Table of Contents

1 AdminAPI

1.1 Introduction

Swivel has developed an API, the **User Admin API**, to allow an external application to perform CRUD (Create, Read, Update, Delete) style operations on users. For most installations, these operations are performed by the User Synchronization job but for larger user populations synchronization may be impractical. The User Admin API addresses this need. In conjunction with the User Admin API, Swivel has also developed another API allowing external applications to perform the functions usually performed by a helpdesk operator from the admin console. These are typically day to day functions not viewed as part of the User Admin API such as user unlock, PIN reset etc. This is the Helpdesk API

There is additionally read-only functionality provided by the Reporting API that can be used to read the status (eg idle, locked) of user accounts.

1.2 PRINCIPLES

Following the principles behind the existing Agent API used for authorisation, both of the API?s are based around XML documents for both request and reply. The basic idea is to build a document containing details of operation(s) to be performed and the user(s) on which they are to be performed, submit it to PINsafe via HTTP and PINsafe will reply with a document detailing which operations succeeded and which, if any, failed. All operations via the API will be logged in the standard PINsafe log. Only authorised PINsafe Agents will be able to submit requests. This is in line with Agents used for authorisation. The API is case sensitive, the correct case shown in the included examples.

In order to use the Admin API, you must create a PINsafe Agent for the computer, or sub-net of computers, that will execute AdminAPI commands. Only the computer(s) defined by this Agent can create, read, update or delete users belonging to this Agent, and the Agent cannot read, update or delete users created by other repositories. The Agent can therefore be regarded as another type of repository. It is possible, but not recommended, to manipulate users created by a normal (e.g. XML or Active Directory) repository, by giving the Agent exactly the same name as the repository. Be aware if you do this, however, that any subsequent User Sync of the repository will potentially overwrite modifications made by the Agent. At present, any PINsafe Agent can act as an AdminAPI repository, but future versions of PINsafe will require you to explicitly enable the "Act as Repository" property of an Agent.

These restrictions do not apply to HelpdeskAPI Agents. Although only Agents can use these commands, it is possible to specify that the HelpdeskAPI command operates on a different repository, or on all repositories.

1.2.1 Sending an Admin API request

An Admin API request must be sent as an HTTP(S) request to the pinsafe application, either as a GET or POST request.

For a GET request, the format is

```
https://[swivelserver]:8080/pinsafe/AdminXML?xml=[request body]
```

For a POST request, the URL is

```
https://[swivelserver]:8080/pinsafe/AdminXML
```

and the XML request should form the content of the POST.

1.3 AdminXML API and Repositories

Users created through the API are assigned to a repository named after the Agent from which the Swivel server receives AdminXML commands. Only users belonging to this repository can be managed by AdminXML commands.

Swivel must create a repository for each Agent. The differences in how the repositories appear vary with Swivel version as follows:

3.9 onwards There is an option on each Agent to enable it to act as a repository and this appear in the User Administration Repository drop-down.

3.8 Only Agents that actually have created users appear in the User Administration Repository drop-down.

3.4 to 3.7 All Agents are listed in the User Administration Repository drop-down.

1.4 AdminXML API and User Syncs

Repositories managed through the AdminXML are usually entirely managed through the AdminXML without user syncs against external user repositories. If changes are made through the API, they may be overwritten by a user sync with a repository, such as a user being added through the API will be deleted if a user sync is made if that user is not present in the repository.

1.4.1 AdminXML workaround for standard repositories

If you need to use the AdminXML API on other repositories, there is a workaround by defining an Agent with exactly the same name as an existing Repository (case-sensitive). However, this is not recommended for write and edit commands, as any changes made using the API will be overwritten the next time a User Sync runs.

Read commands are safe, however, as they do not change the data. If you need to do this, but you already have an Agent that you do not want to rename, you can define multiple Agents on the same IP address, as long as the secret is different.

1.5 USER ADMIN API

An example User Admin request is show below; Note the AdminRequest version number should match the version number of the Swivel server

```
<?xml version="1.0" ?>
<AdminRequest secret="MyAdminAgent" version="3.4">
```

```
 <Create>
   <User name="bob">
     <Credentials pin="1234"/>
     <Groups>
       <Group name="EmailUsers"/>
     </Groups>
     <Policy changePin="true"/>
     <Rights dual="true" single="true"/>
     <Attributes>
       <Attribute name="email" value="bob@home"/>
     </Attributes>
   </User>
 </Create>
</AdminRequest>
```

The above example shows a request to create a new user called ?bob? and his associated account details. Bob is a dual and single channel user with a pin of ?1234? that he must change next time he authenticates.

The AdminRequest element contains a ?secret? attribute, here set to ?MyAdminAgent?, which has the same function as it does in the Agent API ? when coupled with the source IP address it proves that the Agent is authorised to access Swivel.

The version attribute is required, but is only checked to ensure the value is not greater than the maximum allowed. For software versions up to and including 4.0.5, the maximum allowed version is "3.97", but there is no problem if the version is less than the target version, even if it uses features not available in earlier versions. It is essential that version is a valid number, though, so it must be "3.97" rather than "3.9.7".

This request would be POSTed and the response received via HTTP to the AdminXML servlet on the Swivel server, usually to be found here : http://<ip.address.goes.here>:8080/pinsafe/AdminXML

The Create operation is one of several possible operations :

- Create ? create a new user.
- Delete ? delete a user.
- Read ? read details of an existing user.
- Reset ? reset user authentication credentials (Note: the user must have an alert transport to receive the new PIN).
- Sync - Synchronise Swivel with a repository
- Update ? update user details.

All of these operations can take a list of users so multiple users can be supplied in a single request and there are no specific ordering requirements although, obviously, users must have been Created before operations can be performed on them.

A breakdown of the individual operations follows.

1.5.1 Create

Create a new user. Supply any or all relevant details for the user. Refer to the section on sub-elements to see what can be included within the User element

```
 <Create>
   <User name="bob">
   <Credentials password=?itsasecret? pin="1234"/>
   <Attributes>
       <Attribute name="email" value="bob@home.com"/>
   </Attributes>
   <Groups>
       <Group name="EmailUsers"/>
   </Groups>
     <Policy changePin="true"/>
     <Rights dual="true" single="true"/>
     <String name="SMTP" destination="bob@home"/>
   </User>
 </Create>
```

1.5.2 Delete

Delete an existing user. Supply only the username, nothing else is required or allowed.

```
 <Delete>
   <User name="bob"/>
 </Delete>
```

1.5.3 Read

Read user details. Supply only the username, nothing else is required or allowed.

```
 <Read>
   <User name="bob"/>
 </Read>
```

If the user exists the response will include the all the users details setable by the API, apart from their credentials.

1.5.4 Reset

Reset user credentials. Supply only the username, nothing else is required or allowed. This creates new credentials for the user and sends them out via their alert transport, if they have one configured. This is the equivalent of pressing the RESEND button on the user-administration screen.

```
<Reset>
   <User name="bob"/>
```

```
</Reset>
```

1.5.5 Sync

This can be used to syncrhonise Swivel with a repository. This can be used in scenarios where an external application populates a repository and then needs to action changes made within that repository.

The element needs to include a Repository element with the the name of the repository, as named on the Swivel server.

For example

```
<Sync>
<Repository repository="ActiveDirectoryOne"/>
</Sync>
```

1.5.6 Update

Update user details. Supply any elements you wish to update. Refer to the section on sub-elements to see what can be included within the User element

```
 <Update>
   <User name="bob">
     <Attributes>
       <Attribute name="email" value="bob@home"/>
     </Attributes>
   </User>
 </Update>
```

1.5.7 Purge

Purge deleted users from the repository:

```
 <PurgeDeleted />
```

No content is permitted.

1.5.8 Message

Send an arbitrary message to a user. The message can be sent using either the user's alert transport (more usually) or their security strings transport.

```
 <Message>
   <User name="bob">
     <Alert text="message" />
   </User>
 </Message>
```

1.6 Valid User sub-elements and attributes

The user sub-element defines attributes for a new user when within a Create element or new attributes for an existing user when within an Update element Valid sub-elements and their permissible attributes are as follows.

1.6.1 Attributes

This element defines custom attributes for the user. These can be used as message destinations (e.g. email and phone), alternative identifiers when logging in, for searching the user list, or simply as additional information.

The Attributes element should contain 1 or more Attribute elements. Each of these must have a name and value attribute. The name must match the name of an attribute defined in the Repository -> Attributes configuration. The Attribute names should be used directly when using the API: there is no need to map Attributes to local names as with Active Directory / LDAP.

example

```
<Update>
  <User name="bob">
    <Attributes>
      <Attribute name="phone" value="447817360285"/>
      <Attribute name="email" destination="bob@home"/>
    </Attributes>
  </User>
</Update>
```

1.6.2 Alert & String

Deprecated as of 3.9.6

In 3.9.6 and later there is no need to explicitly set transport and alert transports and destinations. This will be determined by setting attributes for the user and by allocating the user to a group configured to use the required transport.

Alert and String refer to the Alert transport and String transport to be used by the user for sending system alerts (Alert) and dual-channel security strings (String).

A name must be supplied and this must match the name on a transport defined on the Swivel server. The destination must be appropriate to that transport

name Required, the Alert or String transport name. destination Required, the destination attribute required by the transport.

eg

```
<Update>
  <User name="bob">
    <Strings name="AQL" destination="447817360285"/>
    <Alert name="SMTP" destination="bob@home"/>
  </User>
</Update>
```

1.6.3 Credentials

Used to set PIN, Password or both. As agents are trusted, the credentials are not tested for conformance to PIN policies.

password Optional, the new password. pin Optional, the new PIN.

```
<Update>
  <User name="bob">
     <Credentials password=?itsasecret? pin="1234"/>
  </User>
</Update>
```

1.6.4 Groups

Groups element is used to specify to what group a user is a member. Group membership is used to determine certain rights within Swivel, for example what authentication agents and NASs via which they can authenticate. Group membership can be specified by using a <Groups> element contains a <Group> element for each group the user is a member.

All group must be included in all updates/creates

```
<Update>
  <User name="bob">
   <Groups>
       <Group name="DualChannelUsers"/>
       <Group name="EmailUsers"/>
       <Group name="AQLUsers"/>
     </Groups>
 </User>
</Update>
```

1.6.5 Policy

There are a number of polices that can be set within a policy element. These relate to the individual and overrule any overall server policies. To apply the policy set the policy to true, to remove the policy set to false

changePin Optional, user must change PIN next authentication

disabled Optional, user account is disabled

locked Optional, user account is locked (until 4.1)

lockedByAdmin Optional, user account is locked by administrator (4.2 onwards)

deleted Optional, user account is marked as deleted

inactive Optional, user account is inactive

lockedPinExpired Optional, user account is locked because the PIN has expired (4.2 onwards)

lockedFailures Optional, user account is locked because of too many authentication failures (4.2 onwards)

pinNeverExpires Optional, user PIN never expires

eg to unlock bob

```
<Update>
  <User name="bob">
   <Policy locked="false" />
 </User>
</Update>
```

NOTE: "locked" does not work in version 4.2: use "lockedByAdmin" instead. The "locked" attribute will be reinstated as a synonym for "lockedByAdmin" in future releases.

1.6.6 Rights

This element can be use to allocate the user rights within Swivel. **Note** that a user cannot be give Administrator rights via this interface

- dual Optional, user can use dual channel.
- helpdesk Optional, user is a helpdesk operator.
- pinless Optional, user is PINless.
- single Optional, user can use single channel.
- swivlet Optional, user can use the Mobile Phone Client or Swivlet.

Rights are added by setting the attribute to true and removed by setting to false.

eg to allow bob to use single channel but to remove helpdesk rights.

```
<Update>
```

```
  <User name="bob">
  <Rights single="true" helpdesk="false"/>
 </User>
</Update>
```

1.6.7 Oath

This element can be used to assign a previously-imported token to a user.

for example

```
<Update>
  <User name="bob">
    <Oath SerialNumber="12345678" />
  </User>
</Update>
```

1.7 Responses

Swivel responses to requests are similar in format. They reflect the command that has been submitted with a FAIL result should any of the commands not been executed.

For example:

```
<?xml version="1.0"? >
<AdminResponse>
 <Create>
   <User name="ann"/>
 </Create>
 <Read>
   <User name="ann">
     <Alert/>
     <Credentials/>
     <Groups/>
     <Policy pinNeverExpires="true" disabled="true"/>
     <Rights dual="true" single="true"/>
     <String/>
   </User>
 </Read>
</AdminResponse>
```

The above shows a response to an AdminRequest where user ?ann? was successfully created and read back.

Successful Create or Update sub-elements (Alert, Credentials?) are not returned, so we can not see from the response which were supplied but they all succeeded.

Obviously the account is only partially populated as ann has no groups, alert or string transport. Any elements that failed would have been returned containing the error ?FAIL? and the cause of the failure logged in the Swivel log. Credentials are never returned for security reasons.

```
<?xml version="1.0"?>
<AdminResponse>
 <Create>
   <User name="sid">FAIL</User>
 </Create>
</AdminResponse>
```

The above response shows a failure to create user ?sid? presumably because the account already exists. The full error will be shown in the Swivel server log.

```
<?xml version="1.0"?>
<HelpdeskResponse>
 <Update>
   <User name="ann"/>
 </Update>
</HelpdeskResponse>
```

```
The above is a Helpdesk response to a successful update.
```

1.8 Error Handling

1.8.1 PARSE ERRORS

Parse errors prevent execution of the request. Parse errors are typically caused by incorrect document structure and missing or invalid attributes; they indicate an application fault. In the event of a parse error, your reply will be a ParseError:

```
<ParseError>
 <Result>FAIL</Result>
 <Error>ADMIN_ERROR_UNSUPPORTED_ATTRIBUTE</Error>
</ParseError>
```

A list of errors can be found later in this document.

1.8.2 EXECUTION ERRORS

Execution errors can occur if a request was successfully parsed but part of the execution failed for some reason:

<HelpdeskResponse>

```
 <Reset>
   <User name="bob">FAIL</User>
 </Reset>
```

</HelpdeskResponse>

Execution errors can occur due to application faults i.e. trying to create a user that already exists or trying to send security strings to a user who has no transport set or doesn?t exist.

