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 Agent-XML

2.1 Agent-XML Overview

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.

Swivel can authenticate users with 2 different protocols.

- RADIUS is a standards-based protocol, for further information see RADIUS How To Guide
- AgentXML is a proprietary Swivel standard. Agent-XML allows greater control as it also allows Reporting, Admin and Helpdesk functionality.

2.2 Swivel requirements

2.2.1 Configuring Swivel to use Agent-XML Requests

Swivel must be configured to allow an agent to communicate, details on how to permit an agent to communicate with the Swivel server are here: Agents How to Guide

2.2.2 Repository and Agent configuration

AdminRequest only works with users in the repository with the same name as the corresponding Agent. The work around is to give the Repository exactly the same name as the Agent (or vice versa). From version 3.9.2 Helpdesk requests are allowed for agents that are not repositories.

If you use HelpdeskRequest instead, you can specify a repository attribute on the Read element, but AdminRequest doesn't support this.

The reason for the distinction is that the AdminXML API was designed to support creation and manipulation of users from outside Swivel. The ability to access users from other repositories was an added feature, hence the use of HelpdeskRequest, where the commands are largely read-only.

2.3 Agent-XML

The Agent-XML API is an XML based API used for integrating Swivel with other applications.

There are 4 subsets of the API, that cover the following areas

- Authentication

Authentication, Change PIN, PIN Reset, Start Authentication Session, Request security Strings etc

- Admin functions

Add new users, set user details (eg mobile phone number), synchronise a repository, delete users

- Helpdesk functions

Unlock user, send user new credentials, set user policies etc

- Reporting functions

Retrieving lists of idle users etc

All these APIs follow the structure described in this article.

2.4 Principles

The APIs 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 Swivel via HTTP and Swivel 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 Swivel log.

Only authorised Swivel 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.

Requests are sent to Swivel via an HTTP POST using http or https depending on the configuration of the Swivel server. For appliances the default will be https over port 8080.

The requests are posted to the Swivel context followed by AgentXML for the authentication API and AdminXML for the Admin, Helpdesk and reporting API.

For example http://<ip address>:8080/pinsafe/AgentXML?xml=<?xml version="1.0"?>

or

https://<ip address>:8080/pinsafe/AdminXML?xml=<?xml version="1.0"?>

2.5 Structure

The structure of a request that all API requests are contained within a request element. This request element specifies the type of request and also includes the shared secret for agent authentication.

e.g. an authentication request would be contained within an SASRequest tag

```
<?xml version="1.0" ?>
<SASRequest secret="MyAdminAgent" version="3.4">
      .
      .
</SASRequest>
```

Whereas an admin API request would be contained within an AdminRequest element.

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

The responses from the Swivel server are similarly contained within response tags eg

```
<?xml version="1.0" ?>
<SASResponse secret="MyAdminAgent" version="3.4">
      .
      .
</SASResponse>
```

and

```
<?xml version="1.0" ?>
<AdminResponse secret="MyAdminAgent" version="3.4">
      .
      .
</AdminResponse>
```

Within these elements for elements that specify the particular request. For details of these elements refer to the article relating to that part of the API.

2.6 Commands

Authentication for more information see: Authentication

The following are valid values for the <Action> element:

- changepin : Change Users Credentials
- changepassword : Change a user password (Sentry or repository)
- checkpassword : Check that a password is correct (Sentry or repository)
- exists : Check if a user have an account.
- existsbyattribute : Check if a user has an account based on custom attributes
- increaselock : Increase a user's lock count (simulate a login failure)
- getuserattribute : Return the value of a named attribute for a user
- getusernamebyattribute : Return the username given an attribute name and value
- hascachedpassword : Check if a user has an AuthControl Desktop password cached
- killsession : terminate a Sentry session
- logevent : add a message to the Sentry logs (available from 4.2.1 onwards)
- login : Perform Sentry authentication
- loginbyattribute : Perform Sentry authentication using an alternative username attribute
- oathsync : resynchronise an OATH token
- ocraverify : Verify an OATH OCRA response to challenge
- ondemandmessage : send an On Demand message
- ping : Ping Swivel application
- provision : provision a mobile application given the provision code
- provisioncode : request a provision code
- reset : Perform a PIN reset (Note: the user must have a transport to receive the new PIN)
- resetcode : Request a PIN Reset code
- sendconfirmationcode
- sessionstart : Start an authentication session
- securitystrings : Request a set of security strings
- transportindex
- validateconfirmationcode

Helpdesk for more information see: Helpdesk functions

<Reset> : Send user new credentials (Note: the user must have a transport to receive the new PIN)

<Strings> : send dual channel security string(s) to user

<Update> : Change user policies or credentials

<OathSync> : Synchronise an OATH token

<PurgeDeleted> : Purge deleted users

Admin for more information see: Admin functions

<Create> : create a new user and optionally set user details.

