Monetra ® Payment System

Data Security Shield™
Secure Storage and Billing System

Revision: 1.2
November 2009

Table of Contents

Secure Storage and Billing System Overview 4
- Introduction 4
- Storage Overview 4
- Billing Overview 4

Installation and Configuration 5
- New System Installation 5
- Upgrading Monetra to support DSS 5

Functional/Protocol Overview 7
- Introduction 7
  - Add Account 7
  - Edit Account 7
  - Delete Account 8
  - List Accounts 9
  - List History 9
  - Clear History 10
- Parameters Table 11

Usage Examples 12
- Add/Store Account in System 12
- Run Sale for Stored Account 12
- Add Account for Recurring Billing 12
- Add Account for Installment Billing 13
- Report/List of Stored/Recurring Transactions 13

Design and Implimentation Notes 15
- Exception Handling 15
- Dates and Time for Scheduling 15

Graphical User Interface Examples 16
- Monetra Manager: DSS Settings 16
- Monetra Client: Add Account 17
- Monetra Client: Edit Account 20

Document Version and Changes 22

1 Secure Storage and Billing System Overview

1.1 Introduction

The Monetra®Data Security Shield (Monetra DSS) feature provides a secure storage repository (using military-grade encryption) for retaining sensitive account information that might be needed for future transactions. Combined with the Recurring Billing System features, Monetra DSS enables businesses to easily and flexibly schedule recurring or installment payments to be processed automatically from within the Monetra Server application.

The ability to securely store sensitive account data and automatically perform recurring billing operations adds convenience for day-to-day business processes while maintaining full compliance with credit card industry security guidelines.


The DSS features are supported within all modern Monetra releases (v 7 or higher).

1.2 Storage Overview

The Monetra DSS provides an advanced and security-compliant method for storing and replacing sensitive account data with “sanitized” pseudo card numbers that can be used for future billing and transaction services. With Monetra DSS, sensitive card data can be removed from any POS or billing system and retained in a safe, security-compliant manner. Card data stored in the Monetra DSS is replaced in the POS system by a generic key, (or “token”), eliminating the 'sensitive' information that otherwise could be subject to compromise. The token is then used in place of the actual card number when performing transactions (such as uthorizations) for that card. When a transaction using that token is performed, Monetra DSS will internally replace the token with the corresponding card number, so the actual card data is never exposed.

1.3 Billing Overview

The Monetra Recurring Billing System (RBS) is built into the core of Monetra, offering merchants a completely integrated and secure way to establish and automatically process repetitive payment transactions. The Monetra RBM permits flexible billing cycles and intervals, and the convenience of scheduling and running both recurrent and installment-based transactions.

2 Installation and Configuration

The Secure Storage and Billing System features are delivered as part of the core functionality of all modern Monetra Server systems. To ensure proper software installation, please reference the most recent Monetra Installation and Monetra Configuration guides.


All documentation is located at http://www.monetra.com/documentation.html

2.1 New System Installation

If you are installing Monetra from scratch, please ensure you are installing the latest Monetra version (7 or higher). Note: When using the Monetra Installer utility, the latest version is listed under the 'Latest Ver' column.

Step 1.	Install the Monetra Payment Engine software following the Installation Guide.
Step 2.	Configure the DSS settings. Step 2a: Open the Monetra Manager and choose the 'advanced' icon. Step 2b: In the custom fields table ensure 'recurring_per_user'(1) is set to YES Step 2c: In the custom fields table ensure 'recurring_starting_bin'(2) is set to your range Step 2d: In the custom fields table ensure 'recurring_schedule_time' is set to the proper time for daily runs.


These settings can be found inside the **prefs.conf** canfiguration file. Alternately, if you use the Monetra Manger to configure your server then you can set these parameters there in the 'Advanced Configuration' section.

2.2 Upgrading Monetra to Support Monetra DSS

If you are upgrading Monetra to support the Monetra DSS then please ensure you are upgrading to the latest Monetra version (7 or higher). Note: When using the Monetra Installer utility, the latest version is listed under the 'Latest Ver' column.

Step 1.	Upgrade the Monetra Payment Engine using the Monetra Installer application.
Step 2.	Configure the Monetra DSS module to load: Step 2a: Open the Monetra Manager and choose the 'modules' icon. Step 2b: Place a check-mark (enable) the tran_recurring listing
Step 3.	Configure the module parameters. Step 3a: Open the Monetra Manager and choose the 'advanced' icon. Step 3b: In the custom fields table ensure 'recurring_per_user'(1) is set to YES

	Step 3c: In the custom fields table ensure 'recurring_starting_bin'(2) is set to your range.

KEY		DESCRIPTION
recurring_per_user	=	This key controls Monetra user access to the billing system. When set to 'Yes', only the Monetra user that added the transaction can use the Token.
recurring_starting_bin	=	This represents the BIN range that the Tokens will be created in.

3 Functional/Protocol Overview

3.1 Introduction

The Monetra DSS is a core Monetra feature. The functions can be programmatically called using the standard *Monetra Protocol* and communication methods (i.e. XML, Direct SSL, Direct IP etc.).

3.1.1 Add Account

The 'Add Account' function is used when adding a card to the storage system, or adding in a recurring transaction for the first time.

KEY		VALUE	TYPE	NOTES
username	=		A/N	Merchant Username associated with account.
password	=		A/N	Merchant Password associated with account.
action	=	admin	A	Denotes admin request type.
admin	=	recurringadd	A	Denotes type of administrative action (recurring).
type	=	store	A	Denotes sub-type of recurring action.
* See parameters table for additional keys that may be sent.				

Response: A standard Monetra response is returned similar to the below.

KEY		VALUE	NOTES
Code	=	AUTH	
Verbiage	=	Inserted into recurring list	
Token	=	Unique Token	Replacement number for the real/stored account.