They can also occur if a more serious database fault occurs. In both cases, a FAIL is returned for the element in question and the real cause of the error logged in the Swivel log.

1.8.3 POSSIBLE PARSE ERRORS

Error	Meaning
ADMIN_ERROR_DOCUMENT_MALFORMED	The document supplied, while possibly valid XML, wasn?t a valid API request.
ADMIN_ERROR_INVALID_START_DATE	An invalid start date was supplied for a report.
ADMIN_ERROR_MISSING_DESTINATION	The destination attribute is required for Transports.
ADMIN_ERROR_MISSING_NAME	The name attribute was missing.
ADMIN_ERROR_MISSING_START_DATE	The start date was missing for a report.
ADMIN_ERROR_UNKNOWN_REPOSITORY	The supplied repository doesn?t exist.
ADMIN_ERROR_UNSUPPORTED_ATTRIBUTE	An unsupported attribute was found.
ADMIN_ERROR_UNSUPPORTED_VERSION	This should be no higher than the API version number supported by the target server. Up to and including software version 4.0.5 the maximum value is "3.97".
ADMIN_ERROR_XML	General server side failure; consult the Swivel log for more details.
AGENT_ERROR_UNAUTHORIZED	

The agent is not authorized to access Swivel

1.9 The Difference Between Admin XML and Agent XML

Admin XML is used when a program needs to make changes to the user database, for example, to add and remove users, or change their details.

Agent XML is used when a program needs to authenticate users to Swivel. It also has functions for changing PIN and other features, but does not have the ability to change user details in the Swivel database.

Both APIs can be used from the same program, and they both require that an Agent is configured on the Swivel server. You can use the same Agent for both APIs, but to use the Admin API to manage users, the Agent must have "Can act as Repository" set to "Yes".

.

2 MS SQL Queries How To Guide

2.1 Overview

When a MS SQL database of Swivel information is used it contains some information that is not available through the Swivel Administration Console. The majority of MySQL queries can be run against the MS SQL database with some exceptions. This document outlines some of the information that can be queried.

The **Audit Log** is where Swivel maintains an activity log for users. For more information on the Audit Log feature within Swivel, see the Audit Log How To Guide. For information on producing reports through the Swivel Administration console see Reporting How to guide.

For information on querying the Internal Swivel database see Reporting Using Agent-XML How to Guide

For information on MySQL database queries see MySQL Queries How To Guide. For information on using MS SQL as a database see MS SQL Database How To Guide

2.2 Prerequisites

- MS SQL Database;
- Audit log requires Swivel 3.4 or higher (for SQL queries on Swivel versions prior to Swivel 3.4 see the note at the bottom of the document).

2.3 Performing MS SQL Queries

Refer to the MS SQL documentation for performing MS SQL queries.

2.3.1 Some differences between MySQL and MS SQL queries

The equivalent function to NOW() in MS SQL Server is GETDATE(). However, SQL Server doesn't support the INTERVAL command either, so the DATEDIFF is used instead.

2.3.2 Querying individual Repositories

Each Repository may be queried my name or Repository ID. To find a Repository ID user:

SELECT A ID, B NAME FROM PINSAFEL

To query by NAME the following is used:

AND PINSAFEL.B = <NAME>

To query by ID the following is used:

AND I = <ID>

Example

Users who have never logged on (Successfully), for all repositories

USE pinsafe_rep; SELECT H Username FROM PINSAFEJ WHERE G NOT IN (SELECT DISTINCT A FROM PINSAFEN WHERE C=0)

Users who have never logged on (Successfully), for repositories <NAME>, Replace <NAME> with the name of the repository.

SELECT H Username FROM PINSAFEJ INNER JOIN PINSAFEL WHERE PINSAFEJ.I = PINSAFEL.A WHERE PINSAFEJ.G NOT IN (SELECT DISTINCT A FROM PINSAFEN WHERE C=0) AND PINSAFEL.B = <NAME>

Users who have never logged on (Successfully), for repository <ID>, Replace <ID> with the ID of the repository.

SELECT H Username FROM PINSAFEJ WHERE G NOT IN (SELECT DISTINCT A FROM PINSAFEN WHERE C=0) AND I = <ID>

2.4 MS SQL Queries

2.4.1 Count the number of users who have never logged in over the last 30 days but created more than 30 days ago

SELECT COUNT(*) FROM PINSAFEJ U JOIN PINSAFEN A1 ON U.G = A1.A AND A1.C = 3 LEFT OUTER JOIN PINSAFEN A2 ON U.G = A2.A AND A2.C=0 WHERE (A1.D < DATEADD(day, -30, GETDATE())) AND (A2.D IS NULL OR A2.D < DATEADD(day, -30, GETDATE()))

2.4.2 Show the number of users who have never logged in over the last 30 days but created more than 30 days ago

SELECT U.H FROM PINSAFEJ U JOIN PINSAFEN A1 ON U.G = A1.A AND A1.C = 3 LEFT OUTER JOIN PINSAFEN A2 ON U.G = A2.A AND A2.C=0 WHERE (A1.D < DATEADD(day, -30, GETDATE())) AND (A2.D IS NULL OR A2.D < DATEADD(day, -30, GETDATE()))

2.5 Troubleshooting

Are we able to isolate which regional Swivel server the user has logged in on from the database?

No not though SQL queries. However this information is available in the Swivel logs and obtainable through a log parser. Depending on how the Swivel agent is configured, there is a 'source' option in the Agent XML. This stores the IP information of the user logging on and there is a field in the database to store this information.

Locked user account list shows some users as not locked

The locked user count will report not only those that are flagged as locked but those accounts that have more than the number of failed authentications. When the user who has exceeded the maximum login attempts but whose account is not marked locked, next tries to login, the account will be marked as locked.

3 MySQL Queries How To Guide

3.1 Overview

The MySQL Swivel database contains some information that is not available through the Swivel Administration Console. This document outlines some of the information that can be queried.

The **Audit Log** is where Swivel maintains an activity log for users. For more information on the Audit Log feature within Swivel, see the Audit Log How To Guide. For information on producing reports through the Swivel Administration console see Reporting How to guide

For information on querying the Internal Swivel database see Reporting Using Agent-XML How to Guide

For information on querying a MS SQL database see MS SQL Queries How To Guide

For database schema information, in order to write your own queries, see Database Schema

3.2 Prerequisites

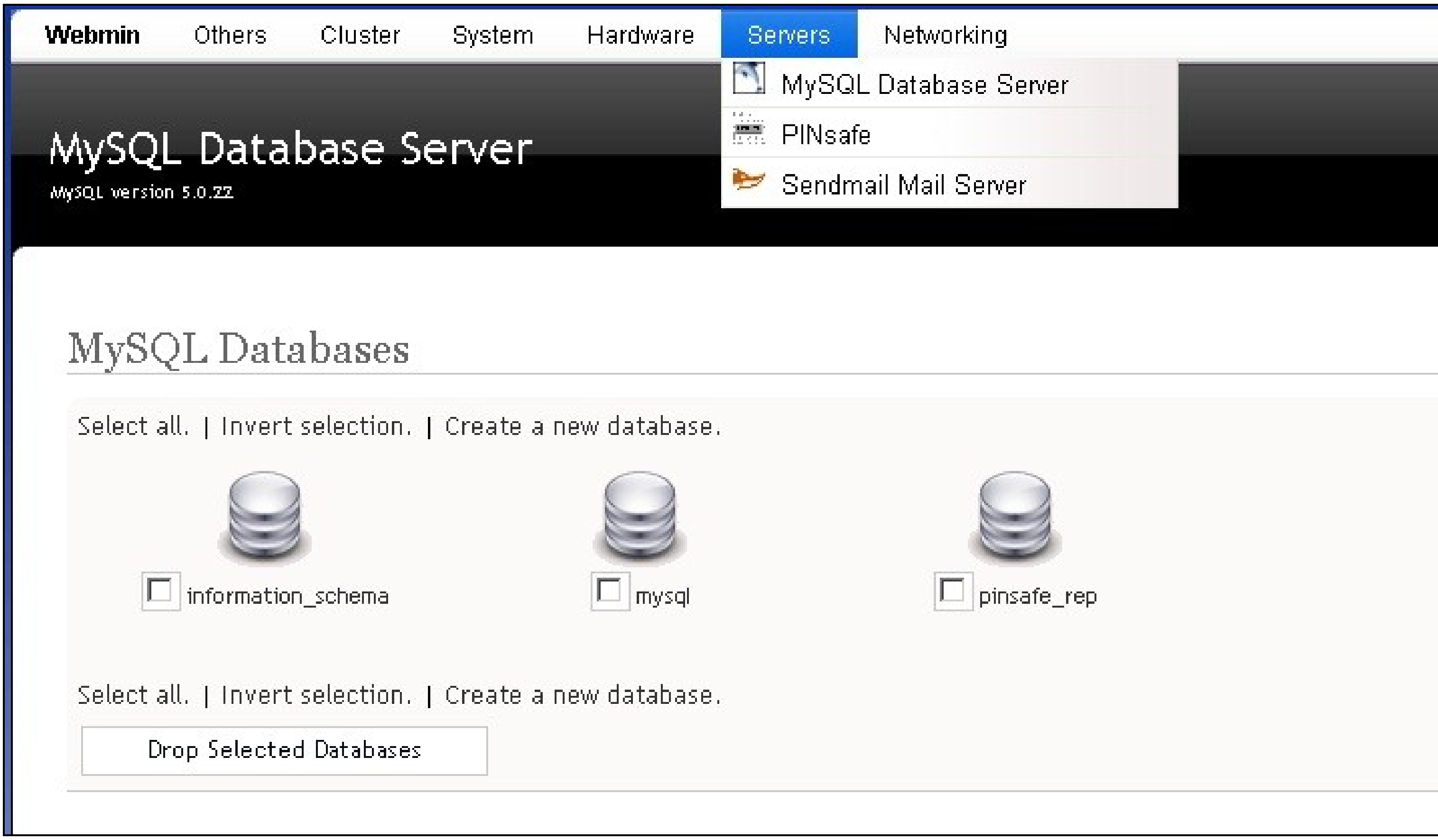
- SQL Database;
- Audit log requires Swivel 3.4 or higher (for SQL queries on Swivele versions prior to 3.4 see the note at the bottom of the document).

3.3 Performing MySQL Queries

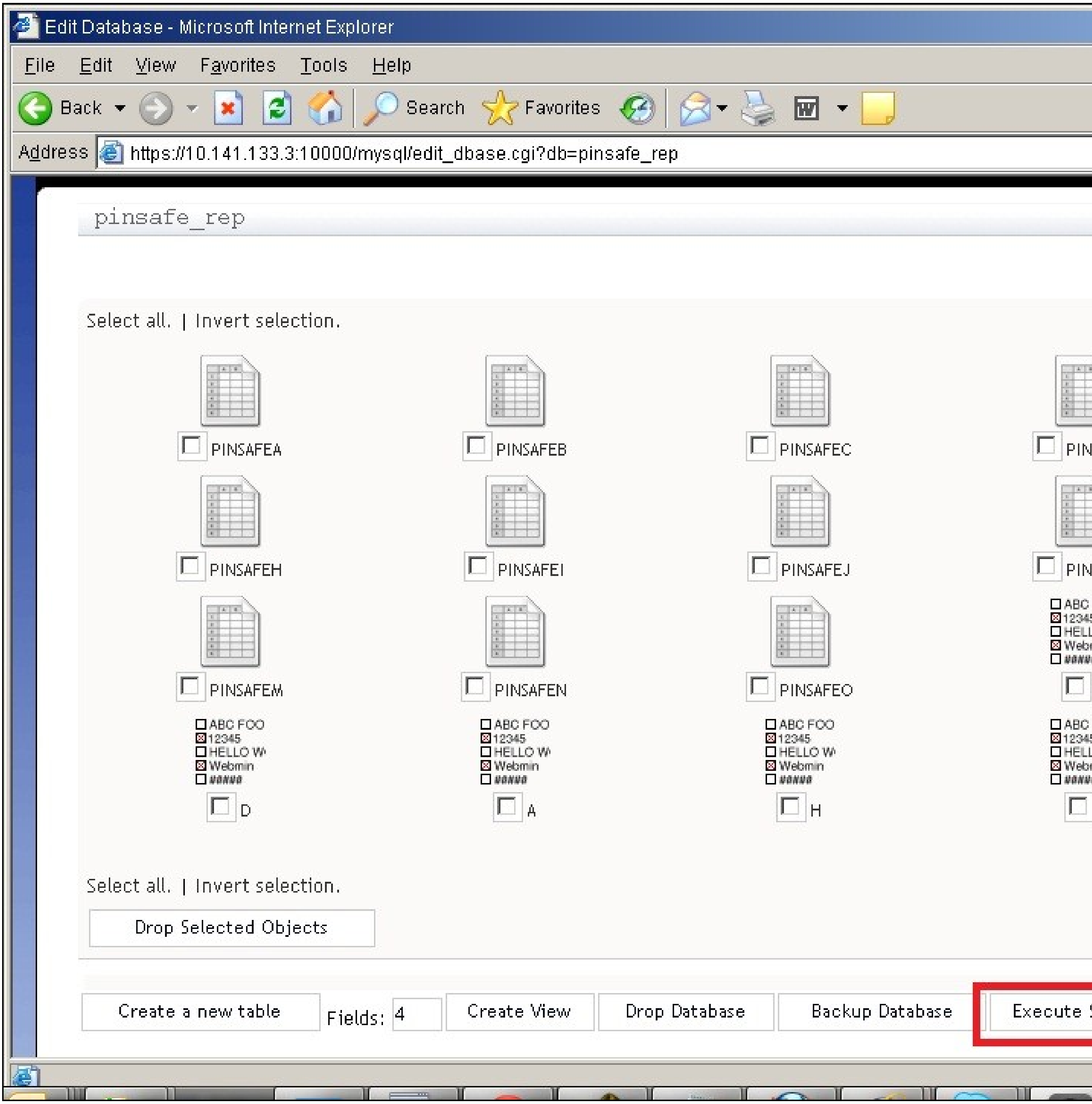
The MySQL database can be queried through the MySQL command line or through the Webmin interface. depending on the installation, the database name may be pinsafe or pinsafe_rep.