<Delete> : delete a user.

<Read> : read the details of a given user.

<Reset> : Send user new credentials (Note: the user must have a transport to receive the new PIN)

<Sync> : Synchronise with a repository

<Update> : Change user policies, credentials, transports etc

<PurgeDeleted> : Purge deleted users.

Reporting for more information see: Reporting functions and Reporting Using Agent-XML How to Guide

<disabled> : Report on disabled users

<idle> : Report on idle users

<locked> : Report on locked users

2.7 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 should 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. See also Agents How to Guide. Try with the IP address instead of hostname, the hostname entry may be case sensitive

AgentXML request failed, error: The XML request sent from the agent was malformed.

The Agent XML request contains an error check the format. Spaces may also cause errors.

ADMIN_ERROR_UNSUPPORTED_VERSION

The agent version is incorrect.

https://127.0.0.1:8080/pinsafe/AdminXML?xml=<AdminRequest%20secret="secret"%20version="3.7"><Report%20repository="local"><Idle%20since="01-jul-201

Swivel 3.7 should use 3.6 for the version.

https://127.0.0.1:8080/pinsafe/AdminXML?xml=<AdminRequest%20secret="secret"%20version="3.4"><Report%20repository="local"><Idle%20since="01-jul-201

Agent XML Versions

Swivel Version	Agent-XML Version
3.7	3.6
3.6	3.6
3.5	3.4
3.4	3.4

Malformed XML

This can occur when a ../AgentXML URL request is used to send an Admin request. Use the ../AdminXML instead.

2.8 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".

3 API How to Guide

The Agent-XML API is an XML based API used for integrating PINsafe with other applications.

There are 4 subsets of the API, that cover the following areas

- Authentication

Authentication, Change PIN, PIN Reset, Start Authentication Session, Request security Strings etc

- Admin functions

Add new users, set user details (eg mobile phone number), synchronise a repository, delete users

- Helpdesk functions

Unlock user, send user new credentials, set user policies etc

- Reporting functions

Retrieving lists of idle users etc

All these APIs follow the structure described in this article.

3.1 Principles

The APIs 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.

Requests are sent to PINsafe via an HTTP POST using http or https depending on the configuration of the PINsafe server. For appliances the default will be https over port 8080.

The requests are posted to the pinsafe context followed by AgentXML for the authetnication API and AdminXML for the Admin, Helpdesk and reporting API.

For example http://<ip address>:8080/pinsafe/AgentXML or https://<ip address>:8080/pinsafe/AdminXML

3.2 Structure

The structure of a request that all API requests are contained within a request element. This request element specifies the type of request and also includes the shared secret for agent authentication.

eg an authentication request would be contained within an SASRequest tag

```
<?xml version="1.0" ?>
<SASRequest secret="MyAdminAgent" version="3.4">
	.
	.
</SASRequest>
```

Whereas an admin API request would be contained within an AdminRequest element.

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

...or else a HelpdeskRequest element.

```
<?xml version="1.0" ?>
<HelpdeskRequest secret="MyAdminAgent" version="3.4">
	.
	.
</HelpdeskRequest>
```

The syntax of AdminRequest and HelpdeskRequest are more or less identical. The major difference is that AdminRequest can make major changes to the database, such as adding and deleting users, whereas HelpdeskRequest is largely read-only, although it does support some changes to existing users, such as PIN change. Conversely, AdminRequest can only affect users in the repository with the same name as the Agent, so the Agent must have "Can Act as Repository" set to Yes. HelpdeskRequest can affect any users.

The responses from the PINsafe server are similarly contained within response tags eg

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

```
      .
</SASResponse>
```

and

```
<?xml version="1.0" ?>
<AdminResponse secret="MyAdminAgent" version="3.4">
      .
      .
</AdminResponse>
```

Within these elements for elements that specify the particular request. For details of these elements refer to the article relating to that part of the API.