3.1.2 Edit Account

The 'Edit Card Account' function is used when an account needs a parameter updated. Uses could include updating a billing ZIP code or replacing out the account number with a new card.

KEY		VALUE	TYPE	NOTES
username	=		A/N	Merchant Username associated with account.
password	=		A/N	Merchant Password associated with account.
action	=	admin	A	Denotes admin request type.
admin	=	recurringedit	A	Denotes type of administrative action (recurring).
type	=	store	A	Denotes sub-type of recurring action.

KEY		VALUE	TYPE	NOTES
token	=	[substitute number]	N	Req: Must pass a valid token.
account	=	[cardnumber]	N	Optional
expdate	=	[card_expire_date]	N	Optional
street	=	[billing street]	A/N	Optional
zip	=	[billing zipcode]	N	Optional
active	=	yes/no	A	Optional. Denotes if active or not.
CardHolderName	=		A	Optional
ClientRef	=		A/N	Optional. Client reference number.
Descr	=		A/N	Optional. Description of account.
* See parameters table for additional keys that may be sent.				

Response: A standard Monetra response is returned similar to the below.

KEY		VALUE	NOTES
Code	=	AUTH	
Verbiage	=	Inserted into recurring list	
Token	=	Unique Token	Replacement number for the real/stored account.

3.1.3 Delete Account

The 'Delete Card Account' function is used when an account needs to be removed.

KEY		VALUE	TYPE	NOTES
username	=		A/N	Req. Merchant Username associated with account.
password	=		A/N	Req. Merchant Password associated with account.
action	=	admin	A	Req. Denotes admin request type.
admin	=	recurringdel	A	Req. Denotes type of administrative action (recurring).
token	=	[substitute number]	N	Req: Must pass a valid token.

Response: A standard Monetra response is returned similar to the below.

KEY		VALUE	NOTES
Code	=	AUTH	
Verbiage	=	Recurring card deleted	

3.1.4 List Accounts

The 'List Account' function provides a convenient report/view of all stored accounts. Note: Once a card is entered for storage, the real account number can never be extracted. Only the card type and a truncated account number will be returned.

KEY		VALUE	TYPE	NOTES
username	=		A/N	Req. Merchant Username associated with account.
password	=		A/N	Req. Merchant Password associated with account.
action	=	admin	A	Req. Denotes admin request type.
admin	=	recurringlist	A	Req. Denotes type of administrative action (recurring).
type	=	store	A	Opt: The type of recurring transaction.
active	=	yes/no	A	Optional. Denotes if active or not.
token	=	[substitute number]	N	Opt: Must pass a valid token.
ClientRef	=		A/N	Opt: Client reference.
expdate_end	=		A/N	Opt: Ending date range for list.

Response: A standard comma separated Monetra response is returned similar to the one below.

```
token,type,active,cardtype,account,expdate,cardholdername,street,zip,descr,clientref,amount,frequency,
bdate,edate,installment_num,installment_total,last_run_id,last_success_date,last_run_date,
next_run_date,next_run_amount
```

```
1001000000000015,store,yes,VISA,XXXXXXXXX8881,1212,,123,32606,johns marine
parts,5674,,,,,,,0,,,,
1001000000000023,recurring,yes,VISA,XXXXXXXXXXXX8883,1212,,123,32606,State road
reality,34523,25.50,monthly,06-01-2007 00:00:00,,,,0,,,06-01-2007 00:00:00,25.50
```

3.1.5 List History

The 'List History' function provides a convenient report/view of all historical transactions that were run via the integrated billing system.

KEY		VALUE	TYPE	NOTES
username	=		A/N	Req. Merchant Username associated with account.
password	=		A/N	Req. Merchant Password associated with account.
action	=	admin	A	Req. Denotes admin request type.
admin	=	recurringhist	A	Req. Denotes type of administrative action (recurring).
type	=	store	A	Opt: The type of recurring transaction.

KEY		VALUE	TYPE	NOTES
token	=	[substitute number]	N	Opt: Must pass a valid token.
ClientRef	=		A/N	Opt: Client reference.
expdate_end	=		A/N	Opt: Ending date range for list.

The history report will only show scheduled transactions. If you store a token and run a Sale or Return against that token the transaction will be reported in the standard Monetra Unsettled/ History reports.

Response: A standard comma separated Monetra response is returned similar to the one below.

```
token,type,active,cardtype,account,expdate,cardholdername,street,zip,descr,clientref,amount,frequency,
bdate,edate,installment_num,installment_total,last_run_id,last_success_date,last_run_date,
next_run_date,next_run_amount
```

```
1001000000000015,store,yes,VISA,XXXXXXXXX8881,1212,,123,32606,johns marine
parts,5674,,,,,,,0,,,,
1001000000000023,recurring,yes,VISA,XXXXXXXXXXXX8883,1212,,123,32606,State road
reality,34523,25.50,monthly,06-01-2007 00:00:00,,,,0,,,06-01-2007 00:00:00,25.50
```

3.1.6 Clear Account

The 'Clear Account' function provides a way to remove (or help cleanup) the recurring history.

KEY		VALUE	TYPE	NOTES
username	=		A/N	Req. Merchant Username associated with account.
password	=		A/N	Req. Merchant Password associated with account.
action	=	admin	A	Req. Denotes admin request type.
admin	=	recurringclear	A	Req. Denotes type of administrative action (recurring).
type	=	store	A	Opt: The type of recurring transaction.
token	=	[substitute number]	N	Opt: Must pass a valid token.
ClientRef	=		A/N	Opt: Client reference.
expdate_end	=		A/N	Opt: Ending date range for list.