3.3.1 MySQL Queries on a Swivel appliance using Webmin

To view the MySQL database on an appliance, log into webmin (https://<pinsafe_server>:10000). Go to Servers -> MySQL Database Server and select the Swivel database by clicing on the icon, this will either be named pinsafe or pinsafe_rep. If you have both, pinsafe_rep will be the active one.



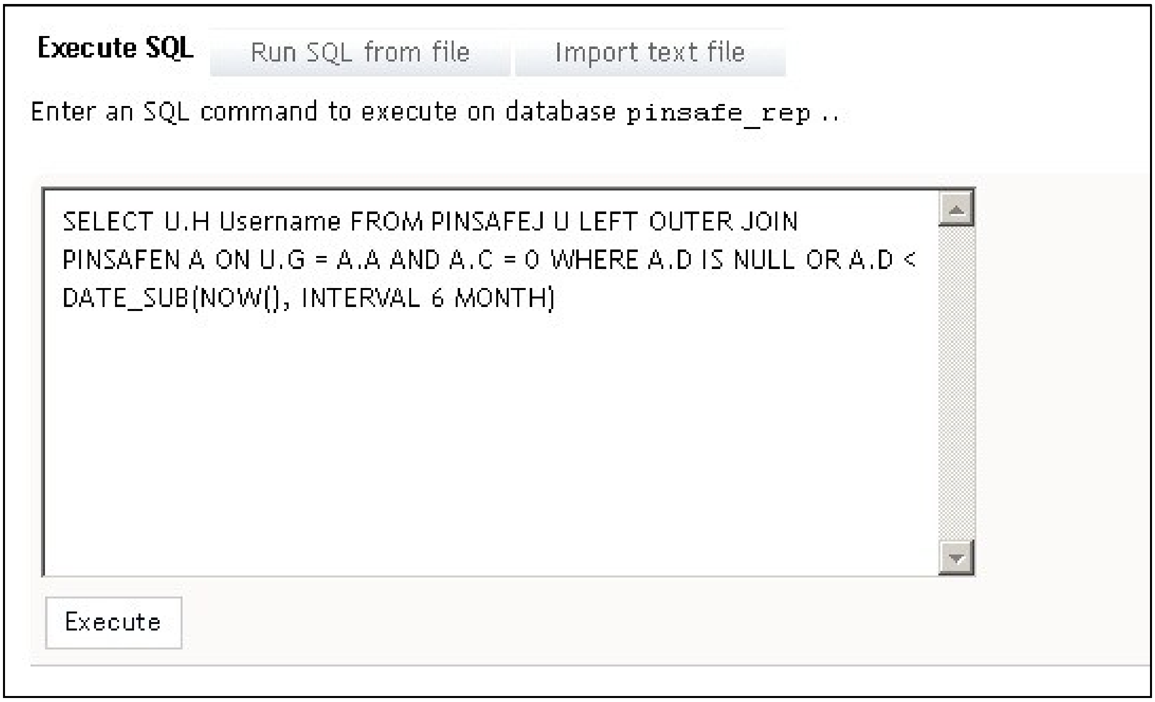
Click on the Execute SQL button.



Address https://10.141.133.3:10000/mysql/edit_dbase.cgi?db=pinsafe_rep

Fields: 4

Enter one of the below queries then click on Execute. For further information on see SQL commands in Webmin



Enter an SQL command to execute on database pinsafe_rep ..

```
SELECT U.H Username FROM PINSAFEJ U LEFT OUTER JOIN
PINSAFEN A ON U.G = A.A AND A.C = 0 WHERE A.D IS NULL OR A.D <
DATE_SUB(NOW(), INTERVAL 6 MONTH)
```

3.3.2 Querying individual Repositories

Each Repository may be queried by name or Repository ID. To find a Repository ID user:

SELECT A ID, B NAME FROM PINSAFEL

To query by NAME the following is used:

AND PINSAFEL.B = <NAME>

To query by ID the following is used:

AND I = <ID>

Example

Users who have never logged on (Successfully), for all repositories

USE pinsafe_rep; SELECT H Username FROM PINSAFEJ WHERE G NOT IN (SELECT DISTINCT A FROM PINSAFEN WHERE C=0)

Users who have never logged on (Successfully), for repositories <NAME>, Replace <NAME> with the name of the repository.

SELECT H Username FROM PINSAFEJ INNER JOIN PINSAFEL WHERE PINSAFEJ.I = PINSAFEL.A WHERE PINSAFEJ.G NOT IN (SELECT DISTINCT A FROM PINSAFEN WHERE C=0) AND PINSAFEL.B = <NAME>

Users who have never logged on (Successfully), for repository <ID>, Replace <ID> with the ID of the repository.

SELECT H Username FROM PINSAFEJ WHERE G NOT IN (SELECT DISTINCT A FROM PINSAFEN WHERE C=0) AND I = <ID>

3.4 MySQL Queries

3.4.1 List users who never have performed a successful login

The following query lists all users that have never (successfully) logged on to PINsafe:

SELECT H Username FROM PINSAFEJ WHERE G NOT IN (SELECT DISTINCT A FROM PINSAFEN WHERE C=0)

3.4.2 List users who have not had a login in a defined time period

The following query lists all users that have not logged in in the last 3 months from the current date and time (obviously, the period can be varied), The query NOW() specifies the current time going back the required time period, to query going back to the start of the day use DATE():

SELECT U.H Username FROM PINSAFEJ U LEFT OUTER JOIN PINSAFEN A ON U.G = A.A AND A.C = 0 WHERE A.D IS NULL OR A.D < DATE_SUB(NOW(), INTERVAL 3 MONTH)

3.4.3 List users who have logged in over the last 30 days

SELECT U.H Username, A1.D CreationTime, A2.D LoginTime FROM PINSAFEJ U INNER JOIN PINSAFEN A1 ON U.G = A1.A AND A1.C = 3 LEFT OUTER JOIN PINSAFEN A2 ON U.G = A2.A AND A2.C = 0 WHERE A2.D > DATE_SUB(NOW(), INTERVAL 30 DAY);

3.4.4 List users who have logged in over the last 3 months

SELECT U.H Username, R.B Repository FROM PINSAFEJ U INNER JOIN PINSAFEL R ON U.I = R.A LEFT OUTER JOIN PINSAFEN A ON U.G = A.A AND A.C = 0 WHERE A.D IS NULL OR A.D < DATE_SUB(NOW(), INTERVAL 3 MONTH)

3.4.5 List users who have logged in over the last 3 months and show alert transport email address

SELECT U.H Username, R.B Repository, T.A Email FROM PINSAFEJ U INNER JOIN PINSAFEL R ON U.I = R.A LEFT OUTER JOIN PINSAFEA T ON U.G = T.C LEFT OUTER JOIN PINSAFEN A ON U.G = A.A AND A.C = 0 WHERE A.D IS NULL OR A.D < DATE_SUB(NOW(), INTERVAL 3 MONTH)

3.4.6 List users who have logged in over the last 3 months and show strings transport email address

SELECT U.H Username, R.B Repository, T.A Email FROM PINSAFEJ U INNER JOIN PINSAFEL R ON U.I = R.A LEFT OUTER JOIN PINSAFEH T ON U.G = T.C LEFT OUTER JOIN PINSAFEN A ON U.G = A.A AND A.C = 0 WHERE A.D IS NULL OR A.D < DATE_SUB(NOW(), INTERVAL 3 MONTH

3.4.7 List users who have logged in over the last 3 months and show email address (version 3.9.1 onwards)

SELECT U.H Username, R.B Repository, A.C Email FROM PINSAFEJ U INNER JOIN PINSAFEL R ON U.I = R.A LEFT OUTER JOIN PINSAFEP A ON U.G = A.A AND A.B = 'email' LEFT OUTER JOIN PINSAFEN A ON U.G = A.A AND A.C = 0 WHERE A.D IS NULL OR A.D < DATE_SUB(NOW(), INTERVAL 3 MONTH)

3.4.8 List last login date and time

The following query lists the last date/time that each user logged in (the oldest login time first - add DESC at the end for most recent first):

SELECT U.H Username, A.D LoginTime FROM PINSAFEJ U INNER JOIN PINSAFEN A ON U.G = A.A AND A.C = 0 ORDER BY LoginTime

3.4.9 List last login date and time by repository

The following query lists the last date/time that each user logged in from a specified repository listed by login time.

SELECT U.H Username, A.D LoginTime FROM PINSAFEJ U INNER JOIN PINSAFEN A ON U.G = A.A AND A.C = 0 INNER JOIN PINSAFEL R ON U.I = R.A AND R.B = '<repos_name>' ORDER BY LoginTime

Replace <repos_name> in the above query with the name of the repository.

3.4.10 List username and repository

SELECT U.H Username, R.B FROM PINSAFEJ U JOIN PINSAFEN A1 ON U.G = A1.A AND A1.C = 3 LEFT OUTER JOIN PINSAFEN A2 ON U.G = A2.A AND A2.C=0 JOIN PINSAFEL R ON U.I = R.A WHERE (A1.D < DATE_SUB(NOW(), INTERVAL 30 DAY)) AND (A2.D IS NULL OR A2.D < DATE_SUB(NOW(), INTERVAL 180 DAY))

3.4.11 Count number of users in each group

SELECT R.B, G.A, COUNT(U.G)

```
FROM PINSAFEJ U
JOIN PINSAFEL R ON U.I=R.A
JOIN PINSAFEI G ON U.G=G.B
GROUP BY R.B, G.A
ORDER BY R.B, G.A
```

3.4.12 List number of users in each group

SELECT R.B, G.A, U.C

```
FROM PINSAFEJ U
JOIN PINSAFEL R ON U.I=R.A
JOIN PINSAFEI G ON U.G=G.B
ORDER BY R.B, G.A, U.C
```

3.4.13 List last login date and time and show users who have never logged in

The following query shows all users and their last login date. Any dates listed as NULL means that the user has never logged in, or never changed their PIN.

```
SELECT U.H Username, A1.D LastLogin, A2.D PINChange FROM PINSAFEJ U LEFT OUTER JOIN PINSAFEN A1 ON U.G = A1.A AND A1.C = 0
LEFT OUTER JOIN PINSAFEN A2 ON U.G = A2.A AND A2.C = 1
```

3.4.14 List login times for a given user

```
SELECT I Username, E DateTime FROM PINSAFEM WHERE A = 0 AND I = ? ORDER BY E
```

Replace the ? with the username in quotes.

3.4.15 List unsuccessful login times for a given user

```
SELECT I Username, E DateTime FROM PINSAFEM WHERE A = 14 AND I = ? ORDER BY E
```

Replace the ? with the username in quotes.

3.4.16 List login times for all users

```
SELECT I Username, E DateTime FROM PINSAFEM WHERE A = 0 ORDER BY I, E
```

3.4.17 List unsuccessful login times for all users

```
SELECT I Username, E DateTime FROM PINSAFEM WHERE A = 14 ORDER BY I, E
```

3.4.18 Count the number of Logins

This gives the total number of authentications in the audit table. (see note above for Audit Log):

```
SELECT COUNT(*) FROM PINSAFEM WHERE A=0
```

3.4.19 Count the number of Failed Logins

This gives the total number of failed authentications in the audit table. (see note above for Audit Log):

```
SELECT COUNT(*) FROM PINSAFEM WHERE A=14
```

3.4.20 Count the number of successful logins over last 30 days

The following query shows the number of successful login attempts over the last 30 days (see note above for Audit Log):

```
SELECT COUNT(*) FROM PINSAFEM WHERE A=0 AND E > DATE_SUB(NOW(), INTERVAL 30 DAY)
```

3.4.21 Count the number of failed logins over last 30 days

The following query shows the number of successful login attempts over the last 30 days (see note above for Audit Log):

```
SELECT COUNT(*) FROM PINSAFEM WHERE A=14 AND E > DATE_SUB(NOW(), INTERVAL 30 DAY)
```

3.4.22 Count the number of logins and login failures per day over the last 30 days

```
select D1 AS Date, L AS Logins, F AS Failures from (select Date(e) D1, count(*) L from pinsafem where a=0 group by date(e)) AS Logins left outer join
(select Date(e) D2, count(*) F from pinsafem where a=14 group by date(e)) AS Failures on d1=d2;
```

3.4.23 Count the number of users who have logged in over the last 30 days

```
SELECT COUNT(*) FROM PINSAFEJ U INNER JOIN PINSAFEN A1 ON U.G = A1.A AND A1.C = 3 LEFT OUTER JOIN PINSAFEN A2 ON U.G =
A2.A AND A2.C = 0 WHERE A2.D > DATE_SUB(NOW(), INTERVAL 30 DAY);
```

3.4.24 Count the number of times each user has logged in

The following query counts the number of times each user has logged in (see Audit Log note above):

```
SELECT I Username, Count(*) Count FROM PINSAFEM WHERE A = ? GROUP BY I
```

Replace ? by 4 for unlocked, 5 for locked, 2 for self-reset, 6 for admin reset and 0 for successful login.

To look for a specific user, change "GROUP BY I" to "WHERE I = 'name'"

3.4.25 Count the users number of unsuccessful logins and self resets

The following query will give the number of self-reset and unsuccessful logins since the last successful login:

```
SELECT H Username, B FailCount, F ResetCount FROM PINSAFEJ
```

to look for a particular user, add "WHERE H = 'name'"

3.4.26 List the users who have reset their PIN from a particular date

(Includes ChangePIN, ResetPIN and admin reset/resend of the PIN)

```
SELECT DISTINCT PINSAFEJ.H FROM PINSAFEJ INNER JOIN PINSAFEN ON PINSAFEJ.G = PINSAFEN.A WHERE (PINSAFEN.C = 1 OR
PINSAFEN.C = 2 OR PINSAFEN.C=6) AND PINSAFEN.D > '2011-11-18 10:30'
```

The report can be customised as follows

The **DISTINCT** option allows each user to be listed only once for the combination of resets.