3.3 Commands

Authentication

<changePIN> : Change Users Credentials

<exists> : Does a user have an account.

<login> : Perfrom PINsafe authentication

<ping> : Ping PINsafe application

<reset> : Perform a PIN reset

<resetcode> : Request a PIN Reset code

<startsession> : Start an authentication session

<securitystrings> : Request a set of security strings

Helpdesk functions

<reset> : Send user new credentials

<strings> : send dual channel security string(s) to user

<update> : Change user policies or credentials

Admin functions

<create> : create a new user and optionally set user details.

<delete> : delete a user.

<read> : read the details of a given user.

<reset> : Send user new credentials

<sync> : Synchronise with a repository

<update> : Change user policies, credentials, transports etc

Reporting functions

<disabled> : Report on disabled users

<idle> : Report on idle users

<locked> : Report on locked users

3.4 Examples

We have a number of applications that use these APIs, available as Windows executable programs. These can be found on a separate page.

4 Java Client

4.1 Introduction

This document describes the Swivel client library for Java applications.

The current version of the client library can be downloaded from here.

4.2 Using the Library

Full documentation of the library is ongoing. Meanwhile, here are some examples of using the library:

4.2.1 Requesting a TURing image

The following code snippet will return a random security string as a TURing image:

```
PINsafeRequest req = new PINsafeRequest("https://swivel:8443/proxy");
req.singleChannelImageByUsername("user");
if (req.send()) {
 byte[] imageBytes = req.getResponseBytes();
 // imageBytes returns the TURing image as a JPG,
 // or if animation is enabled, an animated GIF.
}
```

4.2.2 Requesting a Dual Channel Message

The following code snippet will request a security string to be sent to the user's designated device:

```
PINsafeRequest req = new PINsafeRequest("https://swivel:8443/proxy");
req.dualChannelMessageByUsername("user");
if (req.send()) {
 byte[] confirmBytes = req.getResponseBytes();
 // imageBytes returns the confirmation image as a JPG.
 // This is only required if you wish to display this to the user.
}
```

4.2.3 Authenticating a User

The following code snippet will authenticate a user given the username and one-time code. It is assumed that the user has no Swivel password. If Swivel passwords are used, specify the entered password as the second argument to the login method. The authentication URL must be to pinsafe on port 8080.

```
// The first argument to the constructor is the PINsafe URL.
// The second argument is the agent secret.
AgentXmlRequest req = new AgentXmlRequest("https://swivel:8080/pinsafe", "secret");
// setIgnoreSSLErrors(true) will ignore SSL certificate errors.
req.setIgnoreSSLErrors(true);
req.login(user, "", otc);
if (req.send()) {
 boolean authenticated = req.actionSucceeded();
}
```

5 Microsoft IIS version 7 ASP.NET Forms Integration

5.1 Introduction

Swivel allows ASP.NET application authentication using Agent-XML for IIS 7 and IIS 6 ASP.NET

NOTE: the method listed here uses standard ASP.Net forms-based authentication to authenticate to PINsafe. We now have an alternative solution that uses a HTTP module. This might be an easier solution than the manual method described below, as all installation and configuration is done using provided applications. Documentation for this solution can be found here.

5.2 Prerequisites

PINsafe

ASP.NET application

ASP.NET Server

5.3 Baseline

PINsafe 3.7

IIS6 and IIS7

5.4 Architecture

The ASP.NET application makes authentication requests against the PINsafe server by Agent-XML.