Response: A standard comma separated Monetra response is returned similar to the one below.

```
token,type,active,cardtype,account,expdate,cardholdername,street,zip,descr,clientref,amount,frequency,
bdate,edate,installment_num,installment_total,last_run_id,last_success_date,last_run_date,
next_run_date,next_run_amount
```

1001000000000015,store,yes,VISA,XXXXXXXXX8881,1212,,123,32606,johns marine parts,5674,,,,,,,0,,,,
1001000000000023,recurring,yes,VISA,XXXXXXXXXXXX8883,1212,,123,32606,State road reality,34523,25.50,monthly,06-01-2007 00:00:00,,,,0,,,06-01-2007 00:00:00,25.50

3.2 Parameters Table

PARAMETER		NOTES
Action	=	Admin
Admin	=	recurringadd|recurringedit|recurringdel recurringlist|recurringhist|recurringclear
Type	=	Store/recurring/installment (R add/ O edit/list)
Account	=	Account number on card (R add / O edit)
Expdate	=	Expiration date on card (R add / O edit)
Cardholdername	=	Name on card (O add/edit)
Street	=	Billing street address for card (O add/edit)
Zip	=	Billing zipcode for card (O add/edit)
Descr	=	Description for card (O add/edit)
Clientref	=	Client reference number
Token	=	Unique identifier for entry, returned from add (R edit/list/del/ history)
expdate_end	=	MMYY format. Used to retrieve cards which expire prior to this date (O list)
Active	=	Used to query for cards which hit an exception, or to reset the exception flag (O edit/list)
Hist_id	=	Used to query for specific transaction (O history)
Amount	=	For recurring, amount to charge. For installment, total amount, will be divided equally by installment_total (C add/ O edit)
Installment_total	=	Installment only, total number of payments to be made (C add/edit)
Frequency	=	Installment/Recurring : daily,weekly,biweekly,monthly,bimonthly,quarterly,semiannually,annually (C add/ O edit)
bdate	=	Date to start recurring/installment, if not specified, assume today. Meaningless after first transaction (C add/ O edit)
bdate	=	(In History/ClearHist, date range to pull/clear) (O history/clearhist)
edate	=	Date to end recurring/installment, if not specified, assume infinite. (C add/ O edit)
edate	=	(In History/ClearHist, date range to pull/clear) (C history/clearhist, must exist if bdate exists)

4 Usage Examples

4.1 Add/Store Account in System

Transaction

```
================
username=test_user
password=test123
action=admin
admin=recurringadd
type=store
account=4012888888881
expdate=1212
street=125 main street
zip=32606
clientref=jmf2394h
descr=johns marine parts
running account
================
```

Response

```
================
token=1001000000000015
code=auth
msoft_code=unknown
phard_code=unknown
verbiage=Inserted into
recurring list
================
```

4.2 Run Sale for Stored Account

Transaction

```
================
username=test_user
password=test123
action=sale
token=1001000000000015
amount=27.85
ptrannum=058745
comments=june-2009 billing
statement
```

Response

```
================
ttid=428753
code=AUTH
msoft_code=INT_SUCCESS
phard_code=SUCCESS
auth=TAS137
batch=376
item=6293
cardtype=visa
================
```

4.3 Add Account for Recurring Billing

Transaction

```
===============
username=test_user
password=test123
action=admin
```

Transaction Response

```
===============
Token=1001000000000031
code=AUTH
msoft_code=unknown
```

```
admin=recurringadd
type=recurring
account=4715088888888883
expdate=1212
street=123 main
zip=32606
bdate=06/01/2007
amount=25.50
frequency=monthly
clientref=sr34523
descr=State road reality
================
```

```
phard_code=unknown
verbiage=Inserted into
recurring list
================
```

4.4 Add Account for Installment Billing

Transaction

```
==============
username=test_user
password=test123
action=admin
admin=recurringadd
type=installment
account=4005519255555555
expdate=1212
street=123 main
zip=32606
bdate=06/01/2007
amount=1200.23
installment_total=12
frequency=monthly
clientref=pjr987658
descr=Porter johnson rental-
atlanta
================
```

Transaction Response

```
==============
Token=1001000000000049
code=AUTH
msoft_code=unknown
phard_code=unknown
verbiage=Inserted into
recurring list
================
```

4.5 Report/List of Stored/Recurring Transactions

Transaction

```
==============
username=test_user
password=test123
action=admin
```

