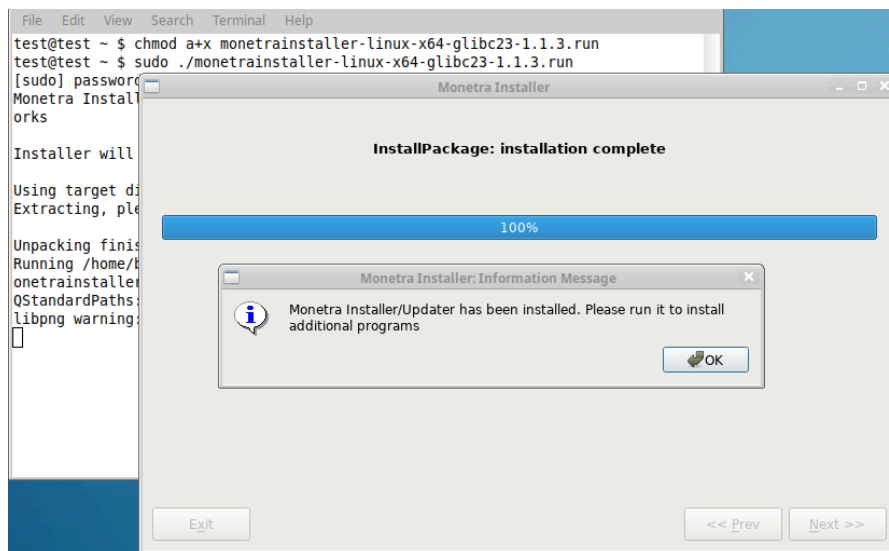
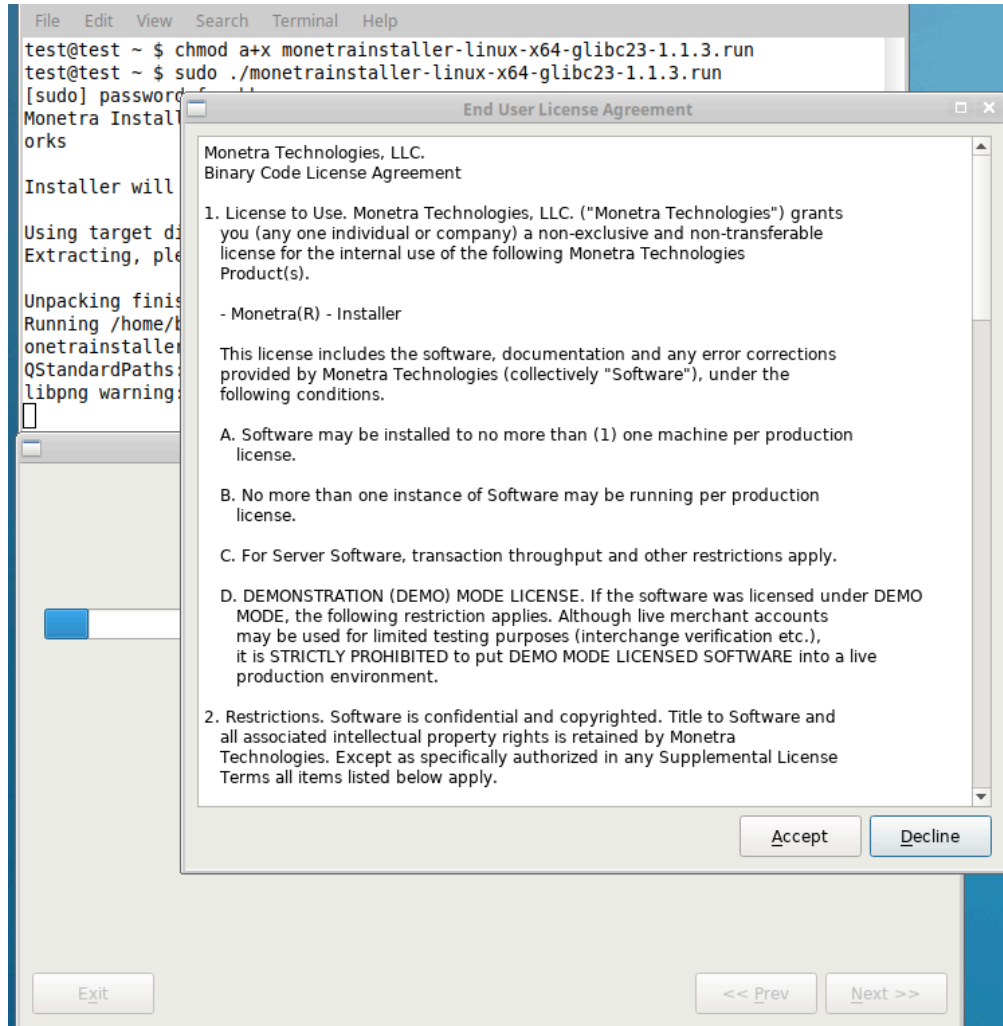
```
chmod +x monetrainstaller-linux-x64-glibc23-1.1.6.run
```

(or similar), then execute it as root or via `sudo`, such as

```
sudo ./monetrainstaller-linux-x64-glibc23-1.1.6.run
```

The Installer will operate in either graphical mode or text mode depending on the presence of a supported graphical subsystem (e.g. X11). The below screen shots are representative of the process.

For a GUI installation:



For a console installation:

```
File Edit View Search Terminal Help
test@test ~ $ chmod a+x monetrainstaller-linux-x64-glibc23-1.1.3.run
test@test ~ $ sudo ./monetrainstaller-linux-x64-glibc23-1.1.3.run
Monetra Installer Self-Extractor Version: 1.1.3, Copyright (c) Main Street Softw
orks

Installer will be extracted to: /home/lu...

Using target directory: /home/lu.../monetrainstaller-linux-x64-glibc23-1.1.3
Extracting, please wait...

Unpacking finished successfully.
Running /home/lu.../monetrainstaller-linux-x64-glibc23-1.1.3/monetrainstaller/m
onetrainstaller.app/bin/monetrainstaller ...

InstallPackage: read package information
[**-----] 5 %

The EULA will be output below
Press enter to continue...


File Edit View Search Terminal Help
Installing: /usr/local/monetrainstaller/libexec/monetrainstaller/lib/libusb-1.0.
Installing: /usr/local/monetrainstaller/libexec/monetrainstaller/lib/mstdlib_sql
Installing: /usr/local/monetrainstaller/libexec/monetrainstaller/lib/mstdlib_sql
Installing: /usr/local/monetrainstaller/libexec/monetrainstaller/lib/mstdlib_sql
Installing: /usr/local/monetrainstaller/libexec/monetrainstaller/lib/mstdlib_sql
Installing: /usr/local/monetrainstaller/libexec/monetrainstaller/bin/monetrainst
Installing: /usr/local/monetrainstaller/libexec/monetrainstaller/bin/monetrainst
Installing: /usr/local/monetrainstaller/libexec/monetrainstaller/bin/imageformat
Installing: /usr/local/monetrainstaller/libexec/monetrainstaller/bin/imageformat
Installing: /usr/local/monetrainstaller/libexec/monetrainstaller/bin/imageformat
Installing: /usr/local/monetrainstaller/libexec/monetrainstaller/bin/imageformat
Installing: /usr/local/monetrainstaller/libexec/monetrainstaller/bin/imageformat
Installing: /usr/local/monetrainstaller/libexec/monetrainstaller/bin/printsuppor
Installing: /usr/local/monetrainstaller/libexec/monetrainstaller/bin/platforms/l
Installing: /usr/local/monetrainstaller/share/monetrainstaller/installer_changel
InstallPackage: installation complete
[*****] 100 %

Monetra Installer/Updater has been installed. Please run it to install additiona
l programs
Press enter to continue...

```

3.3 Starting The Installer

On Windows, the Monetra Installer can be found on the Desktop or in the Programs list under "Main Street Softworks". Simply clicking on the icon and approving administrative rights is all that is required to start it.

On MacOS, the Monetra Installer can be found under Applications. Simply double-clicking the icon and entering your username and password to grant administrative rights is all that is required.

On Linux/Unix systems, the Monetra Installer is installed into `/usr/local/monetrainstaller` by default and can be executed via `/usr/local/monetrainstaller/bin/monetrainstaller`. It must be executed as root, or via `sudo`.

3.4 Installation Methods

3.4.1 Online Installation (Recommended)

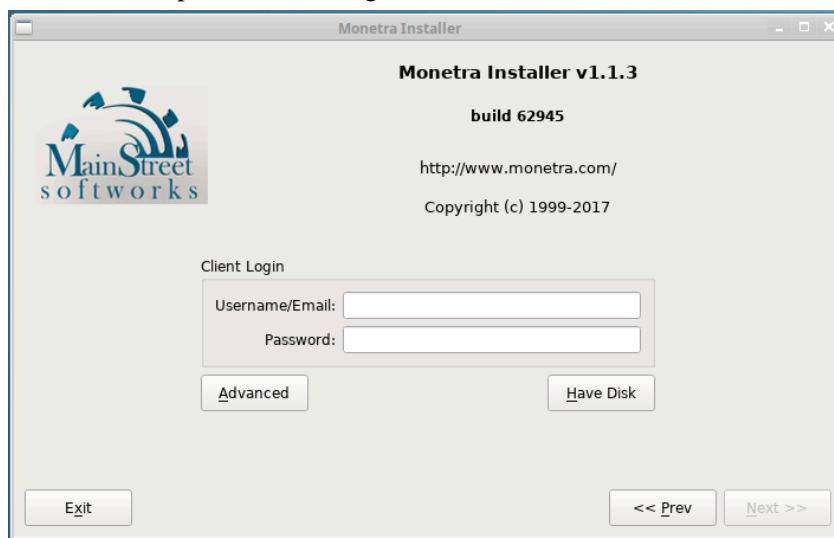
For an online installation, the Monetra Installer will fetch the installation packages and license keys from the servers hosted by Monetra Technologies. This is the most common and recommended method of installation, as well as the quickest. There are only 2 requirements for this method: account credentials and an active Internet connection.

The credentials are the username and password associated with your account, which are the same as you would use to log in to <https://www.monetra.com/>.

The active Internet connection needs firewall outbound/egress (not inbound) rules to:

- `keyserver.mainstreetsoftworks.com` (23.182.192.50) TCP port 443
- `www.monetra.com` (23.182.192.20) TCP port 443

The online installation method is the default in GUI mode. After entering the appropriate username and password, hitting the Next button will send the installation online:



For the console installation, choosing the "I" option for Internet will then bring you to the username and password prompt for online mode:

```
File Edit View Search Terminal Help
test@test ~ $ sudo /usr/local/monetrainstaller/bin/monetrainstaller
Welcome to the Monetra Installer v1.1.3!

(I) perform internet installation
(O) perform offline installation
(A) advanced (proxy settings)
(X) exit

Please choose an option
[]: i

Please enter your username (usually your e-mail) for authentication
[]: 
```

3.4.2 Offline Installation (Alternative)

3.4.2.1 Generating An Offline Installation Package

For an offline installation, you will generate and download the package online, but you will not need Internet connectivity for the actual installation process. Generally, one should use the online method (as detailed above), but there can be specific requirements that necessitate the use of an offline install, like not wanting to open a firewall to Monetra's servers, bandwidth constraints, or corporate rules.

Go to <https://www.monetra.com/> and click on "Log In" on the top right menu, then enter your username and password for your client account/license. Once logged in, select "Offline (monetra installer) packages can be generated here." in the bottom right-hand corner.

Client Interface

[Edit My Information](#)[Change Password](#)

Company

Monetra Technologies - Test
9310 Old Kings Rd S; Unit 1401
Jacksonville, FLORIDA 32257

Contact

Brad
info@monetra.com
800-650-9787

Orders

You don't currently have an open order.

My Licenses

[Purchase a New License](#)

Show entries

Filter:

<input type="checkbox"/> Select	ID	Description	System	Created	Expires
<input type="checkbox"/>	72487	Engine - Enterprise (new) Upgrade	Windows Vista/2008/7/8/2012/10 64bit (x64)	11/18/2019 17:06	11/18/2020 17:06

Showing 1 to 1 of 1 entries

Previous **1** Next

[Renew Selected License\(s\)](#)

Downloads

Monetra® Installer

Operating System	Download
Windows Vista/2008/7/8/2012/10 64bit (x64)	Download

Other Downloads

- All Monetra modules may be found [here](#).
- Offline (monetra installer) packages can be generated [here](#).
- Offline license files ([monetra.lic](#)) can be generated [here](#).

The next screen will show a list of available licenses. Select the licenses you wish to include in the package. Then you need to also select the version you want in the package. If performing an upgrade, you will choose the current version you have installed as the Previous version, and the version you want to upgrade to as the Target version. You will then select the file output format (typically Zip on Windows and MacOS and TarBall Gzipped on Linux/Unix).

Generate Offline Package

<input type="checkbox"/> Select	License #	Product	Previous Version	Target Version	Operating System	Ext
<input checked="" type="checkbox"/> Select all licenses in this group			Generate installation ISO for this group			
<input checked="" type="checkbox"/>	72491	Monetra Installer - Package	N/A	1.1.3	windows-x64	
<input checked="" type="checkbox"/>	72490	Monetra Manager - Package	N/A	8.12.0	windows-x64	
<input checked="" type="checkbox"/>	72489	Monetra Admin - Package	N/A	8.12.0	windows-x64	
<input checked="" type="checkbox"/>	72488	Monetra Client - Package	N/A	8.12.0	windows-x64	
<input checked="" type="checkbox"/>	72487	Engine - Enterprise (new)	N/A	8.12.0	windows-x64	

File Format:

Finally, the offline package will be created and provided for download. Please download and transfer this file to the target machine that will have the licenses installed.

3.4.2.2 Extracting The Offline Installation Package

The package that is generated and downloaded is a compressed archive. It must be extracted before it can be used by the Monetra Installer.

On Windows, a Zip file should have been generated. Right-click on the file and choose to extract. Pay close attention to where this extraction occurs. Generally, it will also create a sub-directory named `offline_licenses` with the contents necessary for use by the Monetra Installer.

On MacOS, a Zip file is most typically used. Simply double-clicking on the file will extract the archive to the same directory as the Zip file. Generally, it will also create a sub-directory named `offline_licenses` with the contents necessary for use by the Monetra Installer.

On Linux or Unix systems, either zip or a gzipped tarball may be used. Extract it using one of the below commands:

```
unzip package.zip
```

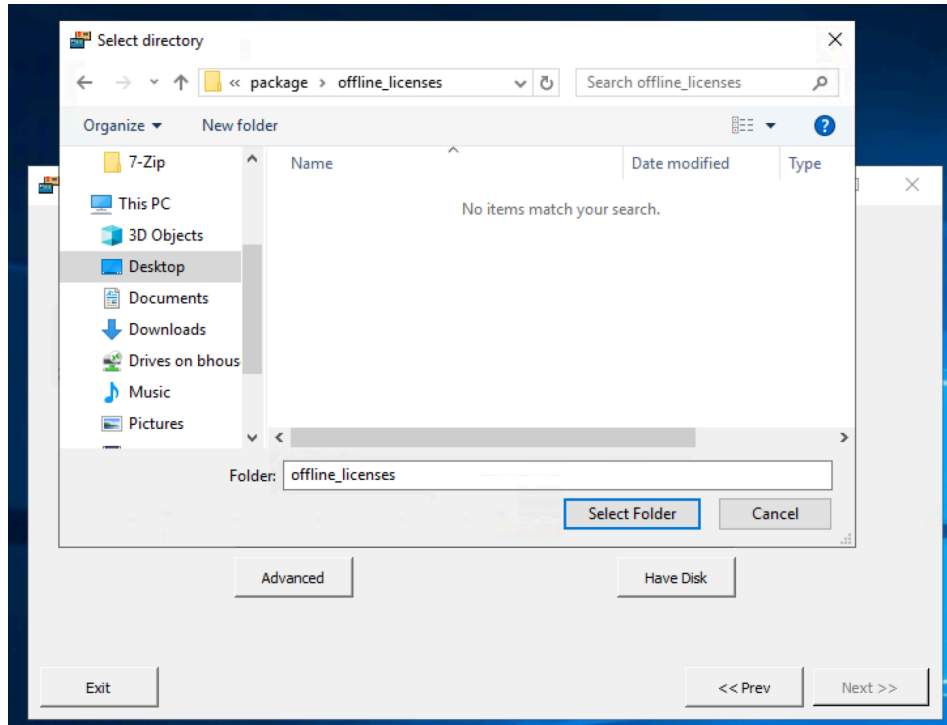
or

```
tar -zxvf package.tar.gz
```

Generally, it will also create a sub-directory named `offline_licenses` with the contents necessary for use by the Monetra Installer.