5.5 ASP.NET Sample Files

ASP.NET Sample File is available here: ASP.NET Sample File

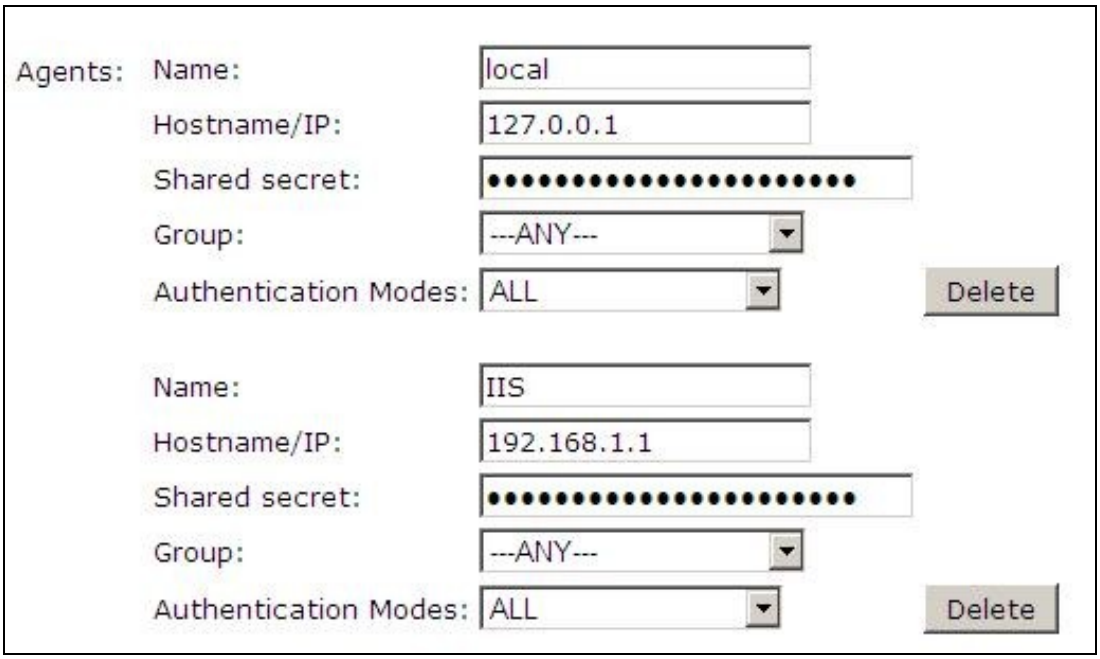
ASP.NET Sample file for 2008 server is available here: ASP.NET for 2008 Server

The pinsafe folder contains an example login page, plus aspx pages which render a TURing image or request a dual channel image.

5.6 PINsafe Configuration

5.6.1 Configure a PINsafe Agent

1. On the PINsafe Management Console select Server/Agent
2. Enter a descriptive name for the Agent
3. Enter the IP address or hostname of the server on which the ASP.NET will be running
4. Enter the shared secret used above on the ASP.NET
5. Click on Apply to save changes



Note: Session creation by username is not required for this integration as PINsafe can use session ID.

5.7 ASP.NET Configuration

5.7.1 Integrating the ASP.NET

First of all, extract the sample zip file to a temporary location. There should be 2 folders:

- App_Code
- pinsafe

and one file:

- web.config.

Copy the pinsafe folder and its contents into the ASP.NET application you want to protect or the root of the website to protect the entire website. It is important that the folder is contained within the application, and is not an application in its own right. You will need to set IIS (or other ASP.NET server) to allow anonymous access to the pinsafe folder, and you may need to modify permissions on the files to ensure that the default IIS (or other ASP.NET server) user has read access.

Copy the contents of the App_Code folder into the App_Code folder of the application or create one if it doesn't already have one.

Edit the web.config file for the application, and add the contents of the enclosed web.config in the appropriate locations. You will need to change the PINsafe server settings as appropriate.

5.7.2 Configure the web.config file

This file contains the information for communication with the PINsafe server. The options are displayed below:

PINsafeServer: The IP address or hostname of the PINsafe server or appliance

PINsafePort: The port used for communication, usually 8080

PINsafeContext: The install name of pinsafe, usually pinsafe

PINsafeSecret: The shared secret key, which must be the same as that entered on the PINsafe server

PINsafeSecure: This is if the connection to the PINsafe server is https for SSL or http. The default value is true, which is for https

PINsafePassword: This is to display the password field, the default value of false will not display a password field

PINsafeImage: This is to display a button to generate a Single Channel Image of the security string

PINsafeMessage: This is to display a button to generate a Dual Channel security string to be sent to the user

PINsafeAcceptSelfSigned: If self signed certificates are accepted, defualt is yes

NOTE: As the requests are made using Agent-XML, they must be made to the pinsafe appliance on port 8080 and the context of pinsafe and not the proxy port of 8443. Security is usually provided by the IIS server proxying the request to the PINsafe server.