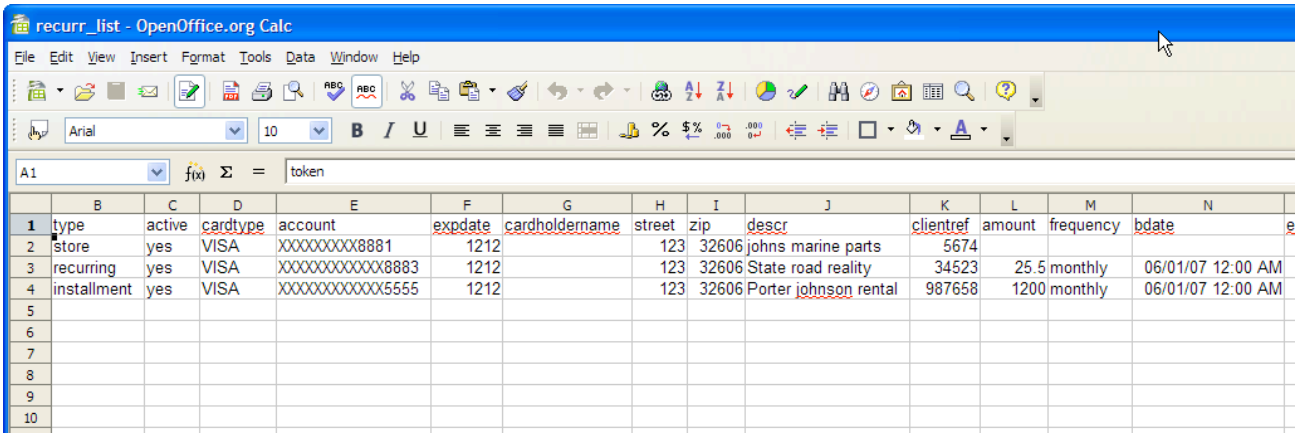
Transaction Response

```
==============
See response below.
================
```

```
admin=recurringlist
================
```

```
token,type,active,cardtype,account,expdate,cardholdername,street,zip,descr,clientref,amount,frequency,bdate,edate,installment_
num,installment_total,last_run_id,last_success_date,last_run_date,next_run_date,next_run_amount
1001000000000015,store,yes,VISA,XXXXXXXXX8881,1212,,123,32606,johns marine parts,5674,,,,,,0,,,,
1001000000000023,recurring,yes,VISA,XXXXXXXXXXXX8883,1212,,123,32606,State road
reality,34523,25.50,monthly,06-01- 2007 00:00:00,,,,0,,,06-01-2007 00:00:00,25.50
1001000000000031,installment,yes,VISA,XXXXXXXXXXXX5555,1212,,123,32606,Porter johnson
rental,987658,1200.00,monthly,06-01-2007 00:00:00,,1,12,0,,,06-01-2007 00:00:00,100.00
```

The report responses are returned in standard Monetra csv format, and can be imported and viewed with any standard spreadsheet application.



	B	C	D	E	F	G	H	I	J	K	L	M	N
1	type	active	cardtype	account	expdate	cardholdername	street	zip	descr	clientref	amount	frequency	bdate
2	store	yes	VISA	XXXXXXXX8881	1212		123	32606	johns marine parts	5674			
3	recurring	yes	VISA	XXXXXXXXXXX8883	1212		123	32606	State road reality	34523	25.5	monthly	06/01/07 12:00 AM
4	installment	yes	VISA	XXXXXXXXXXX5555	1212		123	32606	Porter johnson rental	987658	1200	monthly	06/01/07 12:00 AM
5													
6													
7													
8													
9													
10													

5 Design and Implimentation Notes

Described below are notes and tips from the Monetra developers, to help clarify exactly how the system functions. If there are any questions then please contact us via support@monetra.com

5.1 Exception Handling

There are times when a Recurring or installment transaction will hit an exception. For example the card date could expire, or it might DENY with insufficient funds. When an error condition hits, the affected transactions will be put under Exceptions.

If a transaction hits an exception, the following procedure should be used to get the transaction back on schedule.

- Issue an 'Edit' on the transaction, to fix offending field.
- Reset the 'Active' flag to 'yes'.

This will cause the transaction to be rescheduled for the next time the recurring module process transactions (which is once a day).

IMPORTANT: Any edit will cause a reschedule to occur for the transaction. Be aware of this when you change edit other parameters, such as frequency, and be careful you do not inadvertantly cause an account to be billed more than once within a given billing period.