ChangePIN **PINSAFEN.C = 1**

Self Reset **PINSAFEN.C = 2**

Admin reset **PINSAFEN.C = 6**

3.4.27 List the users who have used ChangePIN from a particular date

```
SELECT PINSAFEJ.H FROM PINSAFEJ INNER JOIN PINSAFEN ON PINSAFEJ.G = PINSAFEN.A WHERE PINSAFEN.C = 1 AND PINSAFEN.D >
'2013-12-23 16:30'
```

3.4.28 List the users who have used Self Reset from a particular date

```
SELECT PINSAFEJ.H FROM PINSAFEJ INNER JOIN PINSAFEN ON PINSAFEJ.G = PINSAFEN.A WHERE PINSAFEN.C = 2 AND PINSAFEN.D >
'2013-12-23 16:30'
```

3.4.29 List the users who have had an Admin reset from a particular date

```
SELECT PINSAFEJ.H FROM PINSAFEJ INNER JOIN PINSAFEN ON PINSAFEJ.G = PINSAFEN.A WHERE PINSAFEN.C = 6 AND PINSAFEN.D >
'2013-12-23 16:30'
```

3.4.30 List user date of creation and last login

The following query will show when the account was created and when the user last connected per day:

```
SELECT U.H Username, A1.D CreationTime, A2.D LoginTime FROM PINSAFEJ U INNER JOIN PINSAFEN A1 ON U.G = A1.A AND A1.C = 3 LEFT
OUTER JOIN PINSAFEN A2 ON U.G = A2.A AND A2.C = 0
```

3.4.31 List users who have been created in the last 30 days

Te following query lists all users which have been created in the last 30 days:

```
SELECT U.H FROM PINSAFEJ U JOIN PINSAFEN A ON U.G=A.A AND A.C=3 WHERE A.D > DATE_SUB(NOW(), INTERVAL 30 DAY)
```

3.4.32 List users who have been created in the last month

Te following query lists all users which have been created in the last month:

```
SELECT U.H FROM PINSAFEJ U JOIN PINSAFEN A ON U.G=A.A AND A.C=3 WHERE A.D > DATE_SUB(NOW(), INTERVAL 1 MONTH)
```

3.4.33 Count the number of users who have never logged in

The following query reports the number of users who have never logged in:

```
SELECT COUNT(*) FROM PINSAFEJ WHERE G NOT IN (SELECT A FROM PINSAFEN WHERE C=0)
```

3.4.34 Count the number of users who have never logged in over the last 30 days

The following query logs the number of users who have not logged in during the last 30 days:

```
SELECT COUNT(*) FROM PINSAFEJ U LEFT OUTER JOIN PINSAFEN A ON U.G = A.A AND A.C=0 WHERE A.D IS NULL OR A.D <
DATE_SUB(NOW(), INTERVAL 30 DAY)
```

3.4.35 Count the number of users who have never logged in over the last 180 days but created more than 30 days ago

Since the above query includes new users, who might not have had a chance to log in yet, the following query logs users who were created over 30 days ago, but have not logged in during the last 30 days:

```
SELECT COUNT(*) FROM PINSAFEJ U JOIN PINSAFEN A1 ON U.G = A1.A AND A1.C = 3 LEFT OUTER JOIN PINSAFEN A2 ON U.G = A2.A AND
A2.C=0 WHERE (A1.D < DATE_SUB(NOW(), INTERVAL 30 DAY)) AND (A2.D IS NULL OR A2.D < DATE_SUB(NOW(), INTERVAL 180 DAY))
```

3.4.36 List the number of users who have never logged in over the last 180 days but created more than 30 days ago

Since the above query includes new users, who might not have had a chance to log in yet, the following query logs users who were created over 30 days ago, but have not logged in during the last 30 days:

```
SELECT U.H Username FROM PINSAFEJ U JOIN PINSAFEN A1 ON U.G = A1.A AND A1.C = 3 LEFT OUTER JOIN PINSAFEN A2 ON U.G = A2.A
AND A2.C=0 WHERE (A1.D < DATE_SUB(NOW(), INTERVAL 30 DAY)) AND (A2.D IS NULL OR A2.D < DATE_SUB(NOW(), INTERVAL 180 DAY))
```

3.4.37 List users connected today who have not previously logged in

This query will show users who connected today, but have not connected before.

```
SELECT U.H Username, A1.D CreationTime, A2.D LoginTime FROM PINSAFEJ U INNER JOIN PINSAFEN A1 ON U.G = A1.A AND A1.C = 3 INNER
JOIN PINSAFEN A2 ON U.G = A2.A AND A2.C = 0 WHERE A2.D > DATE_SUB(NOW(), INTERVAL 1 DAY) AND NOT EXISTS(SELECT G FROM
PINSAFEM A3 WHERE A3.G=U.H AND A3.E < DATE_SUB(NOW(), INTERVAL 1 DAY))
```

3.4.38 lists all the Users Ids, email address, created date, last login, last pin change and repository name

```
SELECT U.H Username, U.E ReposName, T.A Email, A1.D CreateDate, A2.D LoginDate, A3.D PINChange FROM PINSAFEJ U LEFT OUTER JOIN
PINSAFEA T ON U.G = T.C LEFT OUTER JOIN PINSAFEN A1 ON U.G = A1.A AND A1.C = 3 LEFT OUTER JOIN PINSAFEN A2 ON U.G = A2.A AND
A2.C = 0 LEFT OUTER JOIN PINSAFEN A3 ON U.G = A3.A AND A3.C = 1
```

3.4.39 List PIN expiry for users

```
SELECT U.H USERNAME, ADDDATE(GREATEST(COALESCE(A1.D, '2000-01-01'), COALESCE(A2.D, '2000-01-01'), COALESCE(A3.D,
'2000-01-01')), 60) EXPIRY_DATE FROM PINSAFEJ U LEFT OUTER JOIN PINSAFEN A1 ON U.G = A1.A AND A1.C = 1 LEFT OUTER JOIN
PINSAFEN A2 ON U.G = A2.A AND A2.C = 6 LEFT OUTER JOIN PINSAFEN A3 ON U.G = A3.A AND A2.C = 3
```

Replace 60 with the actual PIN validity period.

3.4.40 List all Disabled accounts

```
SELECT U.H Name FROM PINSAFEJ U JOIN PINSAFEC S ON U.G = S.C AND S.B = 0 WHERE S.D = 1
```

3.4.41 List all users with inactive, locked, disabled and deleted state

```
SELECT U.H UID, R.B REPOS, S1.D DISABLED, S2.D LOCKED, S3.D DELETED, S4.D INACTIVE FROM PINSAFEJ U JOIN PINSAFEL R ON
U.I=R.A LEFT OUTER JOIN PINSAFEC S1 ON U.G=S1.C AND S1.B=0 LEFT OUTER JOIN PINSAFEC S2 ON U.G=S2.C AND S2.B=1 LEFT OUTER
JOIN PINSAFEC S3 ON U.G=S3.C AND S3.B=4 LEFT OUTER JOIN PINSAFEC S4 ON U.G=S4.C AND S4.B=5
```

3.4.42 List all users with inactive, locked, disabled and deleted state and show creation, login and PIN change date

```
SELECT U.H UID, R.B REPOS, S1.D DISABLED, S2.D LOCKED, S3.D DELETED, S4.D INACTIVE, A1.D CREATED, A2.D LOGGEDIN, A3.D
PINCHANGE FROM PINSAFEJ U JOIN PINSAFEL R ON U.I=R.A LEFT OUTER JOIN PINSAFEC S1 ON U.G=S1.C AND S1.B=0 LEFT OUTER JOIN
PINSAFEC S2 ON U.G=S2.C AND S2.B=1 LEFT OUTER JOIN PINSAFEC S3 ON U.G=S3.C AND S3.B=4 LEFT OUTER JOIN PINSAFEC S4 ON
U.G=S4.C AND S4.B=5 LEFT OUTER JOIN PINSAFEN A1 ON U.G=A1.A AND A1.C=3 LEFT OUTER JOIN PINSAFEN A2 ON U.G=A2.A AND
A2.C=0 LEFT OUTER JOIN PINSAFEN A3 ON U.G=A3.A AND A3.C=1
```

3.5 Queries on versions prior to Swivel 3.4

For versions of Swivel earlier than 3.4, the table PINSAFED performed a similar function to that now performed by PINSAFEN, i.e. recording the latest date/time of various activities.

SELECT H Username FROM PINSAFEJ WHERE G NOT IN (SELECT DISTINCT A FROM PINSAFED WHERE C=1)

Note that the code for login on PINSAFED is C=1, rather than C=0 for PINSAFEN. Therefore, the query to list last login date and time is

SELECT U.H Username, A.D LoginTime FROM PINSAFEJ U INNER JOIN PINSAFED A ON U.G = A.A AND A.C = 1 ORDER BY LoginTime

3.6 Known Issues

When running reports to list locked users, some users may be listed multiple times due to duplicate locks in the database, however the lock count and the number of locks displayed in the Swivel Administration Console will display the correct value for number of locked users.

3.7 Troubleshooting

Are we able to isolate which regional Swivel server the user has logged in on from the database?

No not though SQL queries. However this information is available in the Swivel logs and obtainable through a log parser. Depending on how the Swivel agent is configured, there is a 'source' option in the Agent XML. This stores the IP information of the user logging on and there is a field in the database to store this information.

Locked user account list shows some users as not locked

The locked user count will report not only those that are flagged as locked but those accounts that have more than the number of failed authentications. When the user who has exceeded the maximum login attempts but whose account is not marked locked, next tries to login, the account will be marked as locked.

4 Reporting How to guide

5 Overview

The Swivel Administration Console allows a limited number of reports to be generated and scheduled. This document outlines some of the information that can be queried through the reporting.

The **Audit Log** is where Swivel maintains an activity log for users. For more information on the Audit Log feature within Swivel, see the Audit Log How To Guide.

For information on querying the Internal Swivel database see Reporting Using Agent-XML How to Guide

For information on querying a MySQL database see MySQL Queries How To Guide and for a MS SQL database see MS SQL Queries How To Guide

Also see ReportingAPI

6 Prerequisites

Swivel 3.8 - Reporting

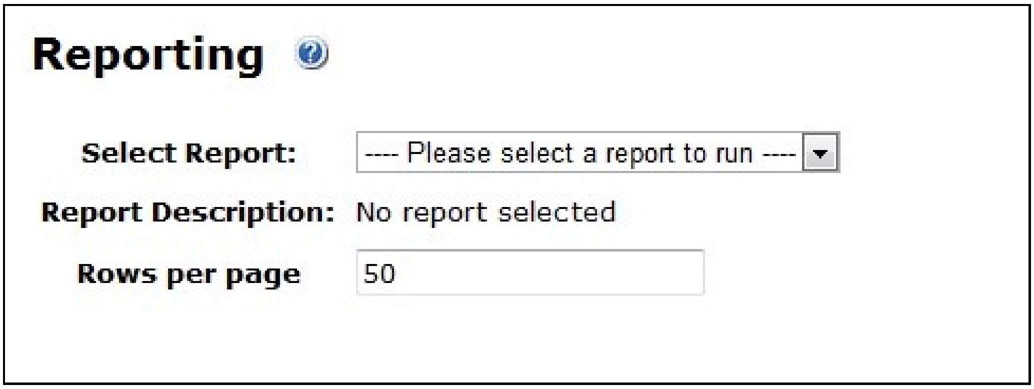
Swivel 3.9 - Scheduled Reporting

7 Reporting access and restrictions

Swivel reports from a global database. Where access is restricted for Helpdesk users to specific repositories and groups it may be ncessary to disable the reporting function for Helpdesk users. However reports can be produced by Administrative level access users creating scheduled reporting based on the required groups and making those reports available to the relevant Helpesk users managing those groups such as through automated emails.

8 Producing Instant Reports

From the Swivel Administration console select Reporting then Instant.



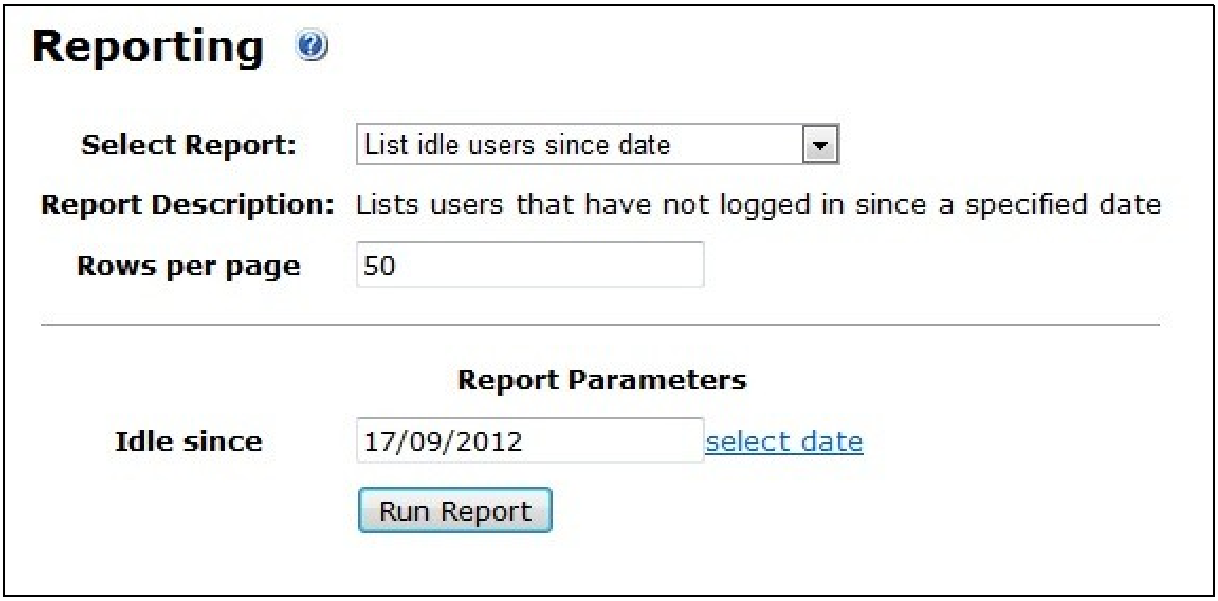
Reporting

Select Report: ---- Please select a report to run ----

Report Description: No report selected

Rows per page: 50

Depending on the report, additional parameters may be selected.