Default Settings, suitable for a software install of PINsafe are:

```
<add key="PINsafeServer" value="pinsafe_server" />
<add key="PINsafePort" value="8080" />
<add key="PINsafeContext" value="pinsafe" />
<add key="PINsafeSecret" value="secret" />
<add key="PINsafeSecure" value="true" />
<add key="PINsafePassword" value="false" />
<add key="PINsafeImage" value="true" />
<add key="PINsafeMessage" value="false" />
<add key="PINsafeAcceptSelfSigned" value="true" />
```

Appliance settings are likely to be:

```
<add key="PINsafeServer" value="pinsafe_server" />
<add key="PINsafePort" value="8080" />
<add key="PINsafeContext" value="pinsafe" />
<add key="PINsafeSecret" value="secret" />
<add key="PINsafeSecure" value="true" />
<add key="PINsafePassword" value="false" />
<add key="PINsafeImage" value="true" />
<add key="PINsafeMessage" value="false" />
<add key="PINsafeAcceptSelfSigned" value="true" />
```

5.7.3 Additional web.config file IIS7 Options

The loginUrl setting assumes that you are protecting the entire website. If you are only protecting an application, add the path for that application to this URL. For example, to protect an application with URL "/secure", loginUrl="/secure/pinsafe/Login.aspx".

The <modules> section is not relevant if you are protecting an application that is ASP.NET only. These changes allow ASP.NET authentication to be used for static web pages as well as .aspx pages. This is a new feature of IIS7.

5.7.4 Enabling Authentication

For IIS, open the IIS manager, locate the website or application that you are protecting, and double-click the Authentication icon. Make sure that anonymous authentication is disabled, and that forms authentication is enabled, and the URL is as set earlier. Go to the pinsafe sub-folder, select Authentication under there, and make sure anonymous authentication is enabled (you need to be able to access the login pages anonymously).

5.8 Additional Configuration Options

5.9 Testing

Navigate to the login page. Attempting to login with a correct username and password but no one time code should result in failure. Only when a correct PINsafe one time code is entered should the user be logged in. If the Single Channel button is displayed then an image should appear.

5.10 Troubleshooting

To verify the Single Channel Image works, on the ASP.NET server enter the following into a web browser, which should display a Turing image if the sever is functioning correctly:

For a PINsafe appliance install:

https://<pinsafe_server_ip>:8080/pinsafe/SCImage?username=test

For a software only install see Software Only Installation

5.11 Known Issues and Limitations

Requesting a Security String Index would require modification of the login page for an existing button. See also Multiple Security Strings How To Guide

5.12 Additional Information

For assistance in the PINsafe installation and configuration please firstly contact your reseller and then email Swivel Secure support at support@swivelsecure.com

6 Microsoft IIS version 7 Integration

6.1 Overview

This document outlines the steps required to integrate the Internet Information Server (IIS) with Swivel using dual or single channel authentication. The Swivel install requires configuring an agent on the Swivel server and setting up a shared secret with the IIS server to allow communication for authentication. An ISAPI filter installed on the IIS server allows access to protected resources through the Swivel authentication.

NOTE: This document refers to the version of the filter numbered 1.2, and the configuration application with the same version number. 32-bit and 64-bit versions of the filter are available. Version 1.3.4, with PINpad support, is available for 64-bit only.

If Windows 2008 Server server is being used, or only ASP.Net applications are being protected an alternative authentication is available, see Microsoft IIS version 7 ASP.NET Integration

6.2 Prerequisites

Internet Information Server on Windows server 2008, 32-bit or 64-bit operating system.

Swivel server

The appropriate Swivel ISAPI filter software can be downloaded from here, depending on your operating system:

The latest release is version 1.3.9. Support for PINpad is included from 1.3.0 onwards. Version 1.3.4 adds PINpad support for change PIN as well:

- 64-bit ISAPI Filter
- 32-bit ISAPI Filter

These links refer to version 1.2 of the filter, provided for legacy purposes.