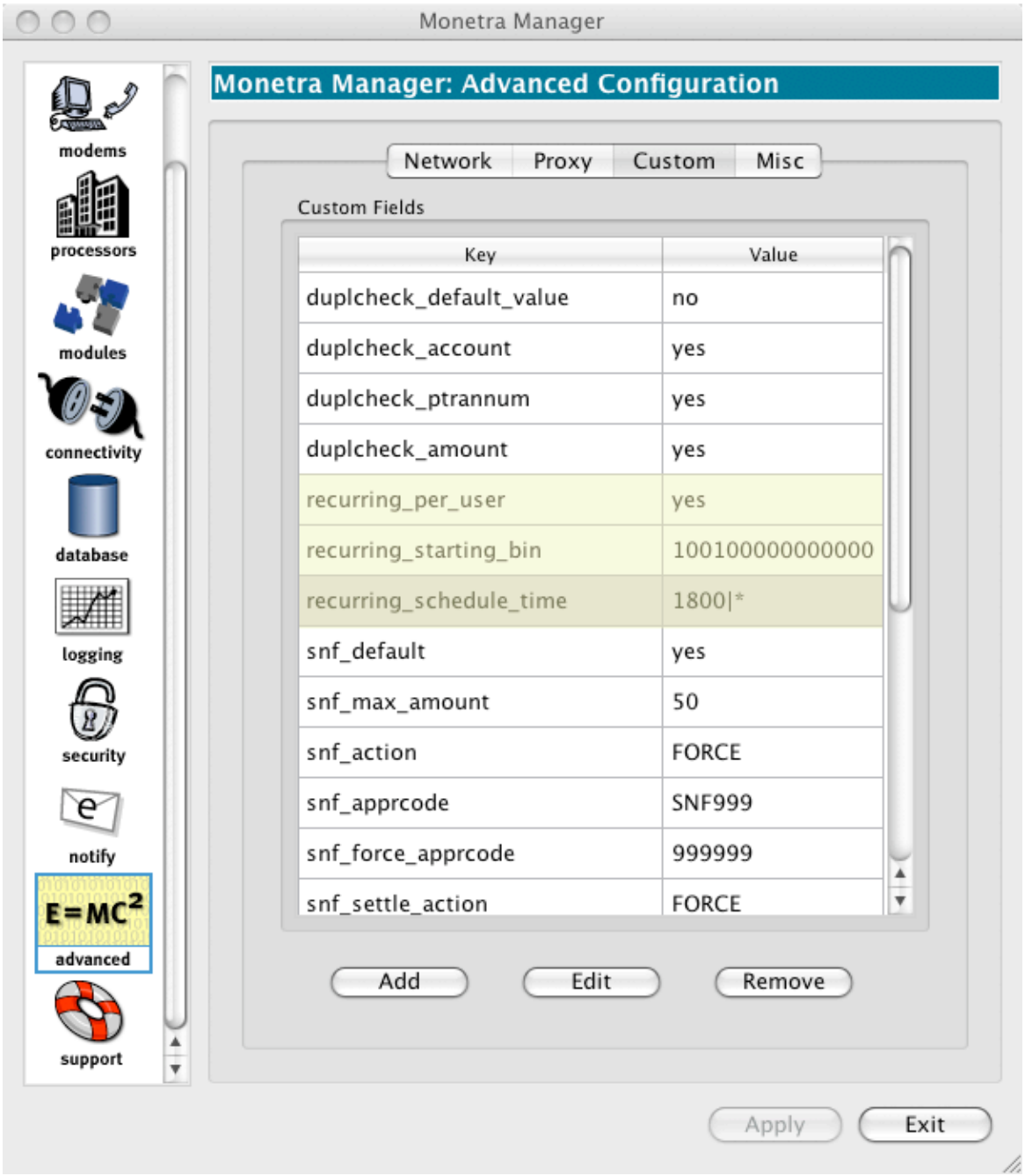
5.2 Dates and Time for Scheduling

The following frequencies are available for scheduling.

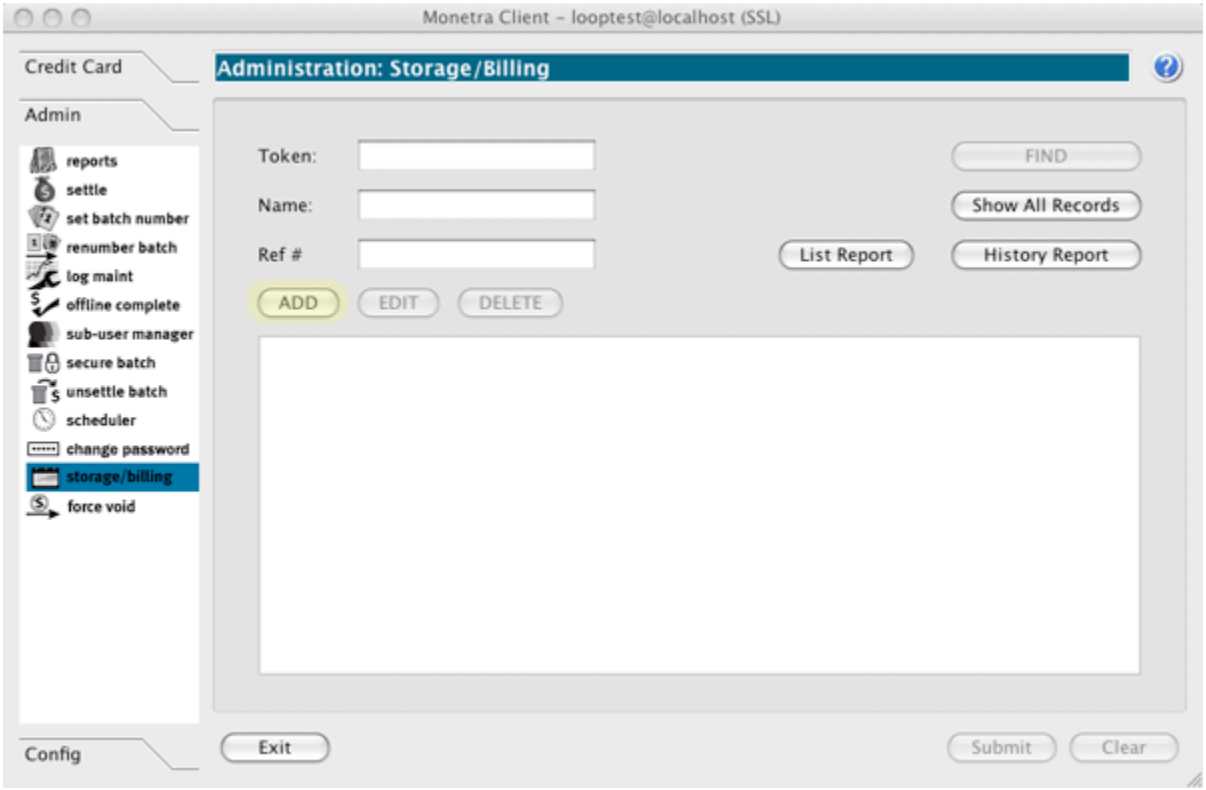
- Daily
- Weekly
- BiWeekly
- Monthly
- BiMonthly
- Quarterly
- SemiAnnuallly
- Annually