3.4.2.3 Selecting Offline Installation

When the Installer is run in GUI mode, the ability to use an offline package is accessed through the "Have Disk" button. Simply browse to the `offline_licenses` directory that was extracted in an earlier step and continue. This will bring you to the license list and will otherwise act exactly as an Online/Internet installation will.



When the Installer is run in Console mode, the ability to use an offline package is accessed through the "O" option for "Offline install". The Installer will then prompt for the full path to the `offline_licenses` directory that was extracted in an earlier step. This path should include the `offline_licenses` directory itself. This will bring you to the license list and will otherwise act exactly as an Online/Internet installation will.

```
test@test ~ $ sudo /usr/local/monetrainstaller/bin/monetrainstaller
Welcome to the Monetra Installer v1.1.3!

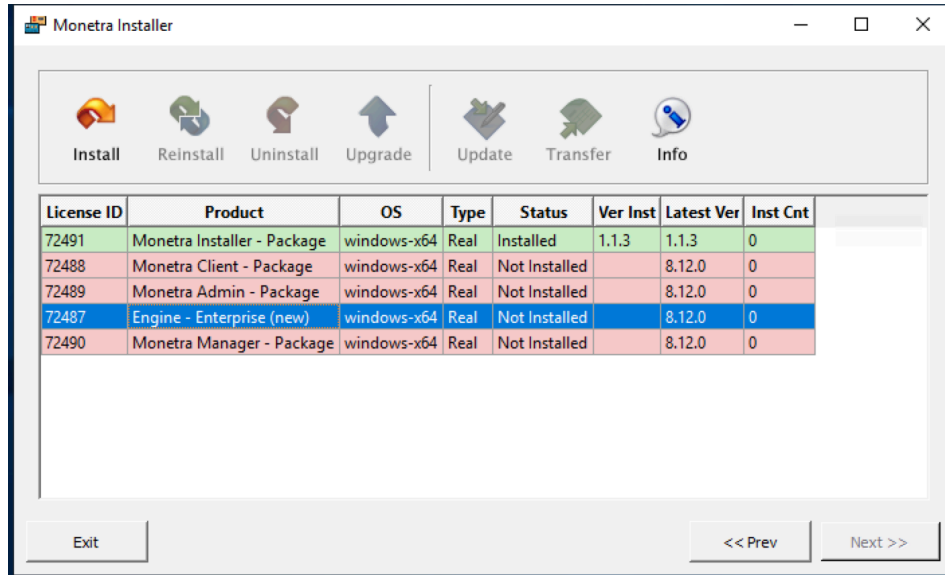
(I) perform internet installation
(O) perform offline installation
(A) advanced (proxy settings)
(X) exit

Please choose an option
[]: o

Please choose a directory where offline_licenses.xml exists
[]: /tmp/offline_licenses/
```

3.5 Interacting With The Monetra Installer

After logging in to the Monetra Installer, you will see a list of licenses. In GUI mode, after you select the license you would like to operate on, the icons at the top of the screen will light up for the available options for that license. In Console mode, you will type in the license number from the provided list for the license you'd like to operate on.



ID	PRODUCT	OS	TYPE	STATUS	INST	LATEST	I
72496	Monetra Installer - Package	linux-x64	REAL	inst	1.1.3	1.1.3	0
72493	Monetra Client - Package	linux-x64	REAL	-		8.12.0	0
72494	Monetra Admin - Package	linux-x64	REAL	-		8.12.0	0
72492	Engine - Enterprise (new)	linux-x64	REAL	-		8.12.0	0
72495	Monetra Manager - Package	linux-x64	REAL	-		8.12.0	0
72487	Engine - Enterprise (new)	windows-x64	REAL	N/A		8.12.0	0
72488	Monetra Client - Package	windows-x64	REAL	N/A		8.12.0	0
72489	Monetra Admin - Package	windows-x64	REAL	N/A		8.12.0	0
72490	Monetra Manager - Package	windows-x64	REAL	N/A		8.12.0	0
72491	Monetra Installer - Package	windows-x64	REAL	N/A		1.1.3	0

Please choose a license (ID) (or Q to Quit)

[I]: 72492
 (I) : Install
 (F) : Info

Please choose a function to perform (or X to exit)

[I]: █

The operations should be self-explanatory, with the exception of "Update" and "Transfer". The "Update" operation is a license update. It is used to apply an updated license (obtained after purchasing an add-on) to the current installation. This is necessary because Monetra never "calls-home" to perform any sort of automatic license update. The "Transfer" option is used to change which license is installed without touching the installation. For this operation, select the installed license, choose "Transfer", and select the target license.

If you are attempting to install a license but find that the operation is not enabled after selecting the license, there are two possibilities. The first is you already have another license installed for the same product. The other possibility is the OS/Architecture for the license does not match the OS/Architecture of the Installer or system (e.g. subtle differences like a 32bit Windows installer running on 64bit Windows cannot install 64bit Windows licenses, you must use the 64bit Windows installer for that).

If you are attempting to upgrade a license but find that the operation is not enabled after selecting the license, you are most likely logged in to the wrong host, as that host does not have the license installed.



Note: After logging in to the Monetra Installer for the first time, you might be asked if you want to install all packages. This is an automated installation of all your licenses and will only be displayed if you have exactly one of each product type in your license list. Answering *yes* will always install the most current version of each product.

3.6 Installing Monetra

After logging in to the Monetra Installer or using an offline package, select the Monetra Payment Engine license from the list. Select the Install option, then choose the version of Monetra from the list to be installed. Follow any on-screen prompts, such as reviewing the EULA and selecting an installation path. When this completes, Monetra should be installed.

3.7 Installing GUI Helper Utilities

There are various graphical helper utilities that can be installed to aid in configuring and testing Monetra. These utilities are not required, as all tasks can be performed by either editing configuration files or running commands via the API; they are provided as a means of assisting an installation or integration. These graphical utilities are only available on Linux, Windows, and MacOS. All utilities below, with the exception of the Monetra Manager, can be run on a remote system and assigned to a different Operating System than the Monetra Engine license. Please contact support@monetra.com with any license OS change requests for this purpose.

Installation of these utilities is generally the same as installing Monetra and is done through the Monetra Installer. Please find descriptions of the available helpers below.

3.7.1 Monetra Manager

This utility facilitates tasks such as editing the configuration files, starting and stopping Monetra, and viewing log files. It can only be run on the same machine as Monetra.

On Windows and MacOS, the Monetra Manager will prompt for privilege escalation. On Linux, it must be run under `sudo`.

3.7.2 Monetra Administrator

This utility is used to perform Monetra-administrative tasks such as adding merchant accounts, creating administrative-level restricted users, and scheduling automated tasks such as transaction history purging.

3.7.3 Monetra Client

This utility is strictly designed for testing and reporting of merchant-level users. While card holder data can be input via the Client, this feature is strictly meant for testing and development, not production use.

3.8 System Tuning

It is the installer's responsibility to ensure the system Monetra is deployed on is tuned to the desired specifications for anticipated load and reliability. This includes, but is not limited to system scheduler selection, network stack tuning, fault tolerance of both the Monetra system(s) as well as database systems, and system process restrictions (e.g. `ulimits`). It is expected any deployment of Monetra will have qualified system administrators and database administrators (DBAs) involved in the management and running of the system.

The most important system tuning settings for the host where Monetra resides, but is often overlooked, is the total number of network connections allowed. On unix systems this is often the number of allowed open file descriptors. For Windows, there are similar settings in the registry controlling maximum user ports. It is important to know during deployment the anticipated number of simultaneous connections and tune appropriately.

3.8.1 Linux Host Tuning Example

Below you will find an example of tuning a Linux host. This is not to be treated as a complete/exhaustive example, or even a recommendation, but simply outlines the thought process when approaching the tuning of a system for Monetra. The below is known to work on RHEL/CentOS based systems, but may not be right for your system even if using RHEL/Centos.

3.8.1.1 Maximum Connections/File Descriptors

To increase the system-wide maximum file descriptors, create a file named `/etc/sysctl.d/maxfiles.conf`:

```
# Increase maximum number of file descriptors
fs.file-max = 943718
```

Monetra-specific settings for limitations to increase file descriptor count as well as allowed thread count can be done via a `systemd` configuration file. Create a file named `/etc/systemd/system/monetra.service.d/limits.conf` with these contents:

```
[Service]
LimitNOFILE=65536
LimitNPROC=65536
```

Once the number of file descriptors Monetra is allowed to use has been raised, the `COMM_max_conns` configuration in Monetra's `prefs.conf` can be increased. It is required to reserve some file descriptors for usage outside of inbound connections, so the value chosen should be less than the limitation set. Monetra uses additional file descriptors for things like database connections, logging, and outbound connections to processors.

3.8.1.2 Network Tuning

When using an external database, Monetra uses the 3rd party library provided by the database vendor for connections. Since Monetra doesn't have direct access to the network connection, it cannot control things like TCP Keep-alive for swift notification of a dropped connection.

Most database libraries do enable TCP Keep-alive support, however rely on the host OS to be properly tuned, but the Linux defaults can take 15 minutes or more to detect some types of failures which is not suitable for a production deployment.

To tune TCP keep-alive to detect broken connections within 30s, you can create a file named `/etc/sysctl.d/keepalives.conf`:

```
# Time before keepalives start
net.ipv4.tcp_keepalive_time = 120
# Keepalive attempts before failure
net.ipv4.tcp_keepalive_probes = 3
# Interval of probes
net.ipv4.tcp_keepalive_intvl = 10
```

The default linux network stack is tuned for memory efficiency and many of the defaults are decades old and have not been tuned for modern networks. For instance it may not evict half-closed connections quickly, be subjected to buffer bloat, or tuned for legacy 10Mb/s networks rather than modern 10Gb/s networks. Create `/etc/sysctl.d/nettune.conf`:

```
# Increase TCP max buffer size setable using setsockopt()
# default 4096 87380 4194304
net.ipv4.tcp_rmem = 4096 87380 8388608
net.ipv4.tcp_wmem = 4096 87380 8388608

# Increase Linux auto tuning TCP buffer limits
# min, default, and max number of bytes to use
# set max to at least 4MB, or higher if you use very high
# Bandwidth-delay product (BDP) paths Tcp Windows etc
# default 131071
net.core.rmem_max = 8388608
net.core.wmem_max = 8388608
# default 124928
net.core.rmem_default = 524287
net.core.wmem_default = 524287

# default net.core.netdev_max_backlog = 1000, set to 30000 for 10Gbit NICs
net.core.netdev_max_backlog = 32768
#
# default net.core.somaxconn = 128
net.core.somaxconn = 4096

# You might also try the following:
# default net.ipv4.tcp_max_syn_backlog = 1024
net.ipv4.tcp_max_syn_backlog = 4096

# default net.ipv4.ip_local_port_range = "32768 61000"
net.ipv4.ip_local_port_range = 9000 65535

# default tcp_fin_timeout = 60
net.ipv4.tcp_fin_timeout = 30

# This allows reusing sockets in TIME_WAIT state for new connections
# when it is safe from protocol viewpoint
net.ipv4.tcp_tw_reuse = 1

# Enable the BBR congestion control algorithm (requires 4.9+ kernel)
net.core.default_qdisc=fq
net.ipv4.tcp_congestion_control=bbr
```

4 Configuration

4.1. Logging	24
4.1.1. File Logging	25
4.1.2. Syslog	26
4.2. Database	26
4.2.1. Driver: MySQL	28
4.2.2. Driver: Oracle	29
4.2.3. Driver: ODBC And DB2	29
4.2.4. Driver: PostgreSQL	30
4.2.5. Driver: SQLite	30
4.3. Key Management For Data At Rest	30
4.3.1. Protection Mechanisms	30
4.3.2. Key Generation	34
4.4. Starting And Stopping Monetra	35
4.4.1. Post-Startup Activation Requirements	35
4.5. Mail Configuration For Notices, Settlement Summaries, MFA, and Receipts	35
4.5.1. SMTP Direct Send	36
4.5.2. External Mailer	36
4.6. SMS configuration for MFA, Password Resets, Receipts	37
4.7. Clustering	37
4.8. Advanced Configuration	37
4.9. Creating Initial Administrator User(s)	38
4.10. Activating Processing Institutions	39
4.11. Configuring Data Retention	40

This section describes the primary configuration options that most integrators will modify during installation of Monetra. It is not a complete listing of all available options. Modifying these settings should be sufficient for optimal operation of Monetra. Most other settings are generally left at their default values.

Monetra is pre-configured to be secure by default and follow the PCI-DSS rules. Please use caution when changing any settings not listed here away from their default values, as this can severely affect the performance and security of Monetra. When in doubt, please contact support@monetra.com with a description of what you are trying to accomplish for guidance.



Note: All configuration changes made in the Monetra Manager or via the configuration files require a restart of Monetra in order to take effect.

4.1 Logging

The Monetra log is a multi-purpose log format. It provides insight into the operations of Monetra and a detailed audit trail of every operation performed by an end user. This includes, but is not limited to, IP address, request types along with request and response parameters, and duration of connection.

The Monetra log will never log sensitive merchant or cardholder data to disk; any data that might be considered sensitive is completely masked in the logs. Monetra does contain the

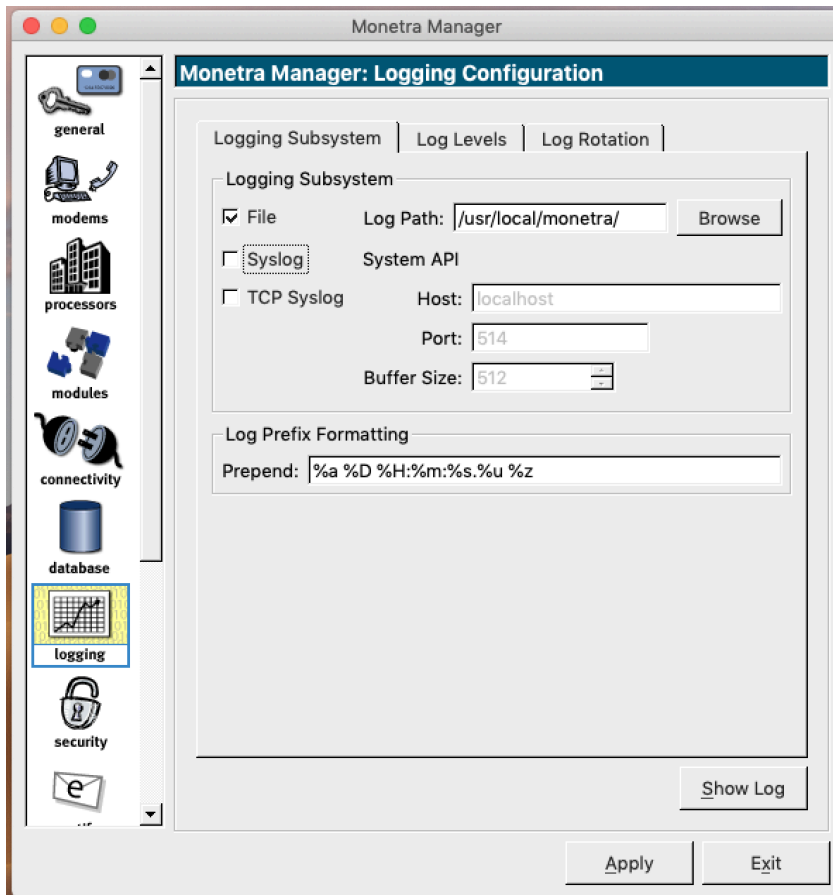
ability to create a debug logging session to view sensitive information for troubleshooting purposes, but it does not provide a means to write that data to disk; it is an in-memory view only and only accessible with Administrative-level authentication. Such logging can be accessed via the Monetra Manager or the included `monetra_setdebug` utility. Care must be taken to never capture such data to disk or other permanent storage. It is meant only for real-time evaluation.