- 32-bit ISAPI Filter
- 64-bit ISAPI Filter

6.3 IIS Filter Version History

1.2 32 bit and 64 bit

1.3.3 (64-bit only): PINpad support added

1.3.4 (64-bit only): added PINpad support for ChangePIN

1.3.5 (64-bit only): enhancements to ChangePIN support

1.3.6 (64-bit only): added a default logout page

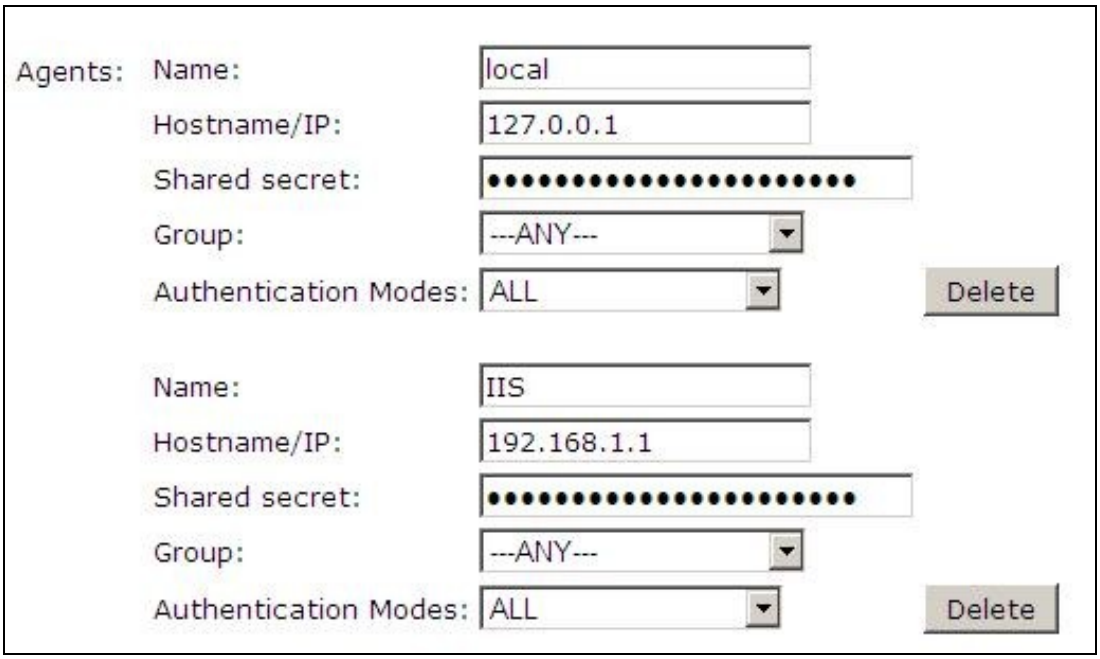
1.3.7-9: various bug fixes

6.4 Swivel Configuration

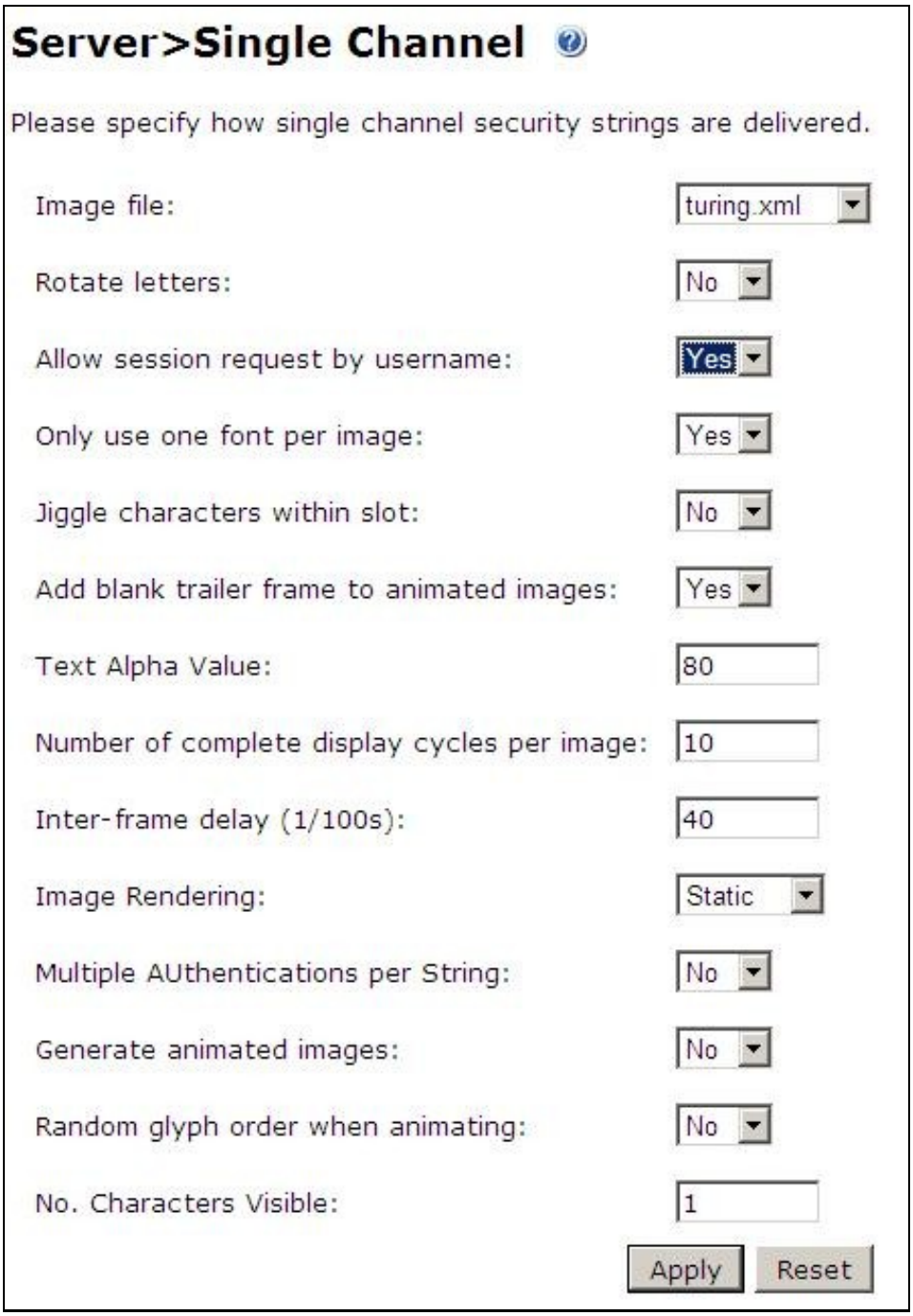