6 Graphical User Interface Examples

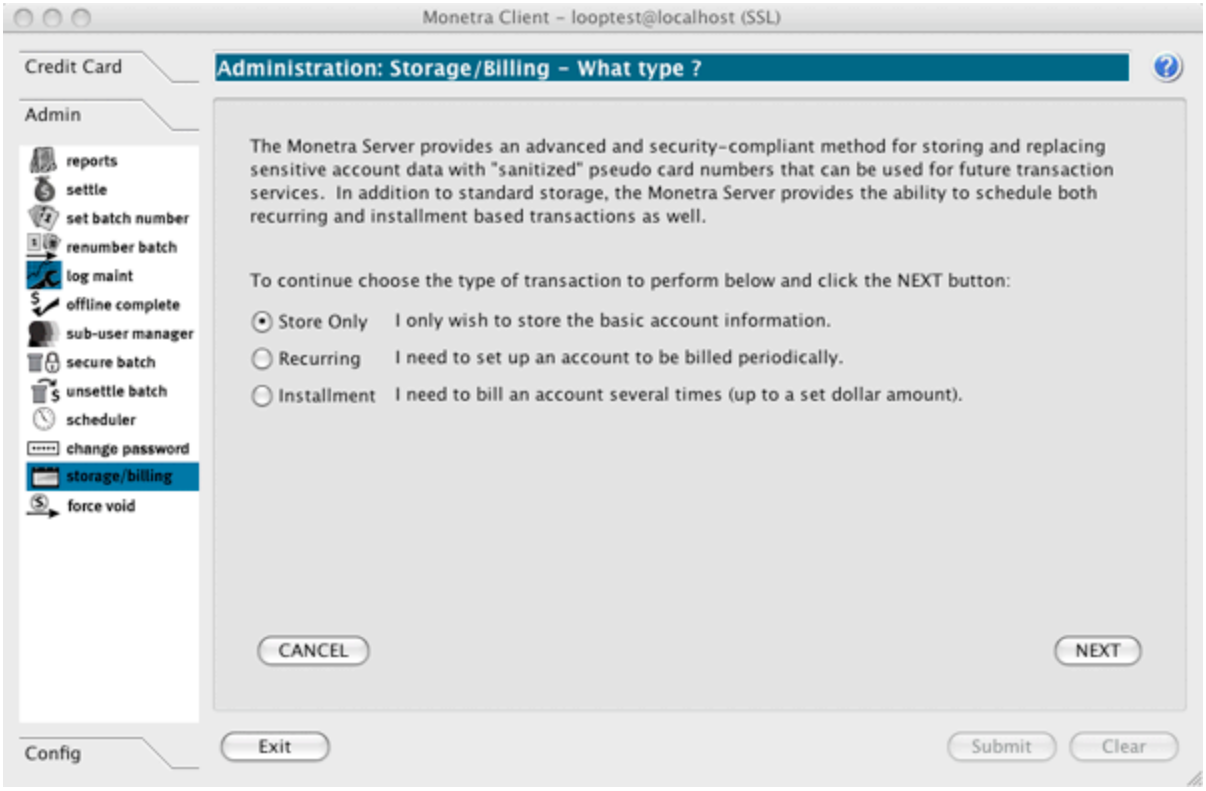
6.1 Monetra Manager: DSS Settings



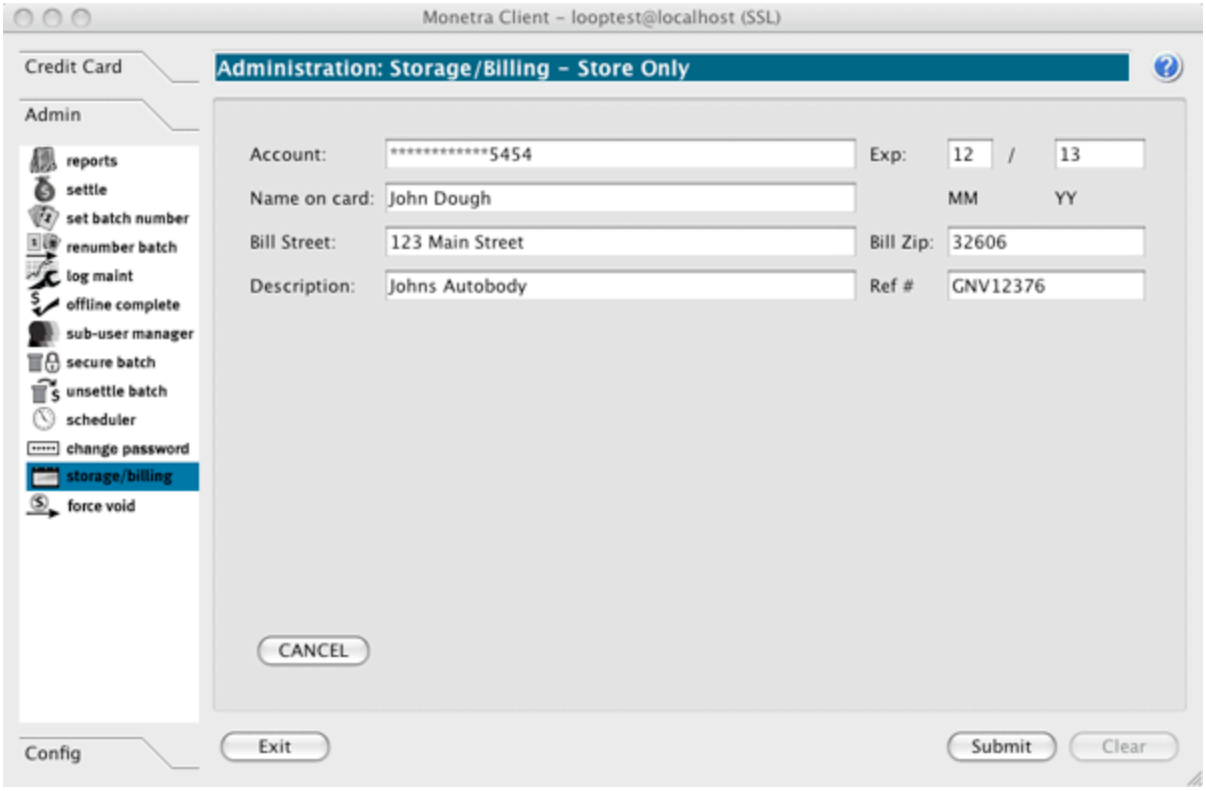
6.2 Monetra Client: Add Account



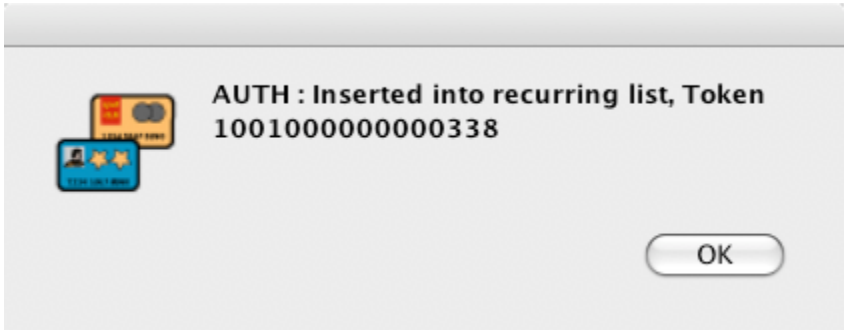
Step 1.	Open the Monetra Client application. Choose the 'Admin' section then the 'storage/billing' icon.
Step 2.	Click on the 'ADD' button.