Monetra facilitates centralized logging via remote syslog facilities, either by using a system-provided syslog API, or by directly supporting Syslog over TCP. See the available configuration options below.

PCI PCI Notice: PCI DSS mandates that logging of all modifications to system-level objects is done on systems containing payment applications. It is required that merchants deploy a logging facility such as file-integrity monitoring on such payment systems to comply with this requirement. Monetra does not provide such a facility directly; this is an external requirement that must be implemented by the merchant.

4.1.1 File Logging

File logging is the default logging option in Monetra. It logs to the Monetra installation directory by default with a filename of `monetra.log`. The settings that control file logging are available in `main.conf`, or via the Monetra Manager. The primary settings are `debugsys=FILE` and `debugdir=[path]`.

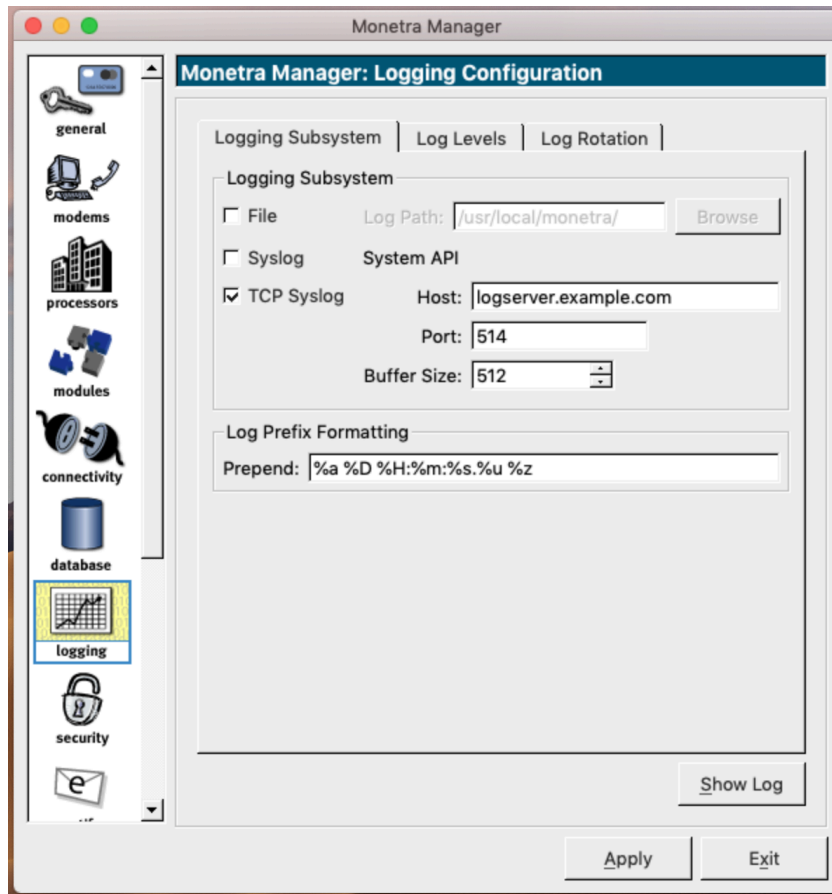




PCI Notice: When using file logging, it is the responsibility of the integrator to ensure that logs are transported to a centralized log server. Options such as `logstash` and `splunk_forwarder` are often used for this purpose.

4.1.2 Syslog

Syslog is a common Unix centralized logging system. All logs are sent to the syslog service in order to be routed by the syslog configuration to the proper endpoint. The settings that control syslog are available in `main.conf`, or via the Monetra Manager. The primary settings are `debugsys=SYSLOG` or `debugsys=TSLOG`. `SYSLOG` uses the OS-provided syslog APIs, while `TSLOG` is an internal implementation that uses the standard TCP Syslog communication protocol to send to a remote syslog server. The latter can be used on systems without native syslog support, such as Windows. When `TSLOG` is specified, `debug_ts_host=[host]` and `debug_ts_port=[port]` must also be specified.



4.2 Database

Monetra supports data storage to just about any SQL database backend, which is facilitated by the `mstdlib` open-source library available at <https://github.com/Monetra/mstdlib>. All sensitive data is encrypted prior to hitting the database abstraction layer using field-level encryption.

When using an external database (e.g. anything other than SQLite), it is required to set up a dedicated database or namespace with a dedicated user for use by Monetra. Monetra should be granted all privileges on the database or namespace, such as table creation or deletion, index creation or deletion, table insert, update, and delete, and possibly additional functionality like creation of procedures and views.



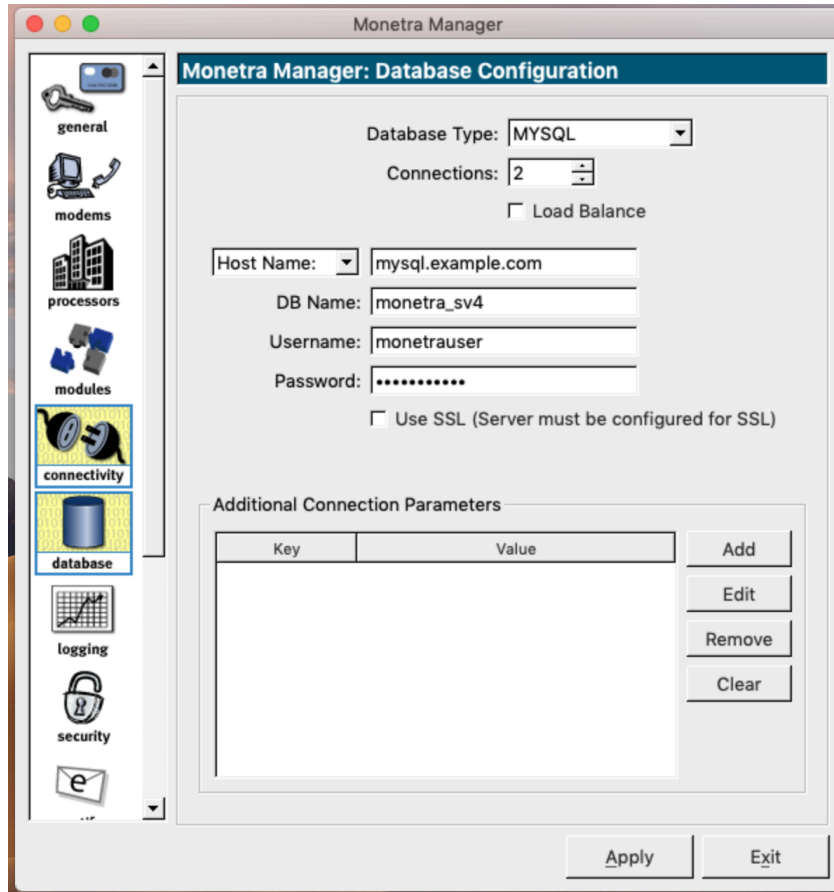
Note: It is the responsibility of the installer or merchant to provide the necessary DBA resources to manage the database in use.



PCI Notice: It is up to the integrator/installer to ensure all access to the Monetra database is logged as per the PCI-DSS requirements.

There are a few main configuration parameters in `prefs.conf` necessary to configure the database in Monetra.

- `SQL_driver` - The name of the driver (database) being used. See the sections below.
- `SQL_conn_str` - A driver-specific connection string consisting of semi-colon separated configuration options.
- `SQL_username` - The username to authenticate against the database
- `SQL_password` - The password to authenticate against the database
- `SQL_max_conns` - The maximum number of simultaneous connections to the database



4.2.1 Driver: MySQL

The MySQL driver can be used to connect to any MySQL derivative such as MariaDB, Percona XtraDB, or similar.

Connection String Fields:

- `db`: Required. Database Name.
- `socketpath`: Conditional. If using Unix Domain Sockets to connect to MySQL, this is the path to the Unix Domain Socket. Use the keyword of 'search' to search for the socket based on standard known paths.
Cannot be used with `host`.
- `host`: Conditional. If using IP or SSL/TLS to connect to MySQL, this is the hostname or IP address of the server. If not using the default port of 3306, may append a `":port#"` to the end of the host. For specifying multiple hosts in a pool, hosts should be comma-delimited. E.g:
`host=10.40.30.2,10.50.30.2:13306`
Cannot be used with `socketpath`.
- `engine`: Optional. Used during table creation, defaults to `INNODB`. The default data storage engine to use with `mysql`. Typically it is recommended to leave this at the default.
- `charset`: Optional. Used during table creation, defaults to `utf8mb4`.
- `max_isolation`: Optional. Sets the maximum isolation level used for transactions. This is used to overwrite requests for `SERIALIZABLE` isolation levels, useful with Galera-based

clusters that do not truly support `Serializable` isolation. Should use "SELECT ... FOR UPDATE" type syntax for row locking. Available settings: "REPEATABLE READ", "READ COMMITTED"

4.2.2 Driver: Oracle

Oracle versions 11.2 or higher are supported.

Connection String Fields:

- `dsn`: Conditional. Data Source Name as specified in `tnsnames.ora`, or a fully qualified connection string. If not specified, both `host` and `service_name` must be specified to dynamically generate a connection string. Use of this parameter negates the ability to use load balancing and failover logic but facilitates the use of Oracle's equivalent functionality. An example of a fully qualified connection string would be:

```
(DESCRIPTION =
(ADDRESS = (PROTOCOL = TCP)(Host = 10.100.10.168)(Port = 1521))
(CONNECT_DATA = (SERVICE_NAME = orcl))
)
```

- `host`: Conditional. If `dsn` is not specified, this parameter must be specified along with the `service_name` parameter. This is the hostname or IP address of the server. If not using the default port of 1521, may append a " :port#" to the end of the host. For specifying multiple hosts in a pool, hosts should be comma-delimited. E.g:
`host=10.40.30.2,10.50.30.2:11521.`
Cannot be used with `dsn`.
- `service_name`: Conditional. If `dsn` is not specified, this parameter must be specified along with the `host` parameter. E.g: `service_name=orcl`
Cannot be used with `dsn`.

4.2.3 Driver: ODBC And DB2

ODBC is a database abstraction layer, where drivers plug into the ODBC layer provided by the operating system and provide a common interface. Most databases offer ODBC drivers, but it is often recommended to use the native Monetra driver rather than the ODBC driver for performance and the best feature support. The ODBC system also requires system configuration to set up the database before it can be referenced by Monetra. This is left as an exercise to the installer as if this option is chosen existing expertise is expected.



Note: Microsoft SQL Server always uses ODBC as there is no non-ODBC native driver provided by Microsoft. The ODBC driver depends on features introduced in SQL Server 2008 and higher.

Connection String Fields:

- `dsn`: Required. Database Source Name.
- `mysql_engine`: Optional. The default data storage engine to use with `mysql`. Used during table creation when the underlying database is `MySQL`. Defaults to `INNODB`. Typically, it is recommended to leave this at the default.

- `mysql_charset`: Optional. Used during table creation when the underlying database is MySQL. Defaults to `utf8mb4`.

4.2.4 Driver: PostgreSQL

The PostgreSQL driver depends on features introduced in v9.5 and higher.

Connection String Fields:

- `db`: Required. Database Name.
- `host`: Required. This is the hostname or IP address of the server. If not using the default port of 5432, may append a `":port#"` to the end of the host. For specifying multiple hosts in a pool, hosts should be comma-delimited. E.g: `host=10.40.30.2,10.50.30.2:15432`
- `application_name`: Optional. Application name to register with the server for debugging purposes.

4.2.5 Driver: SQLite

SQLite is a local on-disk database format, and the SQLite driver contains the entirety of the database layer. There is no connection to an external process or server and no authentication or access control subsystem available in SQLite. This database type is only suitable for small deployments for private/trusted networks.



PCI Notice: It is up to the integrator/installer to ensure the proper permissions are applied to the SQLite database path so that only system users with sufficient privileges are allowed to access the database. It is also required that all accesses to the database are logged via the OS's audit controls.

Connection String Fields:

- `path`: Required. File system path to SQLite database.
- `journal_mode`: Optional. Defaults to `WAL` if not specified. Another option is `DELETE`.
- `analyze`: Optional. Defaults to `true` if not specified. On first connect, automatically runs an analyze to update index statistics if set to `true`.
- `integrity_check`: Optional. Defaults to `false` if not specified. On first connect, automatically runs an integrity check to verify the database integrity if set to `true`.
- `shared_cache`: Optional. Defaults to `false` if not specified. Enables shared cache mode for multiple connections to the same database.

4.3 Key Management For Data At Rest


4.3.1 Protection Mechanisms

Key Management and Protection is the most important security-impacting configuration decision that is made during an installation of Monetra. How you protect the key is the

cornerstone of ensuring the system remains secure and that an adversary cannot compromise cardholder data. The below sections will detail the options. With PassPhrase being the default protection mechanism as of v8.13 and HSM being the most secure.

On startup, Monetra validates the encryption key used can be used to decrypt a data field that was encrypted when first created. If this data field cannot be decrypted, then Monetra will fail to start due to detecting an inappropriate key substitution.

For all defined protection mechanisms, what is being protected is a Key Encryption Key (KEK) which is used to protect the actual data encryption keys. It is recommended that installers take into consideration the relatively small number of keys protected by the KEK when determining the appropriate crypto period.

 **PCI Notice:** For all key custodians involved in key management procedures with Monetra, an executed key custodian agreement that confirms the key custodian acknowledges and accepts their responsibilities is required to be kept on file. A sample key custodian agreement is attached in [Appendix A: \[SAMPLE\] Cryptographic Material Custodian Agreement](#).

It is also required that access to keys is restricted to the fewest number of custodians necessary and keys be stored securely in the fewest possible locations.

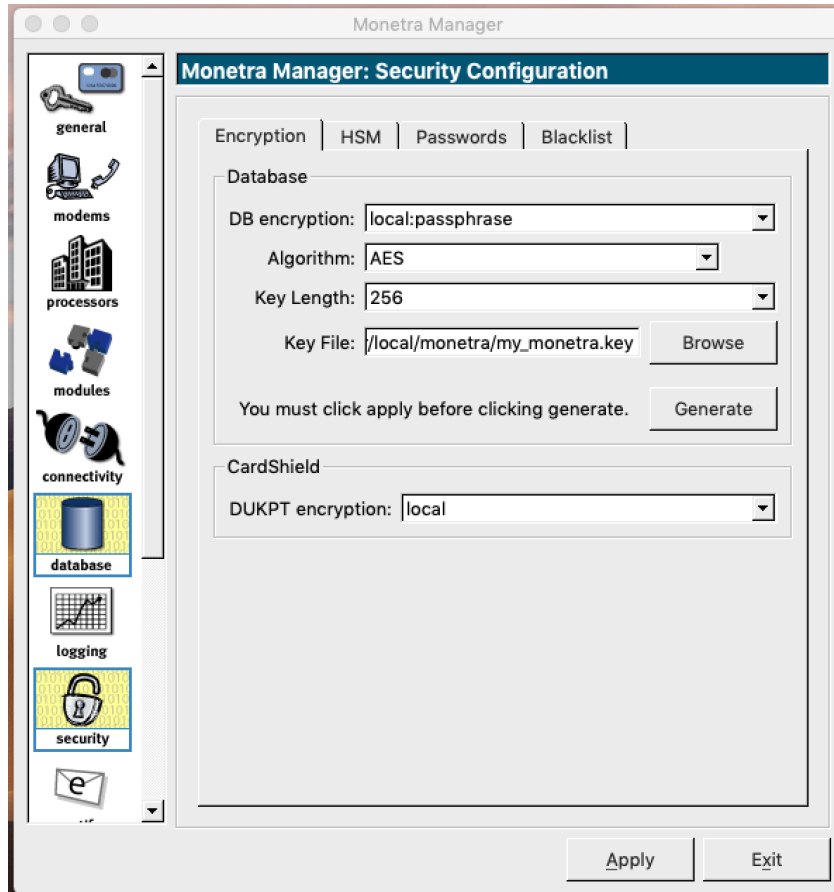
No key protection mechanism ever returns the plain text key to a custodian for use outside of Monetra. It unwraps/decrypts the key in volatile memory for use by Monetra.