On the Swivel server configure the agent that is permitted to request authentication. On the Swivel Administration Console select from the server menu Agents and enter the details of the IIS server IP address and a shared key, then click on apply.

Example:

```
Name : IIS server 1,
Hostname/IP : 192.168.1.1,
shared secret : secret
```



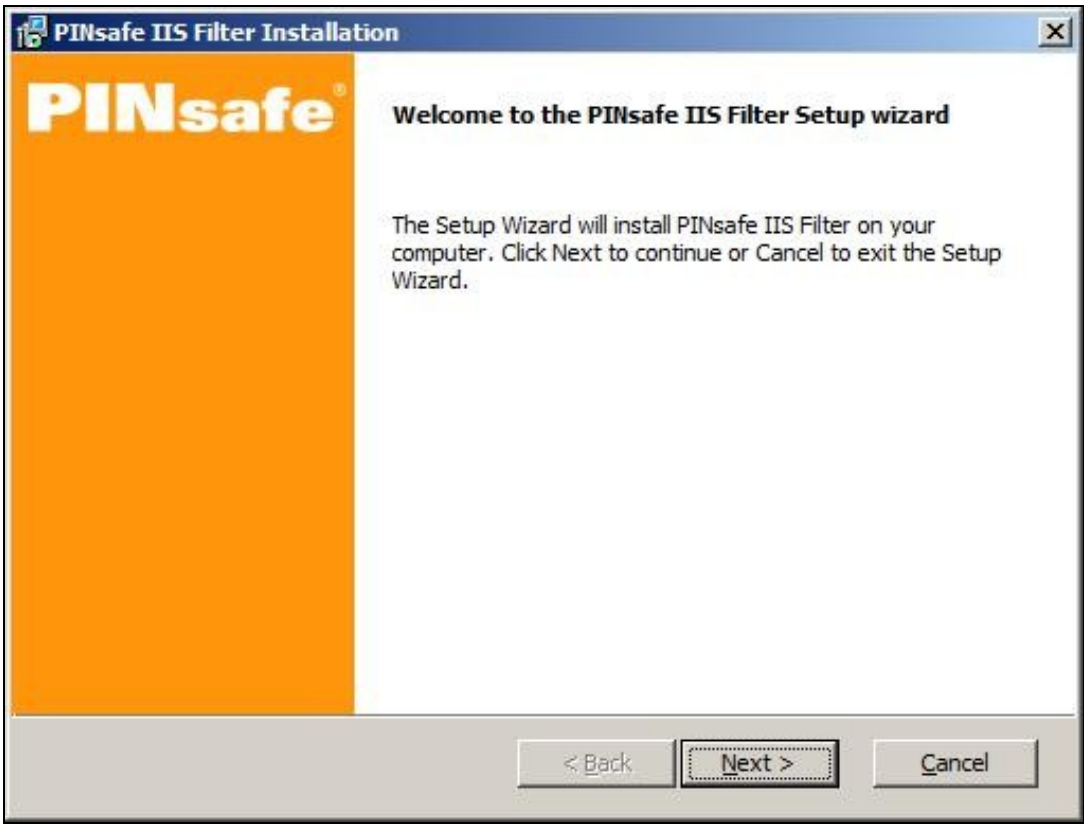
If Single Channel communication is to be used, select from the Swivel Administration Console Single Channel, and set the Allow image request by username to Yes then click on apply.