Step 3.	Choose the type of storage. In this case we will use Store Only.
Step 4.	Click on the 'NEXT' button.

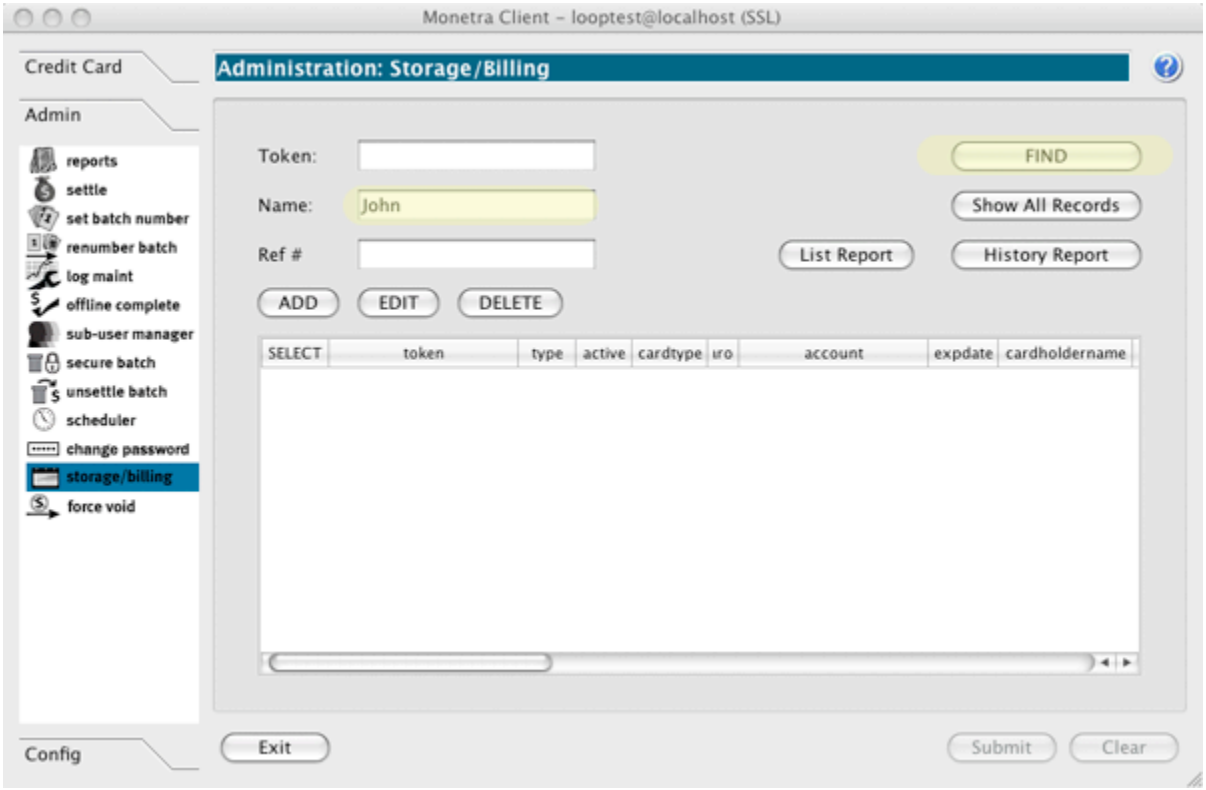


Step 5.	Enter all of the relevant information such as card account, billing street and zip codes etc..
Step 6.	Click on the 'Submit' button.