4.3.1.1 Passphrase

Passphrase key protection is used to cryptographically secure the data encryption key using one or more passwords. When more than one password is used, the key is split using Shamir's secret sharing and can be split using an M of N ratio, meaning M passwords are required out of a total of N when the key was created. When M is greater than 1, this facilitates strong protections with split-knowledge/dual-control.

This is the recommended and default key protection mechanism for Monetra, as it provides a cryptographic means of unlocking the data encryption key. The number of passphrases used is prompted for during creation of the key.

```
main.conf : dbencrypt= Setting is local:passphrase - Do NOT enter the passphrase here; use the literal string 'passphrase'
```



If passphrases need to be rotated in the future, the `monetra_keygen` command line utility should be used as referenced in [Section 4.3.2: Key Generation](#). Simply passing the `--re-key` option will prompt for the required steps.



Note: It is critically important the passphrase used be kept secure and not lost as if insufficient passphrases are available there is no recovery possible. Only those individuals with a need should have access to the necessary passphrase to unlock the database encryption key. It is recommended that each individual have their own passphrase, but it is allowable to share a single passphrase with multiple people. It is, however, required that upon departure of an individual from the assigned group of key custodians that the key passphrases be updated accordingly.

There is no standardized/enforced passphrase strength (other than being at least 7 characters) or rotation policy, however it is recommended that the deploying company provide guidance as per their own business requirements.


4.3.1.2 HSM

An HSM is a Hardware Security Module that is responsible for securely storing key material in both a logically and physically tamper-resistant hardware module. Ideally, an HSM should be validated against the FIPS-140-2 Level 3 standard and be configured to operate in this

mode when used in Monetra. Indeed, such is required if a private label PCI P2PE validation is desired.

Since the key material is secured by an externally validated entity, it is considered the most secure storage mechanism. However, due to cost, high availability requirements, and performance reasons, they are often not deployed.

For performance reasons while protecting data, the HSM is used for key wrapping and not direct data encryption and decryption. The data encryption key is loaded into the volatile memory of the host where Monetra resides. For P2PE data received from a device, however, the HSM handles 100% of the data decryption and the Monetra host never has access to the key material.

 **PCI Notice:** It is recommended that multiple key custodians are used if a PIN is required to unlock the HSM, and that the PIN be split into left and right halves and shared across multiple custodians such that no custodian has both halves.

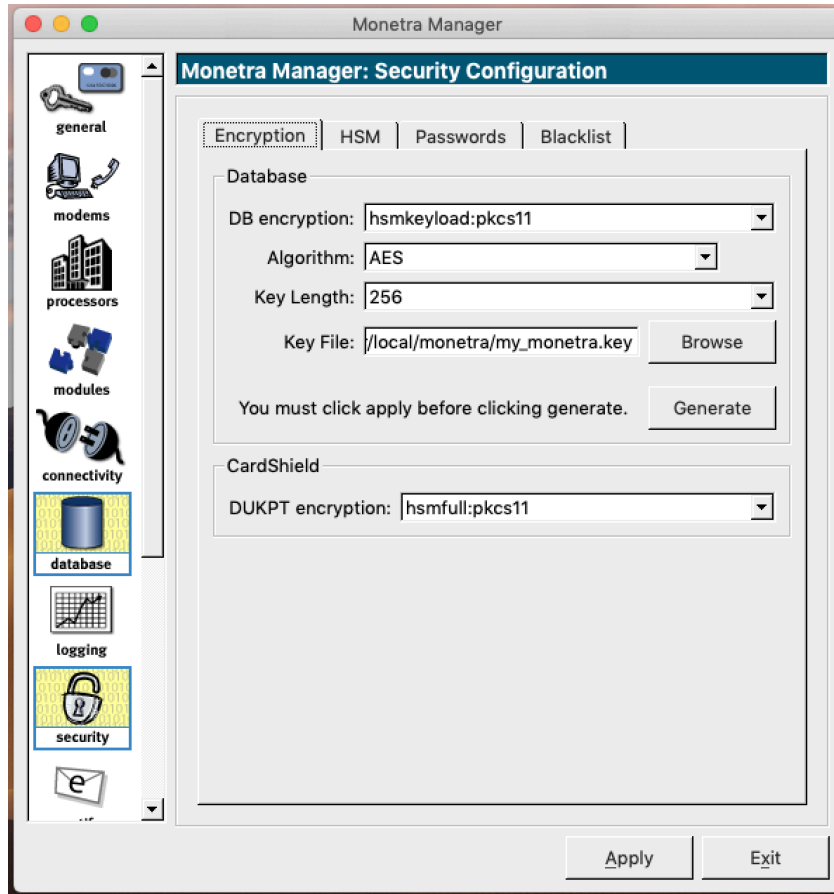
4.3.1.2.1 Configuration Options

Configuration for HSMs requires modification to `main.conf` and `modules.conf`.

`main.conf` : `dbencrypt=hsmkeyload:[module]` where `[module]` can be either `pkcs11` or `nfast` for a Cryptoki/PKCS11 compliant HSM or Thales NFast HSM, respectively.

`modules.conf` : `loadmodule=` The module to load for HSM support. Can either be `crypto_pkcs11.so` or `crypto_nfast.so`

`prefs.conf` : `pkcs11_pin_split_knowledge=` Available settings are `true` and `false`. When enabled, will prompt for left and right halves of the HSM PIN to facilitate split-knowledge/dual-control protections.



4.3.1.3 Local

Local key protection is not recommended. Check with your PCI Auditor for approval and ways to better secure this mechanism, such as through the use of Full Disk Encryption. During key generation, the KEK is encrypted using an RSA public key. During key usage, a mathematically derived symmetric key is computed, then used to decrypt an RSA private key and decrypt the KEK. This means all Monetra instances share the same RSA private key to protect the KEK while at rest and as such is a limit to the security of this mechanism.

```
main.conf : dbencrypt= Setting is local:internal
```

4.3.2 Key Generation

Key Generation can be performed in two ways: either via the Monetra Manager in the same place where the key is configured, using the "Generate" button, or via the command line using the `monetra_keygen` utility in the `bin` subdirectory of the Monetra installation and requires Administrative or `root` privileges. If necessary, both utilities will perform a guided process of generating the key and possibly asking for things such as number of passwords to protect a key with or the PIN to unlock an HSM.

The `monetra_keygen` can typically be called from the installation path, such as:

```
sudo /usr/local/monetra/bin/monetra_keygen
or on Windows via an Administrator cmd.exe shell:
C:\Program Files\Main Street Softworks\Monetra\bin\monetra_keygen.exe
```

4.4 Starting And Stopping Monetra

Monetra is pre-configured to automatically start at boot. It can be started and stopped via the Monetra Manager or the operating system's service management. On Windows, `net start monetra` can be used from the command line; for recent versions of Linux, `systemctl start monetra` can be used; and on older versions of Linux or other Unix systems, `service monetra start` or possibly a utility installed into the system path called `monetrad` can be used.

4.4.1 Post-Startup Activation Requirements

When using Passphrase key protection or an HSM, there is often data that must be sent to Monetra in order to unlock the KEK or to authenticate access to an HSM. A Secure Crypto Channel utility is provided to pass this information to Monetra during the startup process. The utility can be run from the local machine or remotely across a private network, as all communication is secured via TLS.

If Monetra was started using the Monetra Manager, the Manager will automatically call out to the Secure Crypto Channel utility and perform any necessary prompting. Otherwise, if Monetra was started at boot or via the system's service manager, then it might be necessary (if configured to do so) to finish the activation manually using this utility. The utility is named `monetra_scc` and is found in the Monetra installation path's `bin` directory, such as `/usr/local/monetra/bin/monetra_scc`. By default, the utility tries to connect to the local machine and, if the secure crypto channel is running, will perform the requested prompts. More than one simultaneous user may use the utility from different sessions if there are multiple passphrases necessary to unlock a key.

4.5 Mail Configuration For Notices, Settlement Summaries, MFA, and Receipts

It is important that Monetra be configured to be able to send email. Emails are used to notify system administrators about significant events (start, stop, failover, license usage) and to facilitate other notifications such as settlement summaries, MFA codes, Receipts, and P2PE device event notifications.

There are a couple of methods of configuring email support, but all methods require an external mail server or MTA for Monetra to communicate with. The first steps in both methods consist of configuring the destination address and source address. If editing the configuration files, this is in `main.conf` under `email_to` and `email_from`, respectively.



4.5.1 SMTP Direct Send

Monetra can connect to a remote SMTP server via the standard SMTP protocol. Both unencrypted and TLS-encrypted (but not *StartTLS*) communication are supported, as are CRAM-MD5, PLAIN, and LOGIN authentication methods.

To configure SMTP support, edit `main.conf` (or use the Monetra Manager), setting `MailNotices=SMTP` and then `SMTP_host=` and `SMTP_port=` to the appropriate values for the mail server being used. If TLS encryption is required, set `SMTP_TLS=yes`. If authentication is needed, set both `SMTP_authuser=` and `SMTP_authpass=` as well.

4.5.2 External Mailer

On some systems, it may be desirable to use an external mailer such as `sSMTP` or `mSMTP` to send emails from Monetra. A benefit of using an external mailer is the addition of enhanced features (e.g. *StartTLS* support or additional authentication methods). Monetra will reach out to an executable on the local system to submit emails. The executable must support the standard `sendmail` syntax and receive the mail message via `stdin`. Most mailers support this and provide a suitable executable or symlink in `/usr/lib/sendmail`.

To configure external mailer support, edit `main.conf` (or use the Monetra Manager), and set `MailNotices=EXEC`, and then set `sendmail=` as appropriate. In most cases, `sendmail=/usr/lib/sendmail -t` is the appropriate setting to use.

4.6 SMS configuration for MFA, Password Resets, Receipts

SMS, also known as text messaging, is used by Monetra for MFA, Password Resets, and sending of receipts. Currently the only supported SMS provider is **Twilio** [<https://twilio.com>]. The only currently supported authentication method with Twilio is by using an `Account SID` with `Auth Token`.

There are 4 configuration parameters which must be set within `main.conf` to enable messaging:

- `sms_provider=twilio` - Set provider to Twilio
- `sms_from=` E.164 formatted phone number SMS will originate from. e.g. +19041234567, must be linked to account.
- `twilio_sid=` Twilio-assigned Account SID
- `twilio_auth_token=` Twilio-assigned Authentication Token

4.7 Clustering

Clustering in Monetra requires the purchase of cluster licensing on a per-node basis. Once licensing is purchased, Monetra's clustering capabilities rely exclusively on the Database layer. Simply pointing multiple instances of Monetra that are installed on the same OS to the same installation path on their respective nodes to the same database will activate clustering. It is recommended to sync configuration (`*.conf`) files and copy the database encryption key so they are the same on each node.



Note: When clustering is enabled, `action_admin=ticketrequest` support is not cluster-aware. This is used for services such as `PaymentFrame` and `DirectPost` methods of integration. In these cases, it is recommended that Monetra be run in a fail-over environment to offer those services. All other services are fully load-balance capable.

4.8 Advanced Configuration

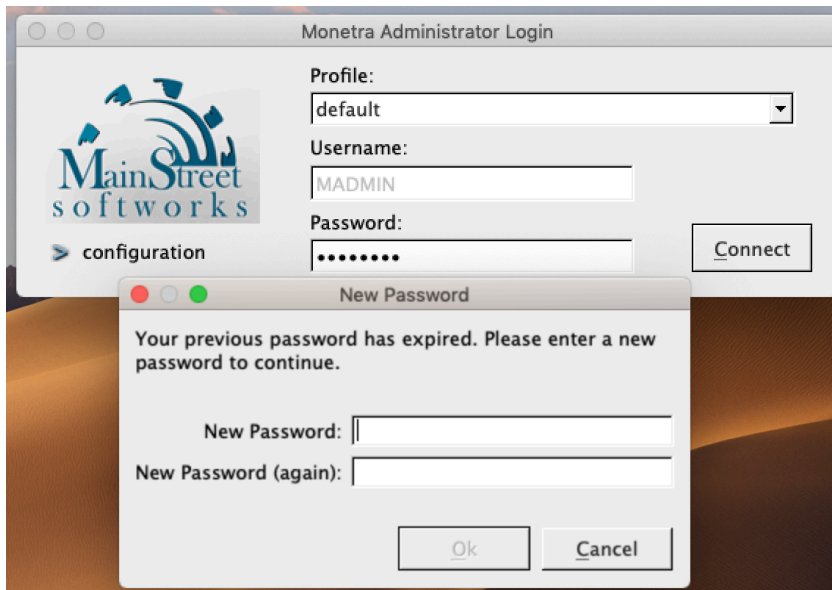
There are dozens of additional configuration options spread across the 4 main configuration files: `main.conf`, `prefs.conf`, `modules.conf`, and `processors.conf`. These configuration files follow the standard `ini` configuration file format and are heavily commented/documented internally. It is recommended that none of the settings that are not mentioned in this document be modified until a support ticket has been opened at support@monetra.com for guidance on your specific needs.

The configuration paths for Windows are the `etc` subdirectory of the Monetra Installation path, by default `C:\Program Files\Main Street Softworks\Monetra\etc`. The configuration paths for Linux and Unix systems are `/etc/monetra`.

4.9 Creating Initial Administrator User(s)

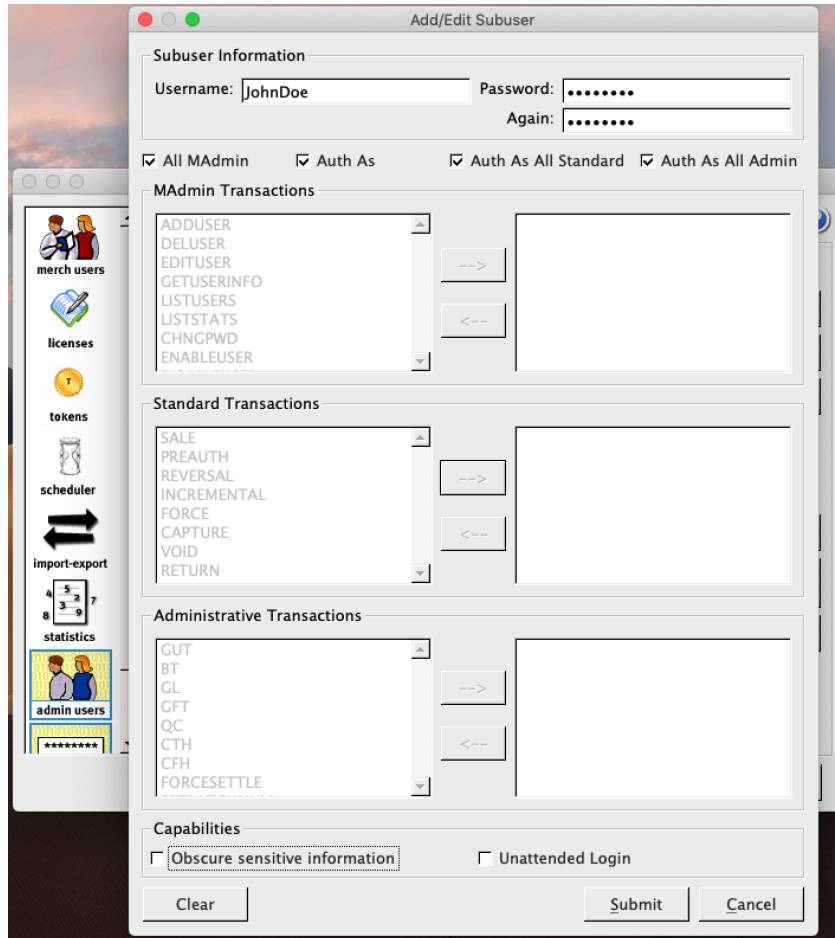
When Monetra initializes its database for the first time, it creates a default user of `MADMIN` with a password of `password`. This default password is marked as expired when created and can only be used to change the password; all other operations are prohibited and enforced.

If using the Monetra Administrator, on first login you will be automatically prompted to set the new `MADMIN` password.