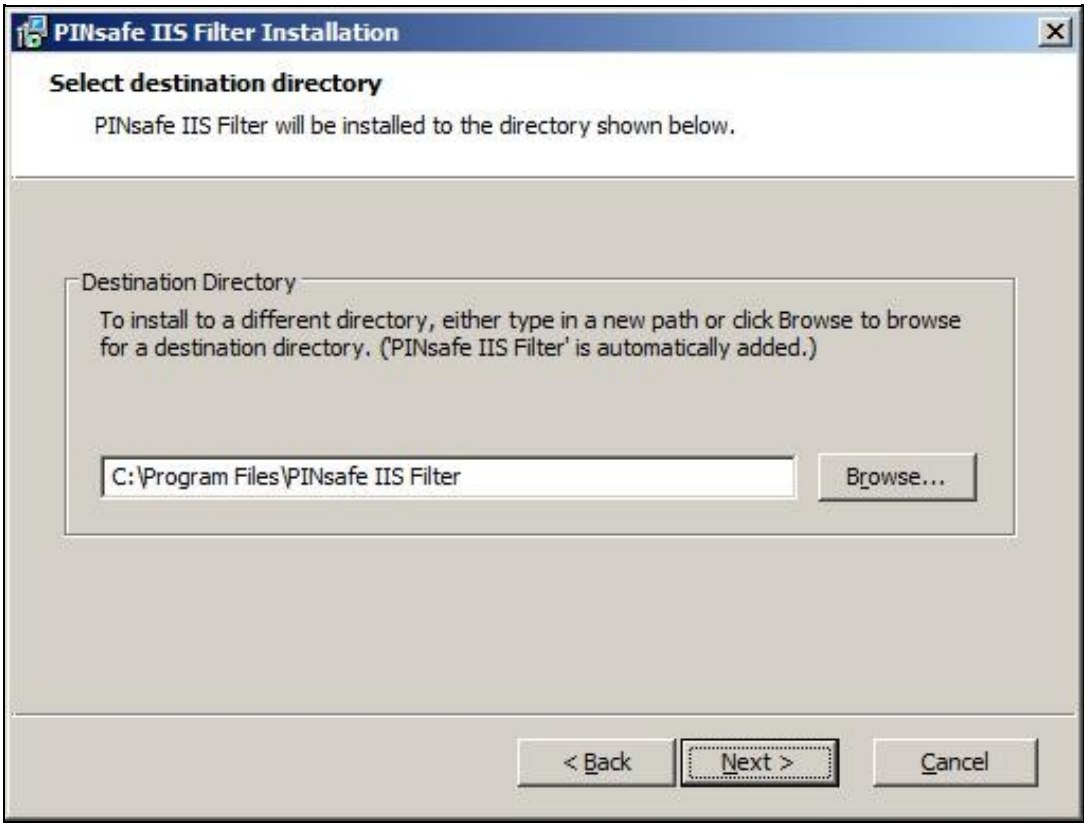
6.5 Configuring the IIS Server

6.5.1 Install the Swivel Filter