Reporting

Select Report: List idle users since date

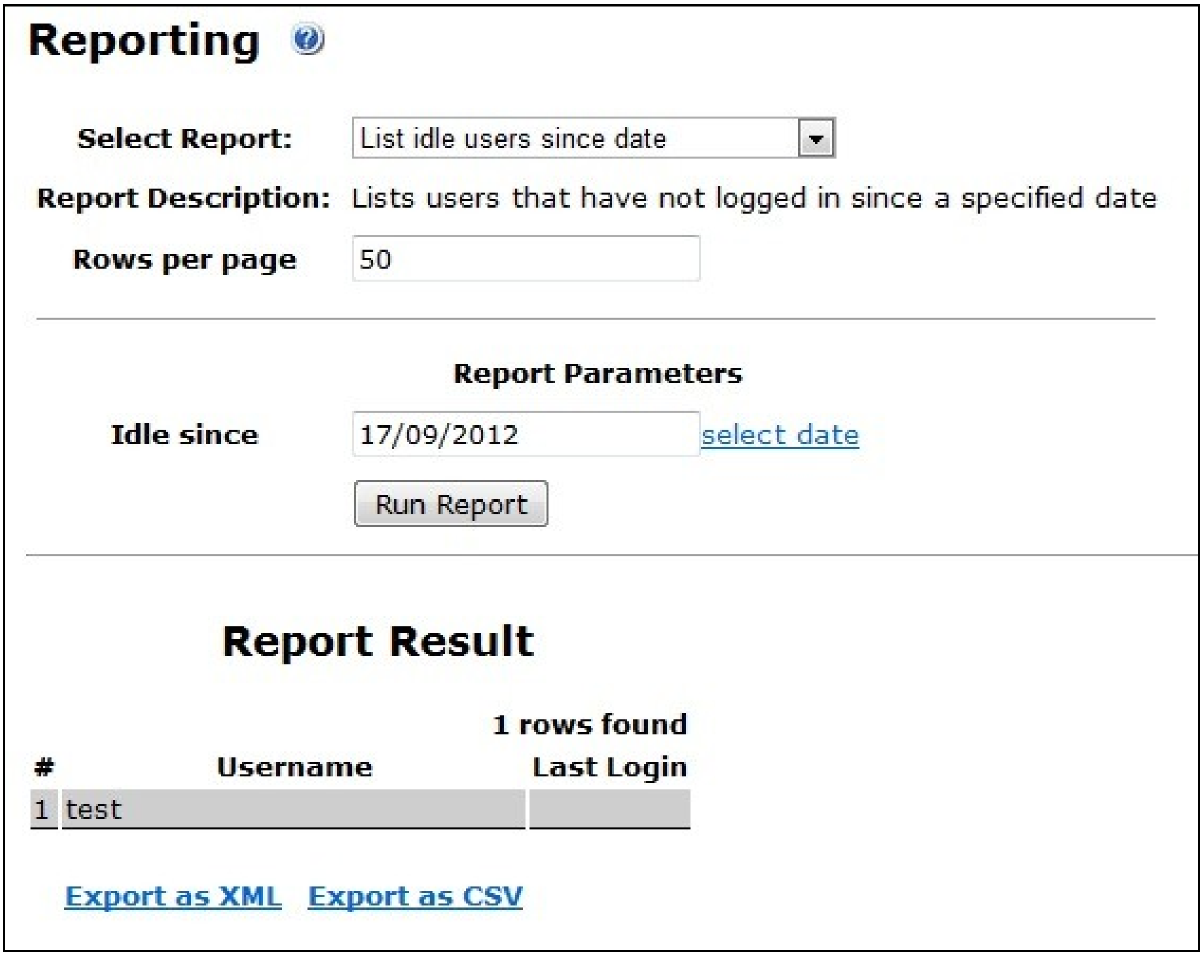
Report Description: Lists users that have not logged in since a specified date

Rows per page: 50

Report Parameters

Idle since: 17/09/2012 select date

Click on Run Report to generate data.



Reporting

Select Report:	List idle users since date
Report Description:	Lists users that have not logged in since a specified date
Rows per page	50

Report Parameters

Idle since	17/09/2012

Report Result

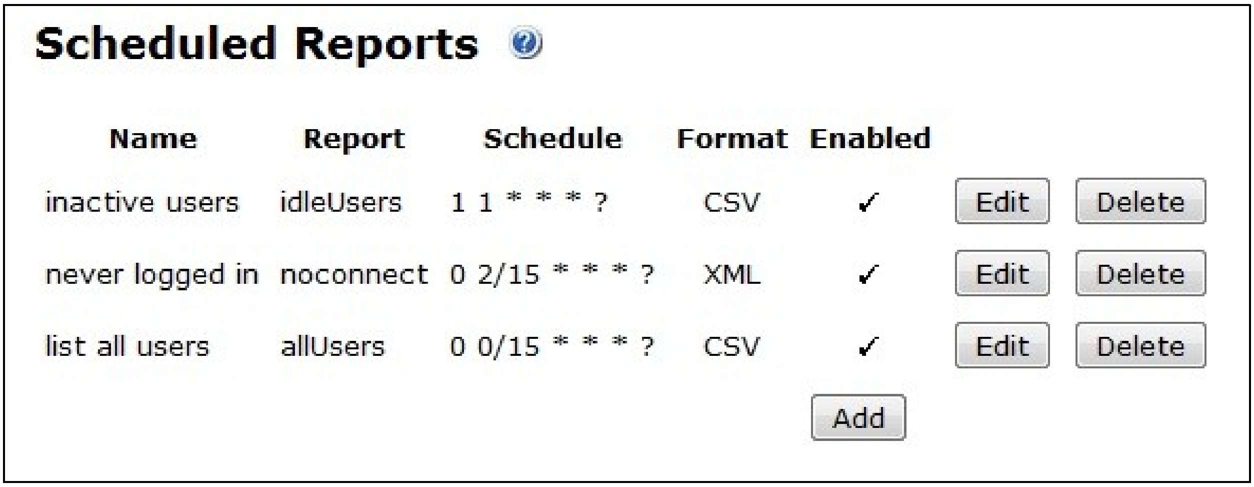
1 rows found

#	Username	Last Login
1	test	

Clicking on *Export as XML* or *Export as CSV* allows the report to be saved from the browser.

9 Producing Scheduled Reports

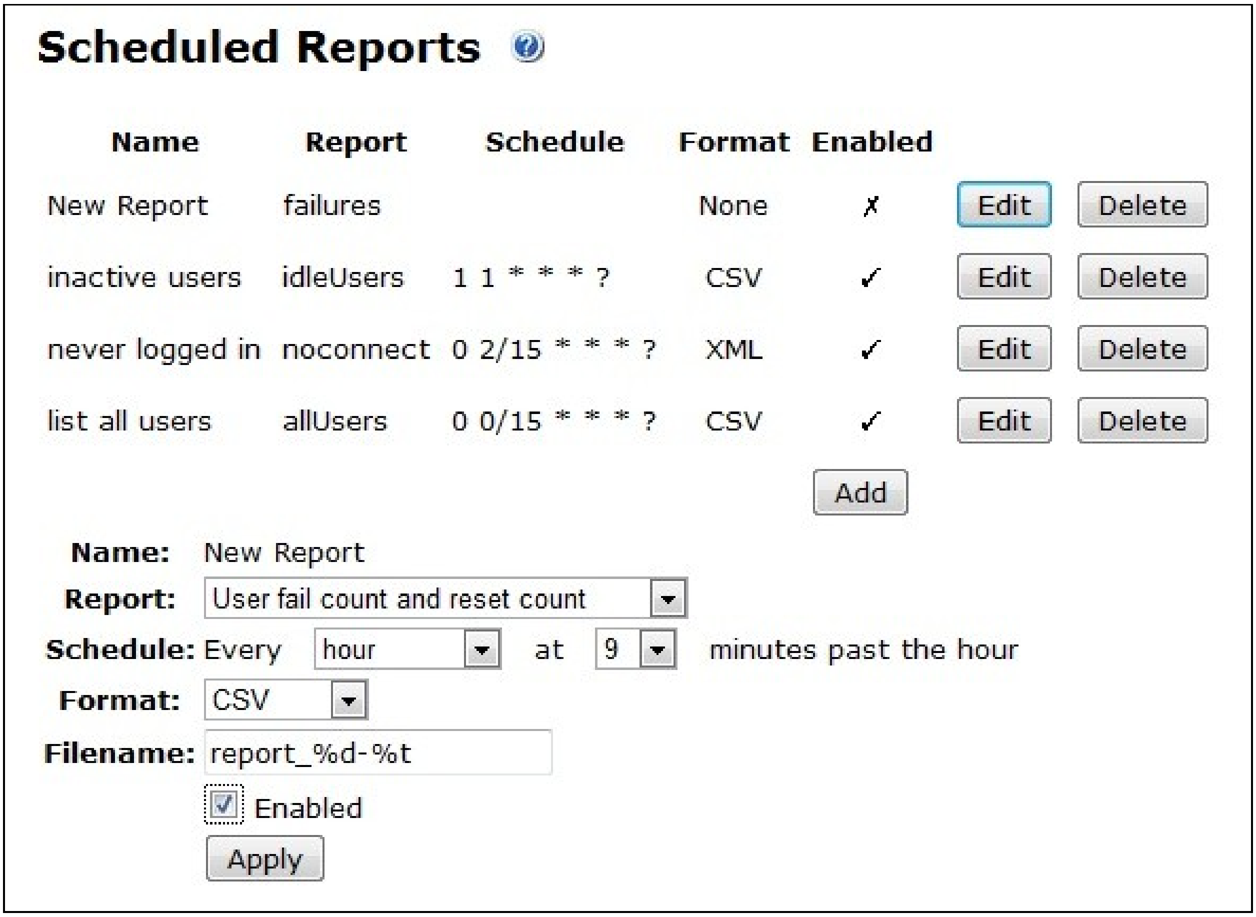
From the Swivel Administration console select Reporting then Scheduled. Existing scheduled reports are listed. For information on creating custom schedules see Schedule.



Scheduled Reports

Name	Report	Schedule	Format	Enabled		
inactive users	idleUsers	1 1 * * * ?	CSV	✓	Edit	Delete
never logged in	noconnect	0 2/15 * * * ?	XML	✓	Edit	Delete
list all users	allUsers	0 0/15 * * * ?	CSV	✓	Edit	Delete

To create a new scheduled report click on *Add*



Scheduled Reports

Name	Report	Schedule	Format	Enabled		
New Report	failures		None	✗	Edit	Delete
inactive users	idleUsers	1 1 * * * ?	CSV	✓	Edit	Delete
never logged in	noconnect	0 2/15 * * * ?	XML	✓	Edit	Delete
list all users	allUsers	0 0/15 * * * ?	CSV	✓	Edit	Delete

Name: New Report

Report: User fail count and reset count

Schedule: Every hour at 9 minutes past the hour

Format: CSV

Filename: report_%d-%t

☑ Enabled

Enter the following information:

Name: Name for the report

Report: Select the required report

Schedule: How often the report is run

Format: The output format for the report, options are XML or CSV

Filename: The filename given to the report, for multiple reports to be run at the same time this should be unique, example *inactive_users_%d-%t.txt* creates a file called inative_users_date-time.txt

Email Ticking this box enables the reporting by email. This function is available from Swivel 3.10 onwards. The email address is set under Policy/Reporting

Enabled Ticking this box enables the report to be run.

Apply the settings to save them.

9.1 Scheduled report file locations

Scheduled reports are stored within the Swivel installation, and varies depending upon the install type and version.

Swivel 3.9.1 onwards stores the reports in <swivel home>/.swivel/reporting for Swivel appliances and software installations.

Swivel 3.9 stores the scheduled reports data in <path to pinsafe>/WEB-INF/reports

For an appliance <path to pinsafe> is: **/usr/local/tomcat/webapps/pinsafe**

For a Windows 3.9.1 installation, <swivel home> will depend on the user account under which Tomcat is running, and will typically be **C:\Users\Account**. You can check the actual directory from the PINsafe Administration Console Status page. Look for **Data Storage Root**.

For a Swivel 3.9 software install under Windows, <path to pinsafe> is usually: **C:\Program Files\Apache Software Installation\Tomcat 5.5\webapps\pinsafe**. Obviously, if you are using a different version of Tomcat, adjust accordingly.

10 Report date Format

The report date format can be set on the Swivel Administration console under Server/Language/Date Format

11 Available Reports

List all users

List all users with no activity since a given date

List all users with no activity within a given number of days

List login failures and self-resets since the last successful login

List creation time and latest login for all users

List users that have never logged in

List number of users hourly logged

12 Custom Reports

Always backup files before editing.

The report definition file is under the following locations:

Appliances after 3.9: /home/swivel/.swivel/conf/reports.xml

Appliances up to 3.9: /usr/local/tomcat/webapps/pinsafe/WEB-INF/conf/reports.xml

Software install after 3.9 <swivel home>/.swivel/conf/reports.xml

Software install up to 3.9 <path to Tomcat>/webapps/pinsafe/WEB-INF/conf/reports.xml

See above for details on finding the location of <swivel home>.