If using the API, the `action_sys=USER_PASSWD` command would be used for this purpose.

Once logged in, it is now time to create an Administrator user for each person to whom you wish to grant that level of access. A unique username and password should be created for each individual; there should be no shared accounts. The username for the `MADMIN` account may be change to reflect the name of the individual with the highest level right. It is required that at least one administrator be granted `ALL` privileges. Additional superuser accounts may be created. If using the Monetra Administrator, this can be done under the `admin users` section. If using an API, the `action_sys=USER_ADD` request would be used for this purpose.



Note: Monetra enforces that passwords are at least 8 characters, and contains at least one uppercase letter, one lowercase letter, one number, and one special character.

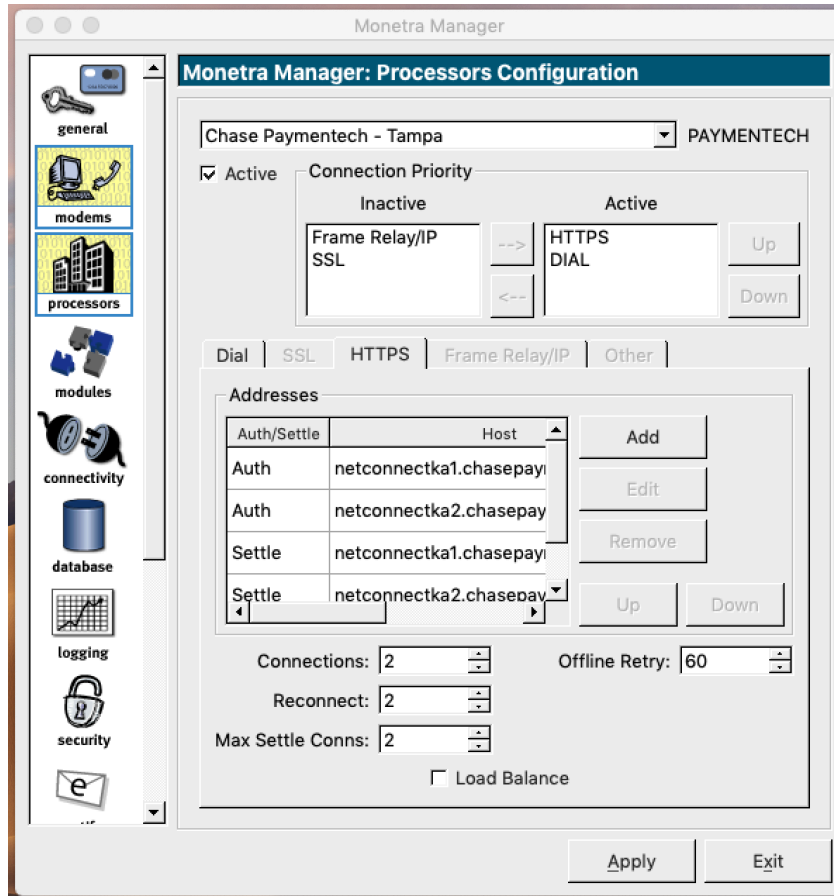


PCI Notice: The initial MADMIN password set at this point should either be destroyed or stored in a safe for future recovery purposes.

4.10 Activating Processing Institutions

By default, Monetra ships with all processing institutions set to a deactivated state in order to save on system resources, as each active processor will cause additional threads to spawn at startup and consume CPU and memory resources. Simply enabling the processing institutions to be used for the integration is all that most deployments need. Most processing institutions support transmittal over the Internet, and Monetra comes pre-configured with the latest URLs required to process transactions.

Processors can be enabled in the Monetra Manager under Processors, using the drop-down of available processors. Please note that Monetra must be running for this list to populate, as queries are sent to Monetra to obtain the complete list. It is recommended to perform this step after setting up the Administrator users, because the Monetra Manager will prompt for a username and password.



Processors can also be enabled via editing the `processors.conf` file. The file is laid out in sections with the processor name enclosed in brackets `[name]`, as is typical for `ini` files. Under each relevant section, setting `active=yes` will enable the given processor.



Note: If using a processing institution via a private circuit (e.g. MPLS, VPN) or attempting to perform settlements with "Big Batch" processors, please contact support@monetra.com for assistance.



Note: Reminder: Monetra must be restarted for changes to take effect.

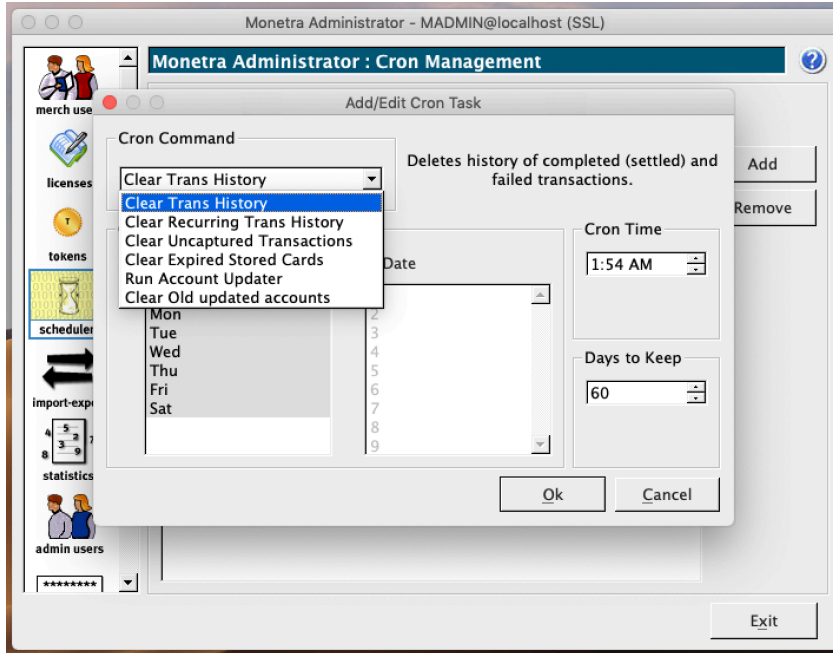
4.11 Configuring Data Retention

Data in Monetra must be configured to be purged when no longer required for legal, regulatory, or business purposes. Please also reference [Section 7.3: Data Retention](#) for additional requirements.

By default, Monetra does not purge any data, as those retention periods must be defined by the merchant. Below, you will find the various data elements that may be purged as well as some recommendations:

- **Transaction History** - Merchant transaction history, such as failed or settled transactions. Due to the possibility of needing to handle customer returns or charge backs, the recommended number of days to keep data is 150 (5 months).
- **Big Batch History** - Each aggregated big batch (not often deployed) contains the cardholder records similar to the Transaction History. However, unlike transaction history, it is of limited use other than for batch re-submission in the event that the processor did not receive a batch. The recommended number of days to keep is 30.
- **Uncaptured Transactions** - Transactions that have been authorized but never accepted into a batch for settlement. These are orphaned transactions and often cannot be settled after 30 days, which is therefore the recommended number of days to keep.
- **Expired Recurring/Tokens** - Cards that are kept on file but have expired are of limited use to merchants. Recommended number of days to keep is 30.
- **Account Updater History** - Every time the card brands notify a merchant of an updated card number, or a merchant edits a card on file, a record is created. The previous card number is kept on file to handle a rollback scenario if an improper update is received and should be purged. The recommended number of days to keep is 60.
- **Order History** - Order history is used for the built-in invoice system, such as is used by SkyLink POS. The recommended number of days to keep is 365.
- **Order Synchronization** - Synchronization history needed to sync between multiple SkyLink POS devices. The recommended number of days to keep is 14.
- **Open Orders Purge** - Dangling open orders that have not been paid. The recommended number of days to keep is 120.
- **Pending Orders Purge** - Orders that a merchant started to create but didn't ever move them to be eligible to be paid are considered pending. The recommended number of days to keep is 30.
- **Canceled Orders Purge** - Orders that a merchant canceled. The recommended number of days to keep is 30.
- **Product Synchronization** - Synchronization history needed to sync the product catalog between multiple SkyLink POS devices. The recommended number of days to keep is 14.

The data retention periods can be configured via the Monetra Administrator under "scheduler" or via the API `action_sys=cron,cron=add` command. Please reference the relevant API documentation for more information.



5 Merchant Configuration

5.1. Merchant Boarding Parameters	43
5.2. Processing Institution Enablement	43
5.3. Adding The Merchant Profile	43
5.4. Adding Additional Routes	45
5.5. Setting Up Users	47
5.6. Automating Settlement Tasks	48

5.1 Merchant Boarding Parameters

When a merchant account is provisioned by a processing institution, the processor will provide a "VAR Sheet" or "Tear Sheet" that contains the relevant information necessary for configuring in Monetra. There can often be a large number of parameters, and mappings can vary from provider to provider.

To begin, it is first recommended to find the processing institution platform among the listings at <https://www.monetra.com/certifications>, using the provided parameter sheet. Once identified, at that same link, click on the "letter" icon to the left of the name to download the Monetra setup worksheet specific to that processor. The setup worksheet will provide descriptions of every field to assist with mapping of what was provided to what Monetra requires. It is recommended to fill out the Monetra-provided sheet before moving on to any further steps.

5.2 Processing Institution Enablement

Before a merchant account can be added to Monetra, the processing institution in use must be enabled. If it has not already been enabled, please do so as described in [Section 4.10: Activating Processing Institutions](#).

5.3 Adding The Merchant Profile

A merchant account can be added via the Monetra Administrator under the "merch users" tab or via the API `action_sys=PROFILE_ADD` and `action_sys=ROUTE_ADD` requests, combined with additional requests to `action_sys=ROUTE_FIELDS` with parameters `route_fields=proclist` and `route_fields=procfields` to determine additional metadata that is required for a given processing institution. Below, we will discuss the usage of the Monetra Administrator to add the merchant profile. API users should reference the relevant sections in the API documentation.

Under the "merch users" tab, click the "Add" button at the top right. You will then be presented with the configuration screen for the account. First, choose an account username and a secure password. You should also enter an email address for the merchant account, which is where automated settlement emails will be sent. The remaining data should be what was filled out in the Merchant Setup Worksheet from the previous section. Select the processing

institution first, then enter the setup parameters as appropriate. Required fields are in red and can change based on the card types selected.

Monetra Administrator - MADMIN@localhost (SSL)

Monetra Administrator : User Management

Search Add

Add Account

Username: bobsburgers

Password: ••••••

Repeat Password: ••••••

Email: bob@burgers.com

Processor: TSYS (aka Vital/VisaNet)

Proc Name: VITAL Help Desk Phone: 800-847-2772

Industry: RETAIL

Card Types: VISA+MC+AMEX+DISC

Fraud Auto Deny:

Field	Value
Bank ID	111111
Merchant ID	111111111111
Store ID	1111
Terminal Number	1111
Agent ID	111111
Chain ID	111111
Zipcode	11111111

Field Information

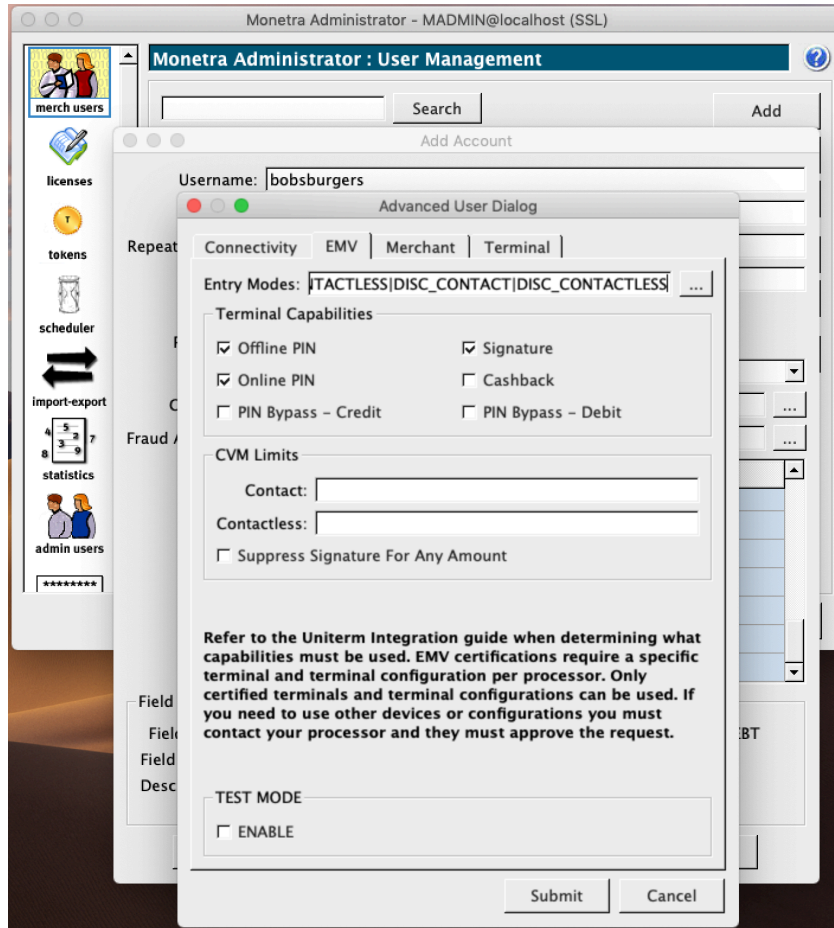
Field Name: merchcat Required: Credit / Debit / Gift / EBT

Field Length: 4 character(s) Field Type: Numeric

Description: Merchant Category Code (aka MCC or SIC)

Advanced Clear Submit Cancel

Before submitting the merchant profile, it is advisable to enter the "Advanced" dialog and visit the "EMV", "Merchant", and "Terminal" tabs to configure as appropriate for the environment. Most users will not use the "Connectivity" tab; it is not recommended to make any modifications in there.



The account may now be submitted. A submission failure can occur if the information provided does not pass the validation checks or, in the event that Monetra needs to reach out to the processing institution to complete the set-up, the processor rejected the data. If the failure occurs due to a processor rejection, please contact your processing institution for assistance.



Note: Behind the scenes a Group specific to the merchant will be created, along with a single user in that group. Then a merchant profile is generated with the associated routes.

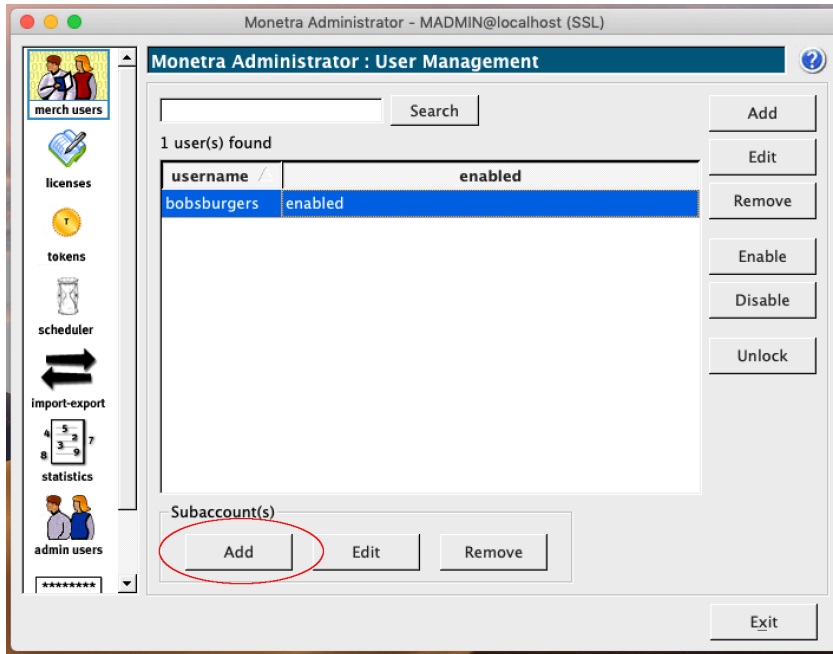


PCI Notice: Once the account is set up, you may create other users with access to the same merchant profile. At no time should a single user account be shared across multiple people.

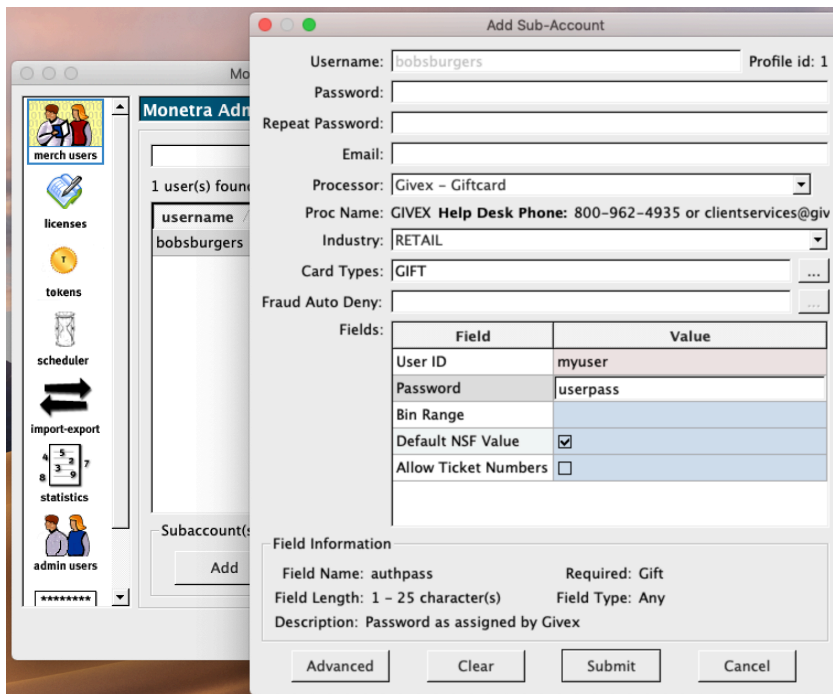
5.4 Adding Additional Routes

Some merchants may have the need to associate multiple routes with the same merchant profile. The most common example of this is routing Credit Cards to one processing institution and Gift Cards to another. Monetra can do this routing automatically behind the scenes using different routes (or sometimes referred to as sub accounts). Such additional routes must not conflict with each other, meaning they cannot be set up to process the same card types and transactions; they must differ in some way.

From the "merch users" tab in the Monetra Administrator, highlight the row of the merchant profile being modified, then click the "Add" button at the bottom of the window (not the top right).



A window very similar to the one used when adding the main merchant profile will show up, but fields like setting a password or email will not be available. Complete the form in the same manner as when adding the original merchant profile, but using the additional setup parameters.



5.5 Setting Up Users

Each merchant group can have multiple users set up that can be used to run transactions, access reports, and perform administrative tasks. The users can be assigned fine-grained permissions, and it is recommended that no user be assigned more privileges than they need. It is always possible to add or remove privileges at a later date. A user should be created for every individual that will directly log in. API Keys or "unattended" users may be used for integrated applications such as Point of Sale systems and Websites, API Keys are the current recommendation, but if using an unattended user you should assign a randomly generated password of at least 20 characters. "Unattended" accounts created for integrated applications must not be used for an individual's login.



Note: Monetra enforces that passwords are at least 8 characters, and contains at least one uppercase letter, one lowercase letter, one number, and one special character.

When using the Monetra Client to add users tied to a merchant, the username is prefixed by the username of the initial merchant profile username. For instance, for a merchant profile named `bobsburgers`, and a user named `John`, they would log in using `bobsburgers:John`. This however is not requirement when using the API.

In order to create users, the Monetra Client may be used, or API users can use the `action_sys=USER_ADD` command. API users should reference the respective API documentation for more information. Monetra Client users will log into the Monetra Client using the username and password created while adding the merchant profile, then navigate to Admin -> sub-user manager and add the relevant accounts.

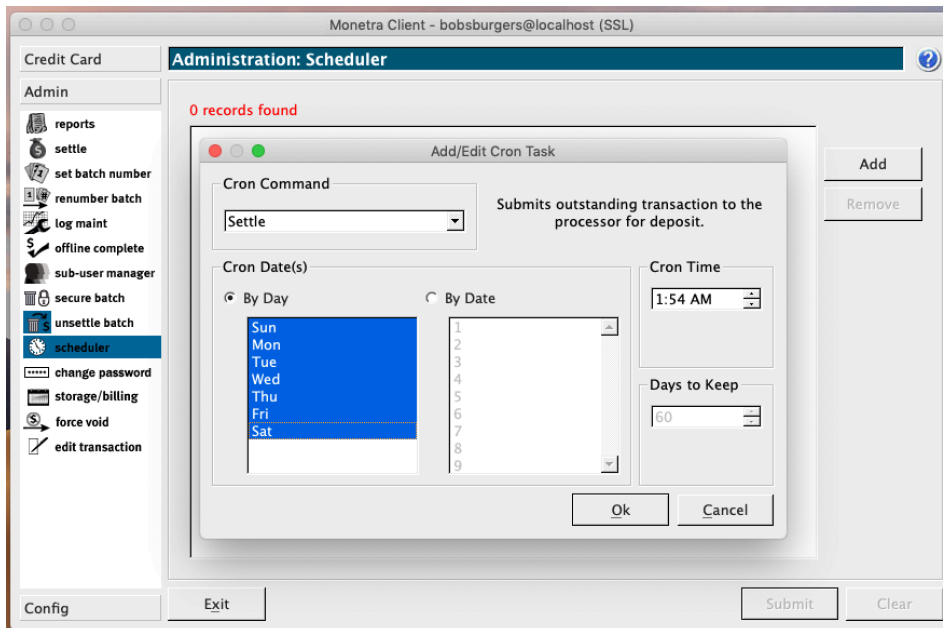


Note: It is strongly recommended at least one individual user is added with the "All Standard" and "All Admin" check boxes enabled to create a merchant-level superuser account.

5.6 Automating Settlement Tasks

Most integrations do not support batch management. Monetra can perform these steps for each account automatically, if configured to do so, and will send a batch summary to the email address associated with the merchant profile. It is recommended to settle every day at the same time, and to settle before your processing institution's cut time. If a cut time is not provided, settlement before 2am EST should work for all US-based processors. That said, if the merchant location is always closed by 5pm, it may make sense to settle long before 2am. Many processors can get overloaded as it approaches 11pm EST. It is therefore recommended that if a late night settlement is necessary, you choose a time frame that is not at an even :15 minute mark (e.g. 12:00, 12:15, 12:30, 12:45), as the in-between times are lower volume and less likely to generate errors.

The automated settlement task can be added via the Monetra Client, or API users can use the `action_sys=cron,cron=add` command. API users should reference the respective API documentation for more information. Monetra Client users will log into the Monetra Client using the username and password created while adding the merchant profile, then navigate to Admin -> scheduler. In this window, a new task can be added using `Settle` as the Cron Command, with all days of the week highlighted (hold down the Shift key and click Sun then Sat with the mouse) and the selected time.



6 Upgrading

6.1. Evaluating Changes	49
6.2. Backing up And Restoring	50
6.3. Implications Of Data Export/Import	50
6.4. Upgrading	51
6.4.1. Cluster Upgrades	51

In order to upgrade to the latest version of Monetra you must be in an active support/maintenance agreement (subscription). Please see [Section 2.4: Licensing](#) for more information.



Note: In order to upgrade to a v9 release, you must first upgrade to a v8 release of 8.20.1 or later. You would then upgrade to the most current v9 release directly, do NOT upgrade to 9.0.0 first as there may be critical data migration bugs fixed in later v9 releases.

6.1 Evaluating Changes

For each Monetra release, it is important to review--in detail--the release notes between the release being run and the target release for upgrade. The release notes for each release are available to all customers with active logins at <https://www.monetra.com/release-notes#monetra-release-notes>. Diligence is required to ensure each minor release between the current and target release are reviewed.

When evaluating the release notes, take note of the database schema version, which is in the form of $x.y$. Any change to the x will result in the necessity to perform a data migration, which will export all data from Monetra and import it into a new database for the new version. This form of upgrade can take a long time and require an extended maintenance window and possibly additional resources on hand such as a DBA or other System Administrator.



Note: A database schema version is in no way correlated to the Monetra versioning scheme. You can have two Monetra releases with the same major version number, but have different database schema version numbers. Always make decisions based on the database schema version from the release notes to know if a major breakage has occurred.

The other thing that must be evaluated is the section in the release notes named "Integration Changes". This is an important section that details, among things, any sort of possible breakage that may occur due to changes in Monetra. More often than not, the changes listed are additions rather than breakages, but care must be taken to fully review the entries.

Finally, before any upgrade takes place in a production system, the upgrade should be tested in a lab environment with a similar setup to production. This serves many purposes:

- Provides an administrator insight into the steps necessary to prevent unexpected downtime
- Tests successful operation of the upgrade itself using data similar to the merchant's production environment



Warning: If doing final upgrade testing using actual production data, you must ensure Monetra's scheduling feature is disabled. Failure to do so may result in duplicate charges,

as any open batches existing in the database could be settled again (First by the production instance, and then again by the test/QA instance). You can disable this using Monetra Manager or by setting `enable_cron=no` in `main.conf`. This should be done before any test system is started when using production data.

- Provides an opportunity to Test/QA any integrated applications to ensure they function in an expected manner with a newer version

6.2 Backing up And Restoring

There are multiple methods of backing up Monetra, depending on how Monetra is deployed. In all scenarios, it is required to stop accepting transactions before backing up. The Monetra Installer does make backups itself for restoration on failure of an upgrade, but there may be scenarios where a rollback is necessary but not done by the installer. Because of this, it is strongly recommended to perform a manual backup before any upgrade.

If Monetra is run on a Virtual Machine, the easiest method is to simply perform an entire system snapshot that can be rolled back if a failure occurs.

In physical environments, or where a snapshot is not desirable, the main components you need for backup are:

- the currently installed version number
- configuration files
- database encryption keys
- if using SQLite, the database itself

The configuration files on a Linux, Unix, or MacOS system are in `/etc/monetra` with a `.conf` extension. For a Windows system, they are in the `etc` sub-directory of the Monetra installation, typically `C:\Program Files\Main Street Softworks\Monetra\etc`. If using SQLite as a database, the database is typically a set of files in the sub-directory named `data` of the Monetra installation; typically `/usr/local/monetra/data` on Linux, Unix, or MacOS systems, and `C:\Program Files\Main Street Softworks\Monetra\data` on Windows. The database encryption key (`my_monetra.key`) is located in the directory where Monetra is installed, typically `/usr/local/monetra/my_monetra.key` on Linux, Unix, or MacOS systems, and `C:\Program Files\Main Street Softworks\Monetra\my_monetra.key` on Windows.

In order to restore a non-snapshot-based backup, it is recommended to remove all installed Monetra software, then re-install and choose the versions noted during the backup. Once done, simply copying the backed-up files to the same place they originated and starting Monetra should restore it to a functioning state.

6.3 Implications Of Data Export/Import

During an upgrade, data export/import may be required due to a schema break or because it is desired by the administrator during the upgrade process (such as for performing a key rotation). When a data migration through export/import is performed, it writes the entire contents of the Monetra database to a compressed and encrypted file and stops the old Monetra

instance. The Monetra Installer will then prompt for a new database connection string and possibly new username and password, start the new instance of Monetra, and create new tables and populate them with the data generated during the export.

The entire data migration process can range anywhere from just a couple of minutes to multiple hours depending on the size of the data set and performance of the underlying systems. It is important for administrators to perform upgrade testing on similar data set sizes with similar performance hardware and system configurations in order to gauge the amount of time necessary for such a migration.

In general, it is recommended to NOT perform a data export/import unless there is a critical business need behind the decision.



Note: There is an option of a "staged" migration for larger customers to minimize the overall system downtime. The "staged" migration can be used to bring a system partially back online and migrate bulk history in the background while the system is otherwise operational. The "staged" migration is a much more complex and manual operation with additional risks, and it is not something facilitated through the Monetra Installer. If interested in obtaining more information on this solution, please email support@monetra.com.

6.4 Upgrading

Upgrading Monetra is accomplished via the Monetra Installer. After logging in or providing an offline package, select the Monetra Engine license from the list, then choose the Upgrade option. The next dialog will confirm the version to upgrade. Follow the on-screen prompts, and the installer will guide you through the process.

During the upgrade process, the Monetra Installer will validate the cryptographic signature of the package being installed and abort if there is any tampering detected.



Note: It is recommended to answer NO to the question regarding if you need to export/import data unless you have an explicit business need.

6.4.1 Cluster Upgrades

Upgrading Monetra in a cluster is very similar to a normal upgrade. In fact, if there has been no schema break and no export/import process performed, there are no additional steps other than to perform the same upgrade on each node. Monetra supports running a cluster on mixed versions during the upgrade process as long as the schema versions are compatible, but this is not recommended for long periods of time.

First, if performing an upgrade that is undergoing an export/import, transactions must be stopped to all nodes. Also, a strict Maintenance Mode should be configured on each node (available via the Monetra Administrator) to ensure that no automated tasks attempt to run. Next, choose the first node to be migrated. One of the questions will be if this is part of a cluster and if it is the primary node in the cluster; select *yes*. This will perform the same steps as a single-node Monetra instance performing an export/import. Finally, after the first node is successfully upgraded, start the upgrade process on the secondary node(s), but select *no*

regarding if this is the primary node in the cluster. It will then perform the prompting regarding any database modifications and instruct you to copy over the database encryption key from the primary node. It will perform no data migration steps, just a binary upgrade and configuration merge.

7 Secure Implementation And Deployment

7.1. Firewalls	53
7.2. Database	54
7.2.1. Deployment	54
7.2.2. Key Rotation	54
7.2.3. Sensitive Data Locations	55
7.3. Data Retention	56
7.4. Inbound Communications Protocols	57
7.5. Cryptographic Subsystem Information	57
7.6. Signed TLS Server Certificates For Public-Facing Installations	57
7.6.1. Generating Signing Request With OpenSSL	58
7.6.2. Installing The Certificate	59
7.7. Multi-factor Authentication (MFA)	59
7.7.1. Time-based One Time Pin (TOTP)	59
7.7.2. Requiring TLS Client (Machine) Certificates	60
7.8. API Keys for Integrated Applications	61
7.9. Self-Service MFA and Password Resets	61
7.10. Token Export	61
7.11. Security-Affecting Configuration Options	62
7.11.1. main.conf	62
7.11.2. prefs.conf	63
7.12. PCI Security And Implementation	64

7.1 Firewalls

Monetra must be installed behind a firewall in all production environments to comply with PCI security requirements. Depending on the deployment type, this may be a DMZ if Monetra is intended to accept requests from the public Internet. In all deployments, only the necessary ports for both ingress (inbound, Internet to Monetra) and egress (outbound, Monetra to Internet) should be opened.

By default, Monetra only listens on two TCP ports: 8665 and 8666. 8665 is used for merchant-level transactions and reporting, while 8666 is used for Monetra Administrator-level actions such as merchant account management. Typically, only port 8665 should be opened for ingress from public or untrusted networks, while port 8666 would only be accessible from a trusted network segment.

For egress, there are a few destinations that are recommended to be open. Egress rules should explicitly list IP address and port combinations and not simply open a port to the entire world:

- Rules necessary for the Monetra Installer to operate as per [Section 3.4.1: Online Installation \(Recommended\)](#)
- DNS ports 53 for both UDP and TCP to trusted DNS servers
- SMTP (Mail) TCP ports 25, 465, and/or 587 to a trusted SMTP server for email notifications
- Rules necessary to connect to the database server, if not using SQLite

- Processing Institution address and ports. URLs can be seen in Monetra's `processors.conf`. However, it may be necessary to contact each processing institution in use to obtain an up-to-date list of possible IP addresses in use. Some may use round-robin DNS, so relying on DNS resolution in a firewall may be insufficient.

PCI PCI Notice: If Monetra is deployed in a DMZ, the database must reside in a different network segment.

7.2 Database

7.2.1 Deployment

A database must be deployed in a firewall zone that is not connected to the Internet. This means that, if Monetra is deployed within a DMZ, the database must be deployed in a separate firewall zone that is not a DMZ.

PCI PCI Notice: If SQLite is used, Monetra cannot be deployed within a DMZ and is not suitable as a public-facing deployment. An external database must be used if Monetra is deployed within a DMZ. If SQLite is used, it is also the installer's responsibility to ensure permissions are set properly on the SQLite database directory and all accesses to the database are logged via the OS audit controls.

7.2.2 Key Rotation

7.2.2.1 Data Encryption Key

The database data encryption key is automatically rotated by the defined `crypto_period_days` within Monetra's `main.conf` configuration file. The valid range is 30-730 days, with a default of 180 days. When a rotation occurs, all new data will be encrypted under the new key, the old key will be retained in order to decrypt previously encrypted data.

The data encryption keys are protected by the Key Encryption Key as defined in the configuration [Section 4.3.1: Protection Mechanisms](#).

7.2.2.2 Key Encryption Key

The simplest method of performing a KEK rotation is to issue an upgrade or re-install via the Monetra Installer for Monetra. When prompted if an export/import is desired, answer `yes` to this question. The next question will be if a key rotation is desired; also answer `yes` to this question and any other follow-up questions regarding key rotation. Once the process completes successfully, the key rotation of the Key Encryption Key is complete. Please reference [Section 6.3: Implications Of Data Export/Import](#) for more information regarding the export/import process.

Since all KEKs are wrapped keys, simply deleting the key file is a sufficient destruction process since there is no retrievable key material. This is automatically performed as part of the KEK rotation as described above.

7.2.3 Sensitive Data Locations

7.2.3.1 Database

This section documents the locations within the Monetra database of all sensitive (and encrypted) data.

Table	Column	Use Case	Protection
Cryptographic Keys			
keys	key	Auto-rotated data encryption keys	KEK
cardshield_keys	enckey	P2PE encrypted data keys for decryption	HSM or Data Encryption Key
Cardholder Data			
transdata	account	PAN storage for transactions	Data Encryption Key
bbtransdata	account	PAN storage for transactions in big batch subsystem	Data Encryption Key
tokens	account	PAN storage for tokens	Data Encryption Key
token_updates	account	PAN storage for token history for rollback	Data Encryption Key
carddenylist	account	PAN storage for deny listing cards (fraud, test cards, etc)	Data Encryption Key
Authentication Data			
users	password	Current user password	Data Encryption Key (plus optional PBKDF2)
historic_passwords	password	Historic user password to prevent re-use	Data Encryption Key (plus optional PBKDF2)
users	mfa_secret	TOTP Secret for MFA	Data Encryption Key
mfa_cookie	cookie	Temporary MFA bypass cookie	Data Encryption Key
secquest	answer1, answer2, answer3, answer4	Answers to security questions	PBKDF2

auth_reset	code	8 digit code used for password and MFA reset	Data Encryption Key
apikey	secret	Secret data used for authentication of API key	Data Encryption Key

7.2.3.2 Export Files

When requested, an encrypted export file representing the contents of the Monetra database can be generated. One of the arguments passed to the export command is the path to write the export file, local to the server running Monetra application. Care must be taken to ensure this data is deleted when no longer needed.

During an upgrade process, an export may be initiated by the Monetra Installer. The folder in which this export is created is in the `archive` sub folder of the Monetra Installer installation path. By default that is `C:\Program Files\Main Street Softworks\Monetra Installer\archive` on Windows, and `/usr/local/monetrainstaller/archive` on Linux.

7.3 Data Retention

PCI SSF requires that cardholder data deletion processes follow industry-accepted standards for secure deletion (e.g., NSA, NIST SP 800-88):

<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-88r1.pdf>

<https://www.nsa.gov/Resources/Media-Destruction-Guidance/>

<https://www.nsa.gov/Portals/70/documents/resources/everyone/media-destruction/PM9-12.pdf>

Customers and integrators must ensure that underlying systems and software used by the application follow industry-accepted standards for secure deletion. Consideration should also be given to the use of other technologies in use, such as redundant disk arrays, system snapshots, and use of any log-structured or journaled file systems (e.g., `reiserfs`, `ext4`, `xf`s).

Monetra does not view legacy methods for file overwrite as sufficient to securely delete stored data, due to the nature of technologies in use today. While tools such as `SDelete` for Windows and `shred` for Linux-based systems may have historically been used by other organizations for overwriting specific files and locations, it does not guarantee that all data and data locations have been fully addressed. Dedicated sanitize commands and Cryptographic Erase for storage media may address additional storage locations, although certified physical destruction provides the most reasonable level of assurance. Please refer to your device manufacturers for additional, vendor-specific guidance and utilities, to comply with this requirement. An example is referenced below:

<https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/ssd-pro-1500-secure-erase-paper.pdf>

Please see [Section 4.11: Configuring Data Retention](#) for configuring Monetra's automatic data purging.

7.4 Inbound Communications Protocols

All communications protocols for inbound requests to Monetra are secured with TLSv1.2 or higher--using an explicitly specified cipher list to only select secure ciphers--and provide full support for forward secrecy. As of Monetra v8.11, TLSv1.3 is also supported. The protocol and cipher negotiated on any given connection can be seen in the Monetra logs when the connection is established. When a connection is established and the first message is exchanged, Monetra performs protocol auto-detection on the channel. Monetra supports multiple protocols such as our proprietary key/value pair protocol, XML over HTTPS, JSON over HTTPS, a ReSTful API, PaymentFrame/iFrame messaging, Direct Post, and a handful of common emulations.

It should be noted that, while Monetra has direct support for some web-based protocols, it is not a web server. Monetra has a minimal HTTP parser that is used to facilitate interpreting of these protocols; it is not fully standards-compliant, as it implements only the minimum required support in order to reduce the possible attack surface. It is not a full blown web server and does not have the ability to serve static files or communicate with external scripting languages.

7.5 Cryptographic Subsystem Information

Monetra utilizes the OpenSSL [<https://openssl.org/>] cryptographic libraries to perform all cryptographic operations including Random Number generation, symmetric cryptography, asymmetric cryptography, one-way hash functions, and TLSv1.2+ network streams. The only exception to this is when an HSM is in use. When an HSM is in use, the HSM will be used to wrap and unwrap the KEK used for all database encryption/decryption operations. An HSM can also be used for CardShield P2PE decryption, where the BDK is never exposed to the system running Monetra.

All cryptographically secure random number generation, which is used for key generation as well as cipher IV generation, utilizes the default OpenSSL Random Number Generator [https://wiki.openssl.org/index.php/Random_Numbers]. Monetra does not manually seed the RNG and instead relies on OpenSSL's internal seed process.

For symmetric cryptography, when the AES-CBC algorithm is chosen, a 128-bit cryptographically secure random number is used as the IV and is padded using PKCS5. When using the AES-GCM algorithm, a 96-bit cryptographically secure random number is used as the IV, there is no padding in use with GCM. Both the 64bit key id referencing the real encryption key, along with the IV are prefixed onto the encrypted data in the same database column as the encrypted data itself. When using GCM mode, the 128-bit tag is appended to the end of the data.

7.6 Signed TLS Server Certificates For Public-Facing Installations

Monetra ships by default with a self-signed TLS server certificate that is sufficient to enable point-to-point encrypted sessions on a trusted network. However, when transactions are needing to flow from the Internet to Monetra, the TLS certificate must be signed by one of the large, trusted Root Certificate Authorities (e.g. Digicert, Verisign, Entrust). By having a

certificate signed by a trusted Root Certificate Authority, the end user can be confident that the endpoint being connected to is who they say they are and that they are not vulnerable to a man-in-the-middle attack. It is imperative that the TLS certificate is set up with a Common Name or a Subject Alternative Name (SAN) that matches the Fully Qualified Domain Name (FQDN) of the Monetra instance.

Monetra supports X509 PEM-encoded keys and certificates. This is the same certificate and key format that the Apache web server uses.

7.6.1 Generating Signing Request With OpenSSL

A system administrator can use whatever tools they are comfortable with for generating a key and a certificate signing request (CSR), but a common method is to use the `openssl` command line utility. An example of using a utility to generate a private key and CSR are provided below.

```
$ openssl genrsa 2048 > monetra.mydomain.com.key
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
```

```
$ openssl req -new -sha256 -key monetra.mydomain.com.key > \
monetra.mydomain.com.csr
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) []:Jacksonville
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
    Monetra Technologies, LLC
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:monetra.mydomain.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

```
$ cat monetra.mydomain.com.csr
-----BEGIN CERTIFICATE REQUEST-----
MIICvjCCAaYCAQAweTELMakGALUEBhMCVVMxEDAObgNVBAGMB0Zsb3JpZGExFTAT
BgNVBACMDepHY2tzb252aWxsZTEiMCAgALUECgwZTW9uZXRYYSBUZWNobm9sb2dp
ZXMsIEExMQZEdMBSGALUEAwWUbW9uZXRYYS5teWRvbWVpbi5jb20wggeiMA0GCSqG
SIb3DQEBAQUAA4IBDwAwggEKAoIBAQC2GncPQTGo8n0fovCNXu+8cPzsw2NvrQYG
fgk/MrTRWWNiu3W56hgKVDK3Ki7BA5eJCVRTsUDDnyCvMfUD/FpA3S7ONLfJZFfd
AXa9Zh43LkdaPU0h6HNdf2CUmyxF7pUhMSn9ME5k+S5zhCqELrx5+2pMqVeu6+zh
xuLpx/yrjferQalR6uHRW6j+UNijQdVc5DqFTz/WgpaAE1IhslrLTBswupP/9w96
M7B/JrUHPdvz6lQNiKjEo6Eabmj92iIY2rxtPyx622ir573TsnpxNUZyvQvmUaLb
MGRgohRXFjQwDd49c+CFFm8cxJ9s+Ev4rKbilvtb54s9VqNr4WUFAGMBAAGgADAN
BgkqhkiG9w0BAQsFAAOCAQEAd9KA0OpMJ4bQoQKu0+zO8oDLg54MXBeeStS9kaTx
+IewYM+Np97XJiXb29ZDLb/EQuAaQuFaORECAyTVj8Hr/ZGg3CpH2vYxd7PfxaWV
FTE79xThMJDCmH18twdxVu+C+Aw62HNOXBfV1Trdy60hiYu8aGeJWfvwtPSS10aV
```