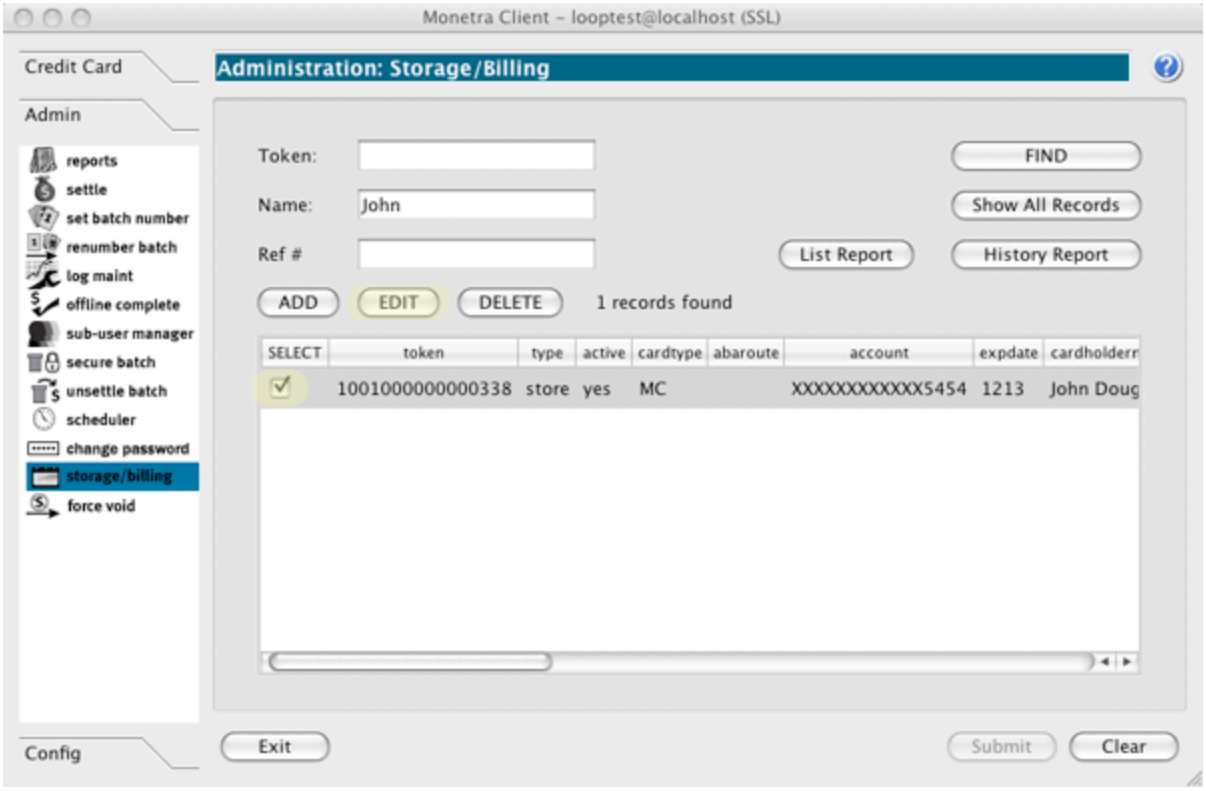


Thats It!. You should see an authorization message similar to the one above.

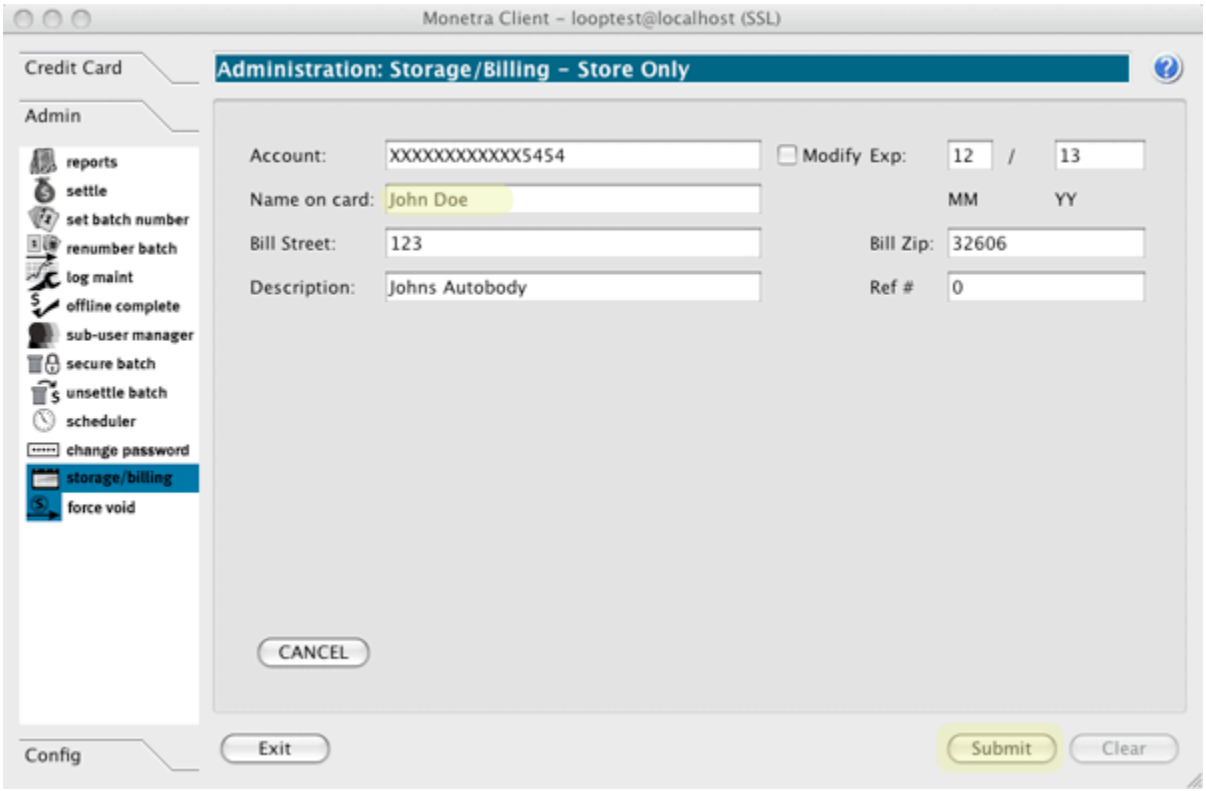
6.3 Monetra Client: Edit Account



Step 1.	Open the Monetra Client application. Choose the 'Admin' section then the 'storage/ billing' icon.
Step 2.	Enter the Token number, Name or Reference number of the account and click 'FIND'.



Step 3.	Place a checkmark in the table 'SELECT' field. next to the record you wish to edit.
Step 4.	Click on the 'EDIT' button.



Step 5.	Edit the desired field and click the 'Submit button'.

7 Document Version and Changes

Date	Version	Change
May, 2007	1.0	Initial version
Mar, 2008	1.1	Updated to enhance installation information
Nov, 2009	1.2	* Re-formatted document for web and print. * Updated for removal of module and migration to core Monetra functions. * Removed deprecated 'recurring=' instances. Added new combined 'admin' action names. * Added references to graphical interfaces.