The default contents of this file at the time of writing is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<reports>
       <report name="">
               <title>---- Please select a report to run ----</title>
       <description>No report selected</description>
       </report>
       <report name="allUsers">
       <title>List all users</title>
               <description>Lists all usernames in the PINsafe database</description>
               <headers>
                       <header>Username</header>
               </headers>
               <fields>H</fields>
               <tables>PINSAFEJ</tables>
       </report>
       <report name="idleUsers">
               <title>List idle users</title>
       <description>Lists users that have not logged in since a specified date</description>
               <headers>
                       <header>Username</header>
                       <header>Last Login</header>
               </headers>
               <fields>U.H, A.D</fields>
               <tables>PINSAFEJ U LEFT OUTER JOIN PINSAFEN A ON U.G = A.A AND A.C = 0</tables>
               <query>A.D IS NULL OR A.D < ?</query>
               <params>
                       <param name="idledate" type="Date" label="Idle since" />
               </params>
       </report>
       <report name="failures">
          <title>User fail count and reset count</title>
      <description>Lists the number of consecutive login failures and self-resets since the last successful login for all users</description>
          <headers>
              <header>Username</header>
              <header>Fail Count</header>
              <header>Reset Count</header>
          </headers>
          <fields>H, B, F</fields>
          <tables>PINSAFEJ</tables>
          <query />
       </report>
       <report name="connection">
       <title>Latest connection for all users</title>
          <description>Lists the creation time and last login time for all users</description>
          <headers>
              <header>Username</header>
              <header>Created</header>
              <header>Last Login</header>
          </headers>
          <fields>U.H, A1.D, A2.D</fields>
          <tables>PINSAFEJ U INNER JOIN PINSAFEN A1 ON U.G=A1.A AND A1.C=3 LEFT OUTER JOIN PINSAFEN A2 ON U.G=A2.A AND A2.C=0</tables>
          <query />
       </report>
       <report name="noconnect">
       <title>Users that have never logged in</title>
          <description>Lists all users that have never successfully logged into PINsafe</description>
          <headers><header>Username</header></headers>
          <fields>H</fields>
          <tables>PINSAFEJ</tables>
          <query>G NOT IN (SELECT DISTINCT A FROM PINSAFEN WHERE C=0)</query>
       </report>
 </reports>
```

12.1 Adding new reports

Currently, requests for new reports can be made via your reseller to PINsafe support supportdesk@swivelsecure.com.

Once you've obtained a new report, the easiest way to add it to the configuration file above, is to use WinSCP. See the WinSCP How To Guide for further information on using WinSCP.

To add custom reports to the built-in reporting feature of PINsafe, you would append this to the reports.xml file. Insert new reports immediately before the last line:

```
 </reports>
```

Once you have installed the report, go to Reporting -> Instant, and the new report should be available from the drop-down list.

You don't need to stop and restart Tomcat for the report to become available: the list is reloaded every time the report page is displayed.

We do not recommend writing your own reports, but for those who are competent with SQL and recognise that the reports are essentially XML-encoded fragments of SQL statements, you can get the full database schema for the Swivel database from here.

NOTE: Any reports that reference the policy flags table, PINSAFEC, will not work with Sentry version 4.2 or later, and must reference the new status flags table, PINSAFES. Custom reports listed below that need changing are noted as such, and two alternatives are given.

12.1.1 Creation time, last login and number of logins

This report lists all successful logins within the last 30 days (the audit trail is only kept for 30 days by default, but can be set to a different value.

```
  <report name="loginCount">
  <title>Login count and latest login for all users</title>
  <description>Lists the creation time, last login time and number of logins for all users</description>
  <headers>
  <header>Username</header>
  <header>Created</header>
  <header>Last Login</header>
  <header>Login Count</header>
  </headers>
  <fields>U.H, MAX(A1.D), MAX(A2.D), COUNT(AU.G)</fields>
  <tables>PINSAFEJ U INNER JOIN PINSAFEN A1 ON U.G=A1.A AND A1.C=3 LEFT OUTER
  JOIN PINSAFEN A2 ON U.G=A2.A AND A2.C=0 INNER JOIN PINSAFEM AU ON U.G=AU.G AND AU.A=0
  GROUP BY U.G, U.H</tables>
  <query />
  </report>
```

If you want to include users that have not logged in within the last 30 days, change "INNER JOIN PINSAFEM" to "LEFT OUTER JOIN PINSAFEM".

12.1.2 List all users, together with their login count

This report lists the following information:

First Name

Last Name

Username

Phone Number

Creation Date

Last Login Date

Login Count

Note: creation date if within length of audit time

This report has been tested on both Internal and MySQL databases.

```
        <report name="user-login-count">
                <title>User details with login count</title>
                <description>Lists all users, together with their login count over the last 30 days, and last login time</description>
                <headers>
                        <header>Username</header>
                        <header>First name</header>
                        <header>Last name</header>
                        <header>Phone</header>
                        <header>Created</header>
                        <header>Latest login</header>
                        <header>Login count</header>
                </headers>
                <fields>U.H, ATT1.C, ATT2.C, ATT3.C, ACT1.D, ACT2.D, AUD.CT</fields>
                <tables><![CDATA[
                PINSAFEJ U
                LEFT OUTER JOIN PINSAFEP ATT1 ON U.G=ATT1.A AND ATT1.B='givenname'
                LEFT OUTER JOIN PINSAFEP ATT2 ON U.G=ATT2.A AND ATT2.B='familyname'
                LEFT OUTER JOIN PINSAFEP ATT3 ON U.G=ATT3.A AND ATT3.B='phone'
                LEFT OUTER JOIN PINSAFEN ACT1 ON U.G=ACT1.A AND ACT1.C=3
                LEFT OUTER JOIN PINSAFEN ACT2 ON U.G=ACT2.A AND ACT2.C=0
                LEFT OUTER JOIN (SELECT G, COUNT(*) CT FROM PINSAFEM WHERE A=0 GROUP BY G) AUD ON U.G=AUD.G
                ]]></tables>
        </report>
```

12.1.3 List users of a particular group showing their last login over the last month

```
 <report name="connectbygroup">
                <title>Authentication by User in a Group</title>
                <description>List of users in a particular group, listing their logins over the last month</description>
                <headers>
                        <header>Group</header>
                        <header>Username</header>
                        <header>Login Date</header>
                </headers>
                <fields>G.A, U.H, A.E</fields>
                <tables>PINSAFEM A INNER JOIN PINSAFEJ U ON A.G=U.G INNER JOIN PINSAFEI G ON U.G=G.B</tables>
                <query>G.A=? ORDER BY U.C</query>
                <params>
                        <param name="group" type="String" label="Group Name" />
                </params>
        </report>
```

You will be prompted for the group *name:*, type in the actual name of the group, remember it's the Swivel group name, not the Active Directory FQDN.

12.1.4 List Inactive Users Within a Period By Group

Note that for it to work properly, set the Policy -> General, Audit Log length to a suitably large value (example 365 days)

```
 <report name="idleUsersByGroup">
 <title>List Inactive Users Within a Period By Group</title>
 <description>List members of a given group who did not log in within a specified time</description>
 <headers>
        <header>Username</header>
        <header>Last Login</header>
 </headers>
 <fields>U.H, A.D</fields>
 <tables>PINSAFEJ U LEFT OUTER JOIN PINSAFEN A ON A.A = U.G AND A.C = 0 JOIN PINSAFEI G ON G.B = U.G AND G.A = ?</tables>
 <query>U.G NOT IN (SELECT G FROM PINSAFEM WHERE A=0 AND E &gt;= ? AND E &lt;= ?) </query>
 <params>
   <param name="group" type="String" label="Group" />
        <param name="startdate" type="Date" label="Start Date" />
        <param name="enddate" type="Date" label="End Date" />
 </params>
 </report>
```

12.1.5 Total number of users

```
<report name="numberOfUsers">
  <title>Total number of users</title>
  <description>Report the total number of users in the database</description>
  <headers>
    <header>Num. Users</header>
  </headers>
  <tables>PINSAFEJ</tables>
  <fields>COUNT(*)</fields>
</report>
```

12.1.6 Total number of users in each group

```
<report name="groupMemberCount">
  <title>Count of users in each group by repository</title>
  <description>List the number of users in each group within each repository</description>
  <headers>
    <header>Repository</header>
    <header>Group</header>
    <header>Num. Users</header>
  </headers>
  <fields>R.B, G.A, COUNT(U.G)</fields>
  <tables>PINSAFEJ U JOIN PINSAFEL R ON U.I=R.A JOIN PINSAFEI G ON U.G=G.B GROUP BY R.B, G.A ORDER BY R.B, G.A</tables>
</report>
```

12.1.7 List users in each group

```
<report name="groupMembers">
  <title>List of users in each group by repository</title>
  <description>List the users in each group by repository</description>
  <headers>
    <header>Repository</header>
    <header>Group</header>
    <header>Users</header>
  </headers>
  <fields>R.B, G.A, U.C</fields>
  <tables>PINSAFEJ U JOIN PINSAFEL R ON U.I=R.A JOIN PINSAFEI G ON U.G=G.B ORDER BY R.B, G.A, U.C</tables>
</report>
```

12.1.8 List all OATH users

```
<report name="TokenUsers">
  <title>List of users allocated OATH tokens</title>
  <description>Lists all users that have been allocated OATH tokens</description>
  <headers>
    <header>User</header>
  </headers>
  <fields>U.H</fields>
  <tables>PINSAFEJ U JOIN PINSAFEQ T ON U.G=T.C</tables>
</report>
```

12.1.9 List users with Mobile permissions

```
<report name="MobileUsers">
  <title>List of users entitle to use mobile client</title>
  <description>Lists users that have the "Mobile" right</description>
  <headers><header>User</header></headers>
  <fields>U.H</fields>
  <tables>PINSAFEJ U JOIN PINSAFEB R ON U.G = R.B</tables>
  <query>R.A = 2</query>
</report>
```

12.1.9.1 List users with a provisioned Mobile device

```
<report name="AllocatedMobileUsers">
  <title>List of users with provisioned mobile clients</title>
  <description>Lists users that have provisioned their mobile clients</description>
  <headers><header>User</header></headers>
  <fields>U.H</fields>
  <tables>PINSAFEJ U JOIN PINSAFEO M ON U.G = M.C</tables>
</report>
```

12.1.9.2 List users with Mobile client rights that have not yet provisioned a device

```
<report name="UnprovisionedMobileUsers">
  <title>List of users that have not yet provisioned a mobile device</title>
  <description>Lists users that have the right to use a mobile client, but have not yet provisioned one</description>
  <headers><header>User</header></headers>
  <fields>U.H</fields>
  <tables>PINSAFEJ U JOIN PINSAFEB R ON U.G = R.B AND R.A=2</tables>
  <query>U.G NOT IN (SELECT C FROM PINSAFEO)</query>
</report>
```

12.1.9.3 List the number of login failures for all users

```
<report name="loginFailCount">
        <title>Count of failed logins for all users</title>
        <description>Lists the number of failed logins for each user, to the extent of the audit records</description>
        <headers>
                <header>Username</header>
                <header>Fail Count</header>
        </headers>
        <fields>I, Count(A)</fields>
        <tables>PINSAFEM</tables>
        <query>A=14 group by I order by I</query>
</report>
```

12.1.9.4 List of users that provisioned before a given date

```
<report name="oldProvisions">
        <title>List of users that provisioned before a given date</title>
        <description>Lists users that last provisioned their mobile app earlier than a specified date.</description>
        <headers>
                <header>Username</header>
                <header>Provision date</header>
        </headers>
        <params>
                <param name="cutoffdate" type="Date" label="Provisioned before"/>
        </params>
        <fields>U.H, A.D</fields>
        <tables>PINSAFEJ U JOIN PINSAFEO M on U.G = M.C JOIN PINSAFEN A on U.G = A.A and A.C=15</tables>
        <query>A.D &lt; ?</query>
</report>
```

12.1.9.5 List unprovisioned Mobile Users for a given date

```
<report name="Unprovisionedbydate" supporteddbs="mariadb,mysql,mssql">
        <title>List Unprovisioned Mobile Users older than given date</title>
        <description>Lists users, together with the latest date that they Provisioned</description>
        <headers>
                <header>Username</header>
                <header>Last Changed</header>
        </headers>
        <fields>U.H, U.C name, MAX(A.D) lastdate</fields>
        <tables>PINSAFEJ U JOIN PINSAFEB R ON U.G = R.B AND R.A=2; PINSAFEJ U JOIN PINSAFEN A on U.G=A.A and A.C in (1,2,3,6) group by A.A ha
        <query>U.G NOT IN (SELECT C FROM PINSAFEO)</query>
        <params>
                <param name="changedate" type="Date" label="Latest date" />
        </params>
</report>
```

12.1.10 List the number of login failures since a given date, for all users

```
<report name="loginFailCountSinceDate">
        <title>Count of failed logins since a given date for all users</title>
        <description>Lists the number of failed logins for each user, since a given date</description>
        <headers>
                <header>Username</header>
                <header>Fail Count</header>
        </headers>
        <fields>I, Count(A)</fields>
        <tables>PINSAFEM</tables>
        <query>A=14 and E>? group by I order by I</query>
        <params>
                <param name="sinceDate" type="Date" label="Cutoff Date" />
        </params>
</report>
```

12.1.11 List the last PIN change for all users

NOTE: this report has two different versions, as the Internal database doesn't support the features used by the first version. You may wish to include only the version that matches your database.

```
<report name="lastchanged" supporteddbs="mysql,mariadb,mssql">
        <title>List of PIN changes by date</title>
        <description>Lists all users, together with the latest date that their PIN was changed, or the creation date if never changed, starti
        <headers>
                <header>Username</header>
                <header>Last Changed</header>
        </headers>
        <fields>U.C name, MAX(A.D) lastdate</fields>
        <tables>PINSAFEJ U JOIN PINSAFEN A on U.G=A.A and A.C in (1,2,3,6) group by A.A order by lastdate</tables>
</report>

<report name="lastchanged" supporteddbs="internal">
        <title>List of PIN changes by date</title>
        <description>Lists all users, together with the latest date that their PIN was changed, or the creation date if never changed, starti
        <headers>
                <header>Username</header>
                <header>Last Changed</header>
        </headers>
        <fields>U.C name, Q.DD lastdate</fields>
        <tables>PINSAFEJ U JOIN (SELECT A, Max(D) DD FROM PINSAFEN where C in (1,2,3,6) group by A) Q on U.G = Q.A order by lastdate</tables>
</report>
```

12.1.12 List PIN changes older than a given date

NOTE: as above, there are different versions for different databases.