```
/XGZmlz+RrfGoRI07++Rl8VMvZV5m+TY9cpDz9TCYTzYrkbA/iPnlOCuXQ/WXVi1
KaHnG4Z4wkwQ6Mo15QnyCXt6WQN8eO9Of0e0krA6zhnHFT+No27Vk+aW+z3EdFH+N
37tLDzxf29MHMcqDtbeMzbb5htDU2NHhtNzLINE8zJV//Q==
-----END CERTIFICATE REQUEST-----
```

Once the CSR is obtained, this must be provided to the Certificate Authority for them to validate and sign. The CA may require additional verification steps such as Domain Control Validation. Please work with the CA on such steps.

7.6.2 Installing The Certificate

Once a key and signed certificate are provisioned, it is simply a matter of placing the files somewhere that Monetra can access them. Often, a good place is in the same folder where the Monetra configuration files are located. The same files and same configuration must be put in place for each node in a clustered environment. When a Certificate Authority returns a signed certificate, they will often also return Intermediate or Bundle certificate(s) as well. Monetra needs these combined with the server certificate in a single file, with the server certificate first in the file. Since the certificates are PEM-encoded, they are plain text, so this can be performed with a text editor if necessary.

In `prefs.conf` or via the Monetra Administrator, the certificate and key paths can be specified via `COMM_ssl_cert` and `COMM_ssl_key`, respectively. If multiple possible certificates are needed to serve multiple domains, these can be specified as semi-colon separated lists of certificates and keys as long as care is taken to ensure the same number of keys exist as certificates and the order is consistent. If it is desirable to use different certificates and keys per port, it is also possible to alternatively specify `COMM_ssl_cert_{port}` and `COMM_ssl_key_{port}` to override the port-specific configuration.

7.7 Multi-factor Authentication (MFA)

Multi-factor authentication combines something you know (e.g. username and password) with something you have (e.g. system certificate, rotating code on a device [TOTP], access to another system via email or SMS). Leveraging multi-factor authentication can greatly reduce the risk of account compromise as if an attacker were to gain access to one factor, the other factor would likely still remain secure, thus denying the attacker access.



Note: Some API requests may require a user to have MFA enabled, such as `action_su=TOKEN_EXPORT`.

7.7.1 Time-based One Time Pin (TOTP)

Monetra supports TOTP as its recommended method for implementation of Multi-factor authentication. This method was popularized by Google with their Google Authenticator product, but is officially defined by [RFC6238](https://datatracker.ietf.org/doc/html/rfc6238) [https://datatracker.ietf.org/doc/html/rfc6238]. In order to maintain compatibility with Google Authenticator, Monetra uses an 80bit cryptographically-secure randomly generated seed (per user), the SHA1 HMAC algorithm, along with a 30 second period and a 6 digit one time pin.

The user is allowed to choose their method of delivery for the TOTP code. Options are:

- `App` - User-managed app such as Google Authenticator, Authy, or Last Pass Authenticator.
- `SMS` - A US-based phone number where text messages are delivered containing the code on demand. Requires [Section 4.6: SMS configuration for MFA, Password Resets, Receipts](#)
- `Email` - An email address accessible to the user where the code will be delivered on demand. Requires [Section 4.5: Mail Configuration For Notices, Settlement Summaries, MFA, and Receipts](#)

Users are allowed to enable and generate a TOTP code using the `action_sys=MFA_GENERATE` API request. If a user is set up for MFA but has not provided an `mfa_code` parameter, the user will be returned an `msoft_code=MFA_REQUIRED` response. If an administrator forcibly enabled MFA for a user but the user has not yet generated their MFA code, the user will be return `msoft_code=MFA_GENERATE` letting them know they must generate an MFA code.

Users may change their MFA code or method of delivery at any time so long as they have not lost their existing MFA code or password by calling the `action_sys=MFA_GENERATE` API call again.

A system administrator may enforce all users contained within a group use MFA by setting a group flag of `REQUIRE_MFA`. This flag will automatically add the MFA flag to the users contained within that immediate group (but will not propagate to child groups and users).

7.7.2 Requiring TLS Client (Machine) Certificates

An additional security feature that may be desirable to enable Administrative connections is the validation of client-machine certificates when they connect to Monetra. Based on the port configuration, each client would connect to Monetra with a unique TLS client certificate. The certificate will be verified against the certificate authority configured in Monetra for the port and also checked to ensure it has not been revoked against the Certificate Revocation List (CRL). This is an additional security control that provides Multi-factor authentication, the standard username and password are still required for all transaction requests. In order to use client certificates, the endpoint connecting to Monetra must support advertising a client certificate. If it does not already, then engineering work will be required for the client application to support this feature.

Though machine certificates are widely considered to be a form of MFA, Monetra does not treat machine certificates as a valid MFA method to satisfy user MFA requirements.

Monetra expects a PEM-encoded Certificate Authority Root and CRL to be provided when enabling Client Certificate Verification. Most customers will set up their own private Root Certificate Authority for this purpose by whatever means works best for them. A good example using OpenSSL to accomplish this is available at <https://jamielinux.com/docs/openssl-certificate-authority/>.

Once a Root Certificate Authority and a CRL have been generated, copy over the relevant files to somewhere Monetra can access them. In general, the recommendation is to use the configuration directory of Monetra. In order to enable verification, in `prefs.conf` or via the Monetra Administrator, set `COMM_ssl_cert_required_{port}=yes` (replacing `{port}` with the port on which verification is being enabled). Then also set

`COMM_ssl_cafile_{port}=` and `COMM_ssl_crlfile_{port}=` to the full paths of the relevant files.

Once these steps have been completed, only clients with valid signed certificates will be able to connect. All others will be rejected before being able to run any requests.

7.8 API Keys for Integrated Applications

API Keys are the recommended authentication mechanism designed for integrated applications. API Requests should generally not contain a username and password but instead be used to generate API keys which are then used for any follow-on requests. API Keys allow fine-grained permissions to be set, and can be created for either a specific user or a specific merchant profile, depending on the desired scope of the API Key. More typically a merchant profile API key will be used for things like websites or POS integrations, while a user API key will be used for things like user login sessions.

In addition to the difference between merchant and user API keys, an integrator can also specify an expiration time of an API key. For keys which expire, there is a flag that can be set to allow an API key to auto-extend on use. It is generally recommended for user sessions to create API keys with an expiration of 15 minutes which auto-extend on use.

Please see our `action_sys=APIKEYS` API documentation for more information regarding usage of API Keys.

7.9 Self-Service MFA and Password Resets

Monetra supports a secure reset mechanism for both user passwords as well as MFA. If a user has lost both their password and MFA, then there is no ability to perform a self-reset. The self-reset is implemented via the `action_sys=AUTH_RESET` API functions.

There are a few pre-requisites for a user to be able to perform self-resets:

- Monetra must be configured for SMS and/or Email as a one-time code must be sent to the user
- The user must have configured a `phone_mobile` and/or `email` on their user account in order to send the one-time code
- The user must have set up 3 security questions with answers. These can be set up via the `action_sys=SECURITY_QUESTIONS` API.

7.10 Token Export

In the rare event that tokens need to be exported from Monetra, an API request of `action_su=token_export` exists. This API request will take in a group id and a PGP public key to encrypt the data under. All tokens contained within the group will be exported as CSV and then encrypted using PGP using the supplied public key.

Since this command provides access to cardholder data, it is required any user initiating this request has both the permission assigned for token export, is in the top-level system group, and

is enabled for Multi-factor authentication. It is the integrator's responsibility to come up with their own policies and procedures surrounding the export of tokens. For instance, it would be strongly recommended to ensure the PGP public key is coming from a remote entity's website secured via HTTPS to ensure no intermediary is trying to compromise the process. It is also strongly recommended to validate the entity the tokens are being exported for is a PCI-DSS level 1 service provider.

7.11 Security-Affecting Configuration Options

This section will provide a complete list of configuration options that in some way can affect the security of Monetra. Some of these may be used to increase the security levels, and some of these must stay at their current values in order to be deployed in a PCI-compliant manner. The values are documented in the below sections based on the configuration file they reside within. Many values may also be found within the Monetra Manager with conceptually similar names and functionality.

7.11.1 main.conf

- `debug` - Logging flags. Default is `INIT|CONF|WARN|INFO|TRAN|TRAN_DETAIL|CONN|PROC|PROC_DETAIL|ERROR|CRIT`. Logging must be enabled for compliance with PCI DSS with at least the minimum flags of `WARN|INFO|TRAN|TRAN_DETAIL|ERROR|CRIT`.
- `dbencrypt` - Database encryption key protection method. Please see [Section 4.3.1: Protection Mechanisms](#).
- `dukptencrypt` - P2PE encryption mechanism. Please see [Section 4.3.1: Protection Mechanisms](#).
- `encalgorithm` - Encryption Algorithm/Cipher to use for database field-level encryption. Possible options are `AES` (AES-CBC), and `AES-GCM` (AEAD cipher). The Default is `AES-GCM` and is considered the strongest possible algorithm. However, `AES-CBC` is still considered strong and is an acceptable option.
- `enckeylen` - Encryption Key Length. Default is 256. May be 128, 192, or 256, depending on the cipher chosen.
- `crypto_period_days` - The number of days the current data encryption key is allowed to be used for encrypting new field-level data. After this time frame a new key will be automatically generated. The valid range is 30-730 days with a default of 180 days. Any of these values are considered compliant and up to the installer's to define an appropriate crypto period.
- `password_protection` - The method in which passwords are stored within Monetra. Available options are `EncryptPBKDF2` and `Encrypt`. `EncryptPBKDF2` is the default and performs a PBKDF2 hash to the password, then encrypts the hash result under the database encryption key. `Encrypt` encrypts the plain-text password under the database encryption key. Both are suitable for PCI-DSS deployments. However, `Encrypt` must be used for Direct Post and PaymentFrame support when NOT using API Keys due to the use of HMAC authentication which requires access to the actual password. API Keys are recommended.

- `password_pbkdf2_iterations` - Configured number of iterations for passwords stored using PBKDF2. Default is 1000, which is the minimum secure setting.
- `max_password_failures` - The number of consecutive password failures until an account is locked out for `password_lockout_seconds`. The default is 6, which is the minimum under PCI rules.
- `password_lockout_seconds` - The number of seconds an account is locked out for password failures. The default is 1800, which is the minimum under PCI rules.
- `require_strong_passwords` - Whether or not passwords are required to contain characters from all of these character sets: uppercase letters, lowercase letters, numbers, and special characters. The default value is `yes`, which is required under PCI rules.
- `password_history_length` - How many historic passwords to keep on file to verify the same password isn't used on rotation. The default value is 4 and is the minimum required value under PCI rules.
- `force_password_change_days` - How many days a password is valid before it expires. The default value is 90, which is the maximum setting under PCI rules.
- `ssl_server_protocols` - These are the supported protocols for inbound communication to Monetra. The default when not specified is `tlsv1.2+`. The only other acceptable value under PCI rules is `tlsv1.2`, which disables TLSv1.3 support.
- `ssl_client_protocols` - These are the supported protocols for outbound communication from Monetra. The default when not specified is `tlsv1.2+`. The only other acceptable value under PCI rules is `tlsv1.2`, which disables TLSv1.3 support.
- `ssl_server_ciphers` - These are the supported ciphers for inbound communication to Monetra. Do not set or change these without consulting with an auditor.
- `ssl_client_ciphers` - These are the supported ciphers for outbound communication from Monetra. Do not set or change these without consulting with an auditor.

7.11.2 `prefs.conf`

- `COMM_whitelist` - whitelist of IP addresses that will not automatically be blacklisted for suspicious behavior. Defaults to private trusted IP addresses: `127.0.0.0/8,192.168.0.0/16,10.0.0.0/8,172.16.0.0/12,::1/128,fc00::/7`. Not recommended to change.
- `COMM_blacklist_threshold` - number of suspicious events from a single IP address (password failures, protocol failures, TLS negotiation failures) before it is blacklisted and disallowed from establishing new connections for `COMM_blacklist_reset_time_seconds`. Default value is 5. PCI has no defined minimum values for such a setting as it falls under the scope of Intrusion Detection and Prevention.
- `COMM_blacklist_reset_time_seconds` - Number of seconds after a blacklist event is triggered that an IP address will remain blacklisted. Default value is 1800. PCI has no

defined minimum values for such a setting as it falls under the scope of Intrusion Detection and Prevention.

- `COMM_ssl_cert[_{port}]` - TLS certificate to use for inbound connections. Please see [Section 7.6: Signed TLS Server Certificates For Public-Facing Installations](#).
- `COMM_ssl_key[_{port}]` - Key for TLS certificate to use for inbound connections. Please see [Section 7.6: Signed TLS Server Certificates For Public-Facing Installations](#).
- `COMM_ssl_cert_required[_{port}]` - Whether or not to enable TLS client certificate verification for a port. Please see [Section 7.7.2: Requiring TLS Client \(Machine\) Certificates](#).
- `COMM_ssl_cafile[_{port}]` - When TLS client certificate verification is enabled, the CA to use for verification. Please see [Section 7.7.2: Requiring TLS Client \(Machine\) Certificates](#).
- `COMM_ssl_crlfile[_{port}]` - When TLS client certificate verification is enabled, the Revocation list to check. Please see [Section 7.7.2: Requiring TLS Client \(Machine\) Certificates](#).
- `COMM_perms_{port}` - The restrictions to enable on a port. By default, port 8665 only allows user requests and port 8666 only allows Administrator requests. This is used to enforce firewall separation of port usage. Change with caution.

7.12 PCI Security And Implementation


The below table details the various security and PCI requirements and how deployments may be impacted. Integrators and distributors should read this section prior to any production deployments. Monetra is designed and configured by default to be compliant with all PCI SSF requirements.



Note: Please use this section along with the official PCI DSS v3.2.1 specification available at https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2-1.pdf

The topics of this section are taken from the PA-DSS v3.2 Appendix A at https://www.pcisecuritystandards.org/documents/PA-DSS_v3-2.pdf. The PCI SSF standard doesn't have an equivalent topic list, this section is kept for additional detail for installers.

TOPIC	DISCUSSION
Delete sensitive authentication data stored by previous payment application versions.	Monetra has never stored any sensitive authentication data in a non-secured or non-approved manner.
Delete any sensitive authentication data (pre-authorization) gathered as a result of troubleshooting the payment application.	Monetra does not have the ability to store sensitive authentication data for troubleshooting purposes. Troubleshooting data is only ever stored in volatile memory. If a merchant copies this data into another form, they must do so as per the requirements of PCI-DSS. Please see Section 4.1: Logging .

Securely delete cardholder data after customer-defined retention period.	Please see how to configure data retention periods in Section 4.11: Configuring Data Retention .
Mask PAN when displayed so only personnel with a business need can see the full PAN.	There is no method to retrieve any of the the PAN data from Monetra other than the last 4 digits and card type in plain text. It is, however, possible to export card tokens using the <code>action_su=token_export</code> API call, but that call requires MFA and also a PGP public key to encrypt the data under and is never used for display purposes.
Render PAN unreadable anywhere it is stored (including data on portable digital media, backup media, and in logs).	Sensitive cardholder data is always unreadable as per Section 2.3.2.2: Databases and Section 4.2: Database . This is always true; there is no configuration option that controls this behavior.
Protect keys used to secure cardholder data against disclosure and misuse.	Key protection and management is described in detail in Section 4.3.1: Protection Mechanisms .
Implement key-management processes and procedures for cryptographic keys used for encryption of cardholder data.	Key protection and management is described in detail in Section 4.3.1: Protection Mechanisms , and rotation is discussed in Section 7.2.2: Key Rotation .
Implement secure key-management functions.	Key protection and management is described in detail in Section 4.3.1: Protection Mechanisms .
Provide a mechanism to render irretrievable cryptographic key material or cryptograms stored by the payment application.	Monetra never supports clear-text output of keys or components. Key protection and management is described in detail in Section 4.3.1: Protection Mechanisms .
Use unique user IDs and secure authentication for administrative access and access to cardholder data.	All access controls used to secure access to cardholder data are provided by Monetra. The only access to direct cardholder data is via the <code>action_su=token_export</code> API call which requires both MFA as well as a PGP public key used to re-encrypt the cardholder data under. Please see Section 4.9: Creating Initial Administrator User(s) , Section 7.11.1: <code>main.conf</code> , and Section 7.10: Token Export .
Use unique user IDs and secure authentication for access to PCs, servers, and databases with payment applications.	<p>Monetra does not provide or facilitate administrative or remote access to PCs, servers, or databases.</p> <p>Monetra provides no direct access to cardholder data (only encrypted token exports), but it does facilitate some management of such data. Please see Section 4.9: Creating Initial Administrator User(s), Section 7.11.1: <code>main.conf</code> and Section 7.10: Token Export.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">  Note: It is the integrator's responsibility to ensure unique user names and secure authentication are </div>

	used to access any PCs, servers, and databases with payment applications and/or cardholder data. This requirement is external to Monetra and not something that can be provided by Monetra on behalf of the merchant or integrator.
Implement automated audit trails.	Audit trails are provided by logging as described in Section 4.1: Logging
Facilitate centralized logging.	Centralized logging is facilitated by the syslog subsystem, or by an installer-configured third-party solution, as described in Section 4.1: Logging
Implement and communicate application versioning methodology.	Please see the Versioning section.
Securely implement wireless technology.	Monetra is not designed to facilitate the use of wireless technologies; however, merchants may deploy Monetra in environments where wireless technologies are in use. Monetra always uses secure communications channels protected by TLS v1.2 or higher. These channels are designed for use in public/untrusted networks to protect sensitive cardholder data while in transit.
Secure transmissions of cardholder data over wireless networks.	Monetra is not designed to facilitate the use of wireless technologies; however, merchants may deploy Monetra in environments where wireless technologies are in use. Monetra always uses secure communications channels protected by TLS v1.2 or higher. These channels are designed for use in public/untrusted networks to protect sensitive cardholder data while in transit.
Provide instructions for secure use of wireless technology.	<p>Monetra is not designed to facilitate the use of wireless technologies; however, merchants may deploy Monetra in environments where wireless technologies are in use. Monetra always uses secure communications channels protected by TLS v1.2 or higher. These channels are designed for use in public/untrusted networks to protect sensitive cardholder data while in transit.</p> <p>Integrators should ensure wireless technology in use is deployed in a manner that aligns with PCI DSS Requirements, including:</p> <ul style="list-style-type: none"> • changing default wireless encryption keys, passwords, and SNMP community strings upon installation. • changing wireless encryption keys, passwords, and SNMP community strings any time anyone with knowledge of the keys/passwords leaves the company or changes positions • installing a firewall between any wireless networks and systems that store cardholder data, and to configure firewalls to deny or control (if such traffic is necessary

	<p>for business purposes) any traffic from the wireless environment into the cardholder data environment</p> <ul style="list-style-type: none"> • use industry best practices to provide strong encryption for authentication transmission
Provide instructions for customers about secure installation of patches and updates.	Monetra update notifications are provided via subscription to the Monetra Release Notes RSS feed: https://www.monetra.com/release-notes . All package updates are cryptographically validated as per Section 6.4: Upgrading .
Use only necessary and secure services, protocols, components and dependent software and hardware, including those provided by third parties.	<p>Monetra communicates only via TLSv1.2 or higher, regardless of whether the communication is on a trusted LAN or over the Internet. There are no additional services, protocols, hardware, or software required to run Monetra as per Section 2.3: System Requirements.</p> <p>It is the integrator's responsibility to ensure only necessary and secure protocols, services, etc., are used on the system.</p>
Store cardholder data only on servers not connected to the Internet	Cardholder data must not be stored on systems connected to the Internet as per Section 7.1: Firewalls .
Implement two-factor authentication for all remote access to payment application that originates from outside the customer environment.	<p>Monetra does not facilitate remote access, so integrators or merchants choosing to provide an external means of remote access must ensure that all remote access originating from outside the customer's network to a payment application must use two-factor authentication. Two-factor means two separate types of authentication: something you know and something you have. A username and password is one factor. A second password would not be considered a second factor; it should instead be something external such as a token, TOTP, certificate, fingerprint, smartcard, etc.</p> <p>Monetra does support direct multi-factor authentication for those who choose to use it. Please see Section 7.7: Multi-factor Authentication (MFA).</p>
Securely deliver remote payment application updates.	All application updates are delivered via the Monetra Installer, which validates package signatures before installation. Please see Section 6.4: Upgrading . Deployments must be done in accordance with the PCI SSF requirement 11. Monetra Technologies does not provide remote access based updates to customer systems.
Securely implement remote-access software.	Monetra Technologies will never reach out to a remote customer network. If an integrator or merchant chooses to support remote access for management, they must do so in compliance with PCI DSS requirements, including Requirement 8 and the use of multi-factor authentication with at least two out of three authentication methods.

<p>Secure transmissions of cardholder data over public networks.</p>	<p>Monetra communicates only via TLSv1.2 or higher using proprietary protocols.</p> <p>Monetra communicates using the PCI DSS-required protocols and cipher suites automatically using the default configuration. Please see Section 7.11.1: main.conf.</p> <p>Monetra performs full validation of the remote's TLS certificate. This feature cannot be disabled.</p>
<p>Encrypt cardholder data sent over end-user messaging technologies.</p>	<p>Monetra does not facilitate or support the use of end-user messaging technologies for cardholder data. While Monetra does have SMS support, such support is limited to sending things like payment receipt links and TOTP or reset codes for MFA.</p>
<p>Encrypt non-console administrative access.</p>	<p>The Monetra application does not provide or facilitate non-console administrative access to the server or PC. Integrators are responsible for ensuring non-console administrative access aligns with PCI DSS Requirement 8, including the use of strong cryptography.</p>
<p>Use multi-factor authentication for all personnel with non-console administrative access.</p>	<p>Monetra does not provide or facilitate administrative access to the server or PC.</p> <p>Installers are required to ensure multi-factor authentication is used for non-console administrative access to the CDE.</p>

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 Cryptographic key unlock for Monetra startup.
- Administrative Integrated Computer Chip (Smart Card) for HSM support.
- Base Cryptographic Key (encrypted) for backup and archive.
- Base Cryptographic Key (plain-text component) 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