```
<report name="lastchanged2" supporteddbs="mariadb,mysql,mssql">
        <title>List of PIN changes older than given date</title>
        <description>Lists users, together with the latest date that their PIN was changed, or the creation date if never changed, starting w
        <headers>
                <header>Username</header>
                <header>Last Changed</header>
        </headers>
        <fields>U.C name, MAX(A.D) lastdate</fields>
        <tables>PINSAFEJ U JOIN PINSAFEN A on U.G=A.A and A.C in (1,2,3,6) group by A.A having date &lt; ? order by lastdate</tables>
        <params>
                <param name="changedate" type="Date" label="Latest date" />
        </params>
</report>

<report name="lastchanged2" supporteddbs="internal">
        <title>List of PIN changes older than given date</title>
        <description>Lists users, together with the latest date that their PIN was changed, or the creation date if never changed, starting w
        <headers>
                <header>Username</header>
                <header>Last Changed</header>
        </headers>
        <fields>U.C name, Q.DD lastdate</fields>
        <tables>PINSAFEJ U JOIN (SELECT A, Max(D) DD FROM PINSAFEN where C in (1,2,3,6) group by A) Q on U.G = Q.A</tables>
        <query>Q.DD &lt; ? order by lastdate</query>
        <params>
                <param name="changedate" type="Date" label="Latest date" />
        </params>
</report>
```

12.1.13 Lists the number of logins for each hour

NOTE: For mariadb and mysql only

```
        <report name="loginsByHour">
        <title>List of logins by hour</title>
        <description>Lists the number of logins for each hour</description>
        <headers>
                <header>Date/Time</header>
        <header>Count</header>
        </headers>
        <fields>HOUR(E), COUNT(*)</fields>
        <tables>PINSAFEM</tables>
        <query>A=0 GROUP BY HOUR(E)</query>
        </report>
```

12.1.14 List number of logins for each day and hour for the last 7 days

```
        <report name="loginsByDayHour">
        <title>List of logins by Day and Hour</title>
        <description>Lists the number of logins for each day and hour for the last 7 days</description>
        <headers>
                <header>Date</header>
                <header>Hour</header>
                <header>Count</header>
        </headers>
        <fields>DATE(E), HOUR(E), COUNT(*)</fields>
        <tables>PINSAFEM</tables>
        <query>A=0 AND E&lt;CURDATE() AND E&gt;DATE_SUB(CURDATE(), INTERVAL 7 DAY) GROUP BY DATE(E), HOUR(E)</query>
        </report>
```

12.1.15 Enhancement to the previous report to include login failures and restrict to a single group

```
        <report name="loginsByDayHourPerGroup">
        <title>List of logins by Day and Hour for a group</title>
        <description>Lists the number of logins or failures for each day and hour for the last 7 days</description>
        <headers>
                <header>Date</header>
                <header>Hour</header>
                <header>Count</header>
        </headers>
        <fields>DATE(E), HOUR(E), COUNT(*)</fields>
        <tables>PINSAFEM</tables>
        <query>(A=0 OR A=14) AND E&lt;CURDATE() AND E&gt;DATE_SUB(CURDATE(), INTERVAL 7 DAY) AND G IN (SELECT B FROM PINSAFEI WHERE A=?) GROU
        <params>
                <param name="group" type="String" label="Group" />
        </params>
        </report>
```

12.1.16 List users that have been recently deleted

```
 <report name="deletedUsers">
   <title>List of users that have been deleted recently</title>
   <description>Lists usernames that have been permanently deleted, but who still appear in the audit table</description>
   <headers><header>Username</header></headers>
   <fields>DISTINCT(I)</fields>
   <tables>PINSAFEM</tables>
   <query>I NOT IN (SELECT C FROM PINSAFEJ)</query>
 </report>
```

12.2 Internal database reports

Edit reports.xml and search for the following line:

```
<report name="idleUsers2" supporteddbs="mysql,mssql,oracle">
```

If this line cannot be found, or there is no *supporteddbs* attribute, then it may be an older version, that does not suppport this.

Modify the above line as follows to add internal to the list of supported databases:

```
<report name="idleUsers2" supporteddbs="mysql,mssql,oracle,internal">
```

12.2.1 Internal database, lists users that have not logged in since a specified date

Add a query as follows, Insert this after the other <query ... lines and before <params>.

```
<query db="internal">A.D IS NULL OR {fn TIMESTAMPDIFF(SQL_TSI_DAY, CURRENT_DATE, A.D)} > ? </query>
```

Example

```
<query>A.D IS NULL OR A.D < ?</query>
<query db="internal">A.D IS NULL OR {fn TIMESTAMPDIFF(SQL_TSI_DAY, CURRENT_DATE, A.D)} > ? </query>
<params>
  <param name="idledate" type="Date" label="Idle since" />
</params>
```

12.3 Custom reports for multiple database types

The following reports will work with differing database types as the database is specified within the report as db="mysql db="mssql" db="oracle" db="internal"

12.3.1 Users marked as deleted

This report needs to be modified to work with Sentry version 4.2 onwards. Two different versions are given below:

Version 4.2 onwards:

```
        <report name="deleted-users">
                <title>List of users marked as deleted</title>
                <description>List all users who are currently marked as deleted.</description>
                <headers><header>Username</header></headers>
                <fields>U.H</fields>
                <tables>PINSAFEJ U INNER JOIN PINSAFES F ON U.G=F.A</tables>
                <query>F.D &amp; 1 = 1</query>
        </report>
        <report name="deleted-users-by-date">
                <title>List of users marked as deleted recently</title>
                <description>List all users who have been marked as deleted in the last #days.</description>
                <headers>
                        <header>Username</header>
                        <header>Date</header>
                </headers>
                <fields>U.H, A.D</fields>
                <tables>PINSAFEJ U INNER JOIN PINSAFEC F ON U.G=F.C AND F.B=4 AND F.D=1 INNER JOIN PINSAFEN A ON U.G=A.A AND A.C=10</tables>
                <query db="mysql">F.D &amp; 1 = 1 AND A.D > DATE_SUB(NOW(), INTERVAL ? DAY)</query>
                <query db="mssql">F.D &amp; 1 = 1 AND A.D > DATEADD(day, -?, GETDATE())</query>
                <query db="oracle">F.D &amp; 1 = 1 AND A.D > ADD_DAYS(SYSDATE, -?)</query>
                <query db="internal">F.D &amp; 1 = 1 AND {fn TIMESTAMPDIFF(SQL_TSI_DAY, A.D, CURRENT_DATE)} &lt; ? </query>
                <params>
                        <param name="days" type="Integer" label="Cutoff days" />
                </params>
        </report>
```

Up to version 4.1.3:

```
        <report name="deleted-users">
                <title>List of users marked as deleted</title>
                <description>List all users who are currently marked as deleted.</description>
                <headers><header>Username</header></headers>
                <fields>U.H</fields>
                <tables>PINSAFEJ U INNER JOIN PINSAFEC F ON U.G=F.C AND F.B=4 AND F.D=1</tables>
        </report>
        <report name="deleted-users-by-date">
                <title>List of users marked as deleted recently</title>
                <description>List all users who have been marked as deleted in the last #days.</description>
                <headers>
                        <header>Username</header>
                        <header>Date</header>
                </headers>
                <fields>U.H, A.D</fields>
                <tables>PINSAFEJ U INNER JOIN PINSAFEC F ON U.G=F.C AND F.B=4 AND F.D=1 INNER JOIN PINSAFEN A ON U.G=A.A AND A.C=10</tables>
                <query db="mysql">A.D > DATE_SUB(NOW(), INTERVAL ? DAY)</query>
                <query db="mssql">A.D > DATEADD(day, -?, GETDATE())</query>
                <query db="oracle">A.D > ADD_DAYS(SYSDATE, -?)</query>
                <query db="internal">{fn TIMESTAMPDIFF(SQL_TSI_DAY, A.D, CURRENT_DATE)} &lt; ? </query>
                <params>
                        <param name="days" type="Integer" label="Cutoff days" />
                </params>
        </report>
```

12.3.2 Users whose PIN never expires

Version 4.2 onwards:

```
         <report name="pin-never-expires">
                <title>List of users whose PIN never expires</title>
                <description>Lists the names of users that have the "PIN never expires" flag set.</description>
                <headers><header>Username</header></headers>
                <fields>U.H</fields>
                <tables>PINSAFEJ U INNER JOIN PINSAFES F ON U.G=F.A</tables>
                <query>F.B=1</query>
        </report>
```

The above report can be modified to show users who are marked as disabled, locked, deleted, inactive or require a PIN change, by changing the query as follows:

- User must change PIN after next login: *<query>F.C=1</query>*

- User disabled: *<query>F.D &amp; 2 = 2</query>*
- User locked (any reason): *<query>F.D &amp; 124 <> 0</query>*
- User is marked as deleted: *<query>F.D &amp; 1 = 1</query>*

Up to version 4.1.3

```
<report name="pin-never-expires">
        <title>List of users whose PIN never expires</title>
        <description>Lists the names of users that have the "PIN never expires" flag set.</description>
        <headers><header>Username</header></headers>
        <fields>U.H</fields>
        <tables>PINSAFEJ U INNER JOIN PINSAFEC F ON U.G=F.C</tables>
        <query>F.B=3 AND F.D=1</query>
</report>
```

The above report can be modified to show users who are marked as disabled, locked, deleted, inactive or require a PIN change, by changing the value of F.B in the query as follows:

- 0 - User disabled
- 1 - User locked
- 2 - User must change PIN after next login
- 3 - User's PIN never expires
- 4 - User is marked as deleted
- 5 - User is inactive

12.3.3 All User Logins, including login location

NOTE: this will only report logins as far back as audit records are retained (by default, 30 days). Also, the location field is not always reported. Most AgentXML integrations (e.g. OWA) do not report the user's location. RADIUS-based integrations will report the calling workstation ID if the NAS reports it, or else the NAS IP address.

```
<report name="connections">
  <title>All recent connections for all users</title>
  <description>Lists the login times and location (where available) for all users, within the audit retention time</description>
  <headers>
    <header>Username</header>
    <header>Created</header>
    <header>Last Login</header>
    <header>Location</header>
  </headers>
  <fields>U.H, A1.D, AU.E, AU.B</fields>
  <tables>PINSAFEJ U INNER JOIN PINSAFEN A1 ON U.G=A1.A AND A1.C=3 LEFT OUTER JOIN PINSAFEM AU ON U.G=AU.G AND AU.C=0</tables>
  <query>1 ORDER BY U.H, AU.E DESC</query>
</report>
```

12.3.4 All user logins and failures for a given group

This reports user logins and failures for a given period from the current date and a given group.

NOTE: this will only report logins as far back as audit records are retained (by default, 30 days). Also, the location field is not always reported. Most AgentXML integrations (e.g. OWA) do not report the user's location. RADIUS-based integrations will report the calling workstation ID if the NAS reports it, or else the NAS IP address. Details also may not have been populated.

```
<report name="LoginDetails">
  <title>Details of User logins</title>
  <description>List of all logins and login failures within a given time for a given group</description>
  <headers>
    <header>Username</header>
    <header>Activity</header>
    <header>Date/Time</header>
    <header>Location</header>
    <header>Details</header>
  </headers>
  <fields>I, if(A=0, 'login', 'failed'), E, B, C</fields>
  <tables>PINSAFEM</tables>
  <query db="mariadb,mysql">(A=0 OR A=14) AND DATEDIFF(CURRENT_DATE(), E) &lt; ? AND G IN (SELECT B FROM pinsafei WHERE A=?) ORDER BY I, E d
  <params>
    <param name="days" label="Number of days" type="Integer" />
    <param name="group" label="Group" type="String" />
  </params>
</report>
```

12.3.5 All login failures in the last few hours

This report gives a summary of the number of login failures for each user within a given time from the current time. The time difference is given in hours.

This report will only go back as far as audit records are retained (by default 30 days), so specifying a very large number of hours will not give complete results.

```
<report name="RecentLoginFailCount">
  <title>Count of failed logins in the last x hours</title>
  <description>Lists the number of failed logins for each user in the last few hours</description>
  <headers>
    <header>Username</header>
    <header>Fail Count</header>
  </headers>
  <fields>I, Count(A)</fields>
  <tables>PINSAFEM</tables>
  <query db="mariadb,mysql">A=14 and E>DATE_SUB(CURRENT_TIMESTAMP(), INTERVAL ? HOUR) group by I order by I</query>
  <params>
    <param name="cutoffHours" type="Integer" label="# hours"/>
  </params>
</report>
```

12.4 Other reports

The following reports were tested and made available regarding specific needs:

12.4.1 Idle Users for the last 180 days but created over 30 days

```
<report name="idleusers180createdover30">
  <title>Users who never logged in over 180 days but created over 30</title>
    <description>List the number of users who have never logged in over the last 180 days but created more than 30 days ago</descripti
  <headers>
  <header>Username</header>
  </headers>
  <fields>U.H</fields>
  <tables>PINSAFEJ U JOIN PINSAFEN A1 ON U.G = A1.A AND A1.C = 3 LEFT OUTER JOIN PINSAFEN A2 ON U.G = A2.A AND A2.C=0</tables>
  <query><![CDATA[(A1.D &lt; DATE_SUB(NOW(), INTERVAL 30 DAY)) AND (A2.D IS NULL OR A2.D &lt; DATE_SUB(NOW(), INTERVAL 180 DAY))]]></q
</report>
```

12.4.2 List inactive users, locked, disabled, deleted and show creation date, login and last PIN change date

Version 4.2 onwards:

```
 <report name="inactivelockeddisabledUsers">
 <title>Users inactive, locked, disabled, deleted and show creation, login, PIN change date</title>
<description>List all Users inactive, locked, disabled, deleted and show creation, login, PIN change date</description>
<headers>
 <header>UID</header>
         <header>REPOS</header>
         <header>DISABLED</header>
         <header>LOCKED</header>
         <header>DELETED</header>
         <header>INACTIVE</header>
         <header>CREATED</header>
         <header>LOGGEDIN</header>
         <header>PINCHANGE</header>
</headers>
<fields><![CDATA[U.H, R.B, S.D & 2 = 2, S.D & 116 <> 0, S.D & 1 = 1, S.D & 8 = 8, A1.D, A2.D, A3.D]]></fields>
<tables><![CDATA[PINSAFEJ U JOIN PINSAFEL R ON U.I=R.A JOIN PINSAFES S ON U.G=S.A LEFT OUTER JOIN PINSAFEN A1 ON U.G=A1.A AND A1.C=3 L
 </report>
```

Up to version 4.1.3:

```
<report name="inactivelockeddisabledUsers">
   <title>Users inactive, locked, disabled, deleted and show creation, login, PIN change date</title>
   <description>List all Users inactive, locked, disabled, deleted and show creation, login, PIN change date</description>
  <headers>
    <header>UID</header>
    <header>REPOS</header>
    <header>DISABLED</header>
    <header>LOCKED</header>
    <header>DELETED</header>
    <header>INACTIVE</header>
    <header>CREATED</header>
    <header>LOGGEDIN</header>
    <header>PINCHANGE</header>
 </headers>
  <fields>U.H, R.B, S1.D, S2.D, S3.D, S4.D, A1.D, A2.D, A3.D</fields>
  <tables><![CDATA[PINSAFEJ U JOIN PINSAFEL R ON U.I=R.A LEFT OUTER JOIN PINSAFEC S1 ON U.G=S1.C AND S1.B=0 LEFT OUTER JOIN PINSAFEC S
</report>
```

12.4.3 List Users IDs, email, creation date, last login, last pin change and repository name

```
<report name="allUsersdatalastloginandrepo">
       <title>Users IDs, email, date, last login, last pin change and repo name</title>
      <description>Lists all the Users Ids, email address, created date, last login, last pin change and repository name</description>
  <headers>
   <header>Username</header>
   <header>ReposName</header>
   <header>Email</header>
   <header>CreateDate</header>
   <header>LoginDate</header>
   <header>PINChange</header>
  </headers>
  <fields>U.H, U.E, T.A, A1.D, A2.D, A3.D</fields>
  <tables><![CDATA[PINSAFEJ U LEFT OUTER JOIN PINSAFEP T ON U.G = T.A AND T.B='email' LEFT OUTER JOIN PINSAFEN A1 ON U.G = A1.A AND A1
 </report>
```

12.5 Writing Your Own Reports

This section is intended only for customers who are familiar with writing SQL queries, and with XML.

12.5.1 Database Schema

While writing your report, refer to the Database Schema page for reference on the database table and column names used by Swivel.

12.5.2 Report Definition Format

A report definition is an xml element, which includes elements of a SQL query, plus display information. The format is:

```
<report name="reportName">
 <title>Report Title</title>
 <description>Report description</description>
 <headers>
   <header>header1</header>
   ...
 </headers>
 <fields>SQL-fields</fields>
 <tables>SQL-tables-and-joins</tables>
 <query>SQL-query</query>
 <params>
   <param name="param-name" label="param-label" type="String|Integer|Date" />
   ...
 </params>
```

```
</report>
```

The report name is used internally, but must be unique.

<title>, <description> and <headers> are display elements: the title is displayed in the drop-down and at the top of the report, while the description is shown when the report is selected. Headers are individual column headers, and the number of headers should match the number of columns in the field list.

The SQL query is built as

```
SELECT SQL-fields FROM SQL-tables-and-joins WHERE SQL-query
```

The <query> element is optional: if missing, no WHERE clause is added. There are no elements for ORDER BY or GROUP BY, but you can add "ORDER BY" or "GROUP BY" to the query element, or to the tables element if there is no query.

The <params> elements are used to set values for replaceable parameters within the query. <param> elements are applied in the order listed. The name attribute is used to identify the parameter internally. The label attribute is displayed on-screen when requesting parameter values. The type attribute must be "String", "Integer" or "Date" - these values are case-sensitive.

13 Troubleshooting

Versions 3.8 to 3.9.3 may produce a HTTP Status 500 error when selecting a new report:

```
java.lang.IllegalArgumentException: fromIndex(50) > toIndex(0)

java.util.SubList.<init>(Unknown Source)

java.util.AbstractrList.subList (Unknown Source)

com.swiveltechnologies.pinsafe.server.reporting.ReportResult.getRows (ReportResult.java:122)
```

This is caused by last page you were on when you used a report. For example if you were on page 4 of one particular report, but then attempt to access another report which does not contain 4 pages, it will attempt to access page 4 of that report instead of page 1 and fail with the error you sent.

So the solution is to upgrade, or as a workaround, make sure that you request page 1 of a report before leaving the reporting page. Closing down the web browser should also alleviate this.

14 Reporting Using Agent-XML How to Guide

14.1 How To Report using Agent-XML

14.2 Overview

The Internal Swivel database contains some information that is not available through the Swivel Administration Console. This document outlines some of the information that can be queried.

The **Audit Log** is where Swivel maintains an activity log for users. This is set on the Swivel Administration console under Policy General Audit Log length (days): default value is 30. The Audit log is present in Swivel versions 3.4 onwards, previous versions will return a value of 0 for the data.

For further information on Agent-XML see: Agent-XML and ReportingAPI. For information reporting through the Swivel Administration console see Reporting How to guide.

14.3 Prerequisites

Internal Swivel Database

Audit log requires Swivel 3.4 or higher

Swivel Agent configured to allow XML-Authentication.

Agent-XML requests are made against port 8080 on Swivel virtual or hardware appliances rather than through the proxy port

14.4 Performing Agent_XML Queries

The Internal database can be queried using a web browser connected from a system that is configured as an Agent. To configure an XML Agent, on the Swivel Administration console select Server then Agents, enter the following information:

Agents:

- Name: A descriptive name
- Hostname/IP: IP or hostname from which the XML queries will be made
- Shared secret: a value that needs to be the same on the Swivel server and the device from which XML queries will be made
- Group: default ANY A group of users permitted to make authentication requests
- Authentication Modes: default ALL whether single or dual channel authentication requests are permitted

14.5 Agent-XML Queries

Swivel virtual or hardware Appliance:

```
https://IP:8080/pinsafe/AdminXML?xml=<AdminRequest secret="secret" version="3.4"><Report repository="local"><query/></Report></AdminRequest>
```

Software Install:

```
http://IP:8080/pinsafe/AdminXML?xml=<AdminRequest secret="secret" version="3.4"><Report repository="local"><query/></Report></AdminRequest>
```

Report Format:

- Replace IP with the Swivel server name or IP address. Depending on your

configuration, you may also need to change the port number, for Swivel virtual or hardware appliances this is 8080.

- Replace **local** with the name of the Swivel repository you wish to query. You can only

query one repository at a time.

- The machine from which you make the query must be a Swivel
- Replace **secret** with the shared secret for that agent.
- Replace <query/> with the query to be specified, see below
- Replace **version** with the required version, usually 3.4 or 3.6

14.5.1 Show Idle user Accounts

<Idle since="dd-mmm-yyyy"/>

- Replace {dd-mmm-yyyy} with the date to check. This must be in the format (for example) 01-Jul-2009.

This query will show users who have never logged in, this report does not show users that have NEVER logged in:

Example:

```
http://127.0.0.1:8080/pinsafe/AdminXML?xml=<AdminRequest secret="secret" version="3.4"><Report repository="local"><Idle since="01-jan-2010"/>
```

Example output:

```
   <?xml version="1.0" ?>
 - <AdminResponse>
 - <Report repository="local">
 - <Idle>
   <User name="qwerty" />
   </Idle>
   </Report>
   </AdminResponse>
```

14.5.2 Show Disabled User Accounts

<Disabled/>

This query will show which user accounts have the status disabled.

Example:

```
http://127.0.0.1:8080/pinsafe/AdminXML?xml=<AdminRequest secret="secret" version="3.4"><Report repository="local"><Disabled/></Report></Admin
```

Example output:

```
   <?xml version="1.0" ?>
 - <AdminResponse>
 - <Report repository="local">
 - <Disabled>
   <User name="qwerty" />
   </Disabled>
   </Report>
   </AdminResponse>
```

14.5.3 Show Locked User Accounts

<Locked/>

This query will show which user accounts have the status disabled.

Example:

```
http://127.0.0.1:8080/pinsafe/AdminXML?xml=<AdminRequest secret="secret" version="3.4"><Report repository="local"><Locked/></Report></AdminRe
```

Example output:

```
   <?xml version="1.0" ?>
 - <AdminResponse>
 - <Report repository="local">
 - <Locked>
   <User name="qwerty" />
   </Locked>
   </Report>
   </AdminResponse>
```

14.6 Agent-XML Errors

AgentXML request failed, error: The agent is not authorised to access the server.

An Agent-XML request is being made against the Swivel server but is not permitted to do so. If access shoul;d be allowed create an entry on the Swivel Administration Console under Server/Agents. If an entry exists verified the shared secret is the same on Swivel and the access device.

15 ReportingAPI

15.1 Introduction

Swivel has developed an API, the User Admin API, to allow an external application to perform CRUD (Create, Read, Update, Delete) style operations on users. For most installations, these operations are performed by the User Synchronization job but for larger user populations synchronization may be impractical. The User Admin API addresses this need. In conjunction with the User Admin API, Swivel has also developed another API allowing external applications to perform the functions usually performed by a helpdesk operator from the admin console. These are typically day to day functions not viewed as part of the User Admin API such as user unlock, PIN reset etc. This is the Helpdesk API

There is additionally read-only functionality provided by the **Reporting API** that can be used to read the status (eg idle, locked) of user accounts.

15.2 PRINCIPLES

Following the principles behind the existing Agent API used for authorisation, both of the API?s are based around XML documents for both request and reply. The basic idea is to build a document containing details of operation(s) to be performed and the user(s) on which they are to be performed, submit it to PINsafe via HTTP and PINsafe will reply with a document detailing which operations succeeded and which, if any, failed. All operations via the API will be logged in the standard PINsafe log. Only authorised PINsafe Agents will be able to submit requests. This is in line with Agents used for authorisation. The API is case sensitive, the correct case shown in the included examples.

In order to use the Admin API, you must create a PINsafe Agent for the computer, or sub-net of computers, that will execute AdminAPI commands. Only the computer(s) defined by this Agent can create, read, update or delete users belonging to this Agent, and the Agent cannot read, update or delete users created by other repositories. The Agent can therefore be regarded as another type of repository. It is possible, but not recommended, to manipulate users created by a normal (e.g. XML or Active Directory) repository, by giving the Agent exactly the same name as the repository. Be aware if you do this, however, that any subsequent User Sync of the repository will potentially overwrite modifications made by the Agent. At present, any PINsafe Agent can act as an AdminAPI repository, but future versions of PINsafe will require you to explicitly enable the "Act as Repository" property of an Agent.

These restrictions do not apply to HelpdeskAPI Agents. Although only Agents can use these commands, it is possible to specify that the HelpdeskAPI command operates on a different repository, or on all repositories.

15.2.1 Sending an Admin API request

An Admin API request must be sent as an HTTP(S) request to the pinsafe application, either as a GET or POST request.

For a GET request, the format is

```
https://[swivelserver]:8080/pinsafe/AdminXML?xml=[request body]
```

For a POST request, the URL is

```
https://[swivelserver]:8080/pinsafe/AdminXML
```

and the XML request should form the content of the POST.

15.3 REPORTING API

See Also: Reporting Using Agent-XML How to Guide

Although listed here as a separate API, the Reporting API is actually part of the User Admin API and these elements are only valid as part of an AdminRequest eg

```
<?xml version="1.0" ?>
<AdminRequest secret="MyHelpdeskAgent" version="3.4">
 <Report repository="local">
   <Disabled/>
 </Report>
</AdminRequest>
```

15.3.1 Disabled

Report disabled user accounts. This returns a list of all users from a given repository that are marked as disabled in the PINsafe database. **Note** that this interrogates the PINsafe database and not the repository.

```
 <Report repository="local">
   <Disabled/>
 </Report>
```

15.3.2 Idle

Report users idle since the specified date.

```
 <Report>
   <Idle repository="myRepository" since="12-Mar-2007"/>
 </Report>
```

Note: This does not include users that have been created, but have never used PINsafe.

15.3.3 Locked

Report locked user accounts.

```
 <Report repository=?*?>
   <Locked/>
 </Report>
```

All elements support the optional repository attribute and ?*? is taken to mean ?all repositories.? Therefore the above example will instruct PINsafe to return a list of ALL locked users.

15.3.4 Count

(Since 3.8)

Rather than list users you can request a count of users: For example

```
<AdminRequest secret="secret" version="3.8">
<Report repository="*">
<CountUsers/>
</Report>
</AdminRequest>
```

15.3.5 License information

Check the license details (expiricy and number of users)

```
<SASRequest>
<Version>3.8</Version>
<Secret>secret</Secret>
<Action>GetLicenceInfo</Action>
<LicenceNumber>your-license-key</LicenceNumber>
</SASRequest>
```

15.3.6 AllUsers

(Since 3.8)

Report all users in a repository.

```
<Report repository="*">
 <AllUsers/>
</Report>
```

15.3.7 AllUsersDetailed

(Since 3.9)

Report details of all users in a repository.

```
<Report repository="*">
 <AllUsersDetailed/>
</Report>
```

15.4 Responses

The responses to reporting requests are a list of users that meet the definition of the request.

For example

```
<?xml version="1.0"?>
<AdminResponse>
<Report repository="fred">
<Locked>
<User name="jim1"/>
</Locked>
</Report>
</AdminResponse>
```

The exception to this is the idle report as this also includes for each user the time that they last authenticated

```
<?xml version="1.0"?>
<AdminResponse>
<Report repository="*">
<Idle>
<User name="test" lastLogin="2009-06-03 10:38:46.648"/>
<User name="test2" lastLogin="2009-06-03 16:45:19.413"/>
</Idle>
</Report>
</AdminResponse>
```

Response to the Count report is the number of users.

```
<?xml version="1.0"?>
<AdminResponse>
<Report repository="*">
<CountUsers>
<total>53</total>
<licensed>101</licensed>
</CountUsers>
</Report>
</AdminResponse>
```

Note: The <licensed> element in the Count report response only appears from 3.9.2 onwards.

General Errors and parsing errors are described in Admin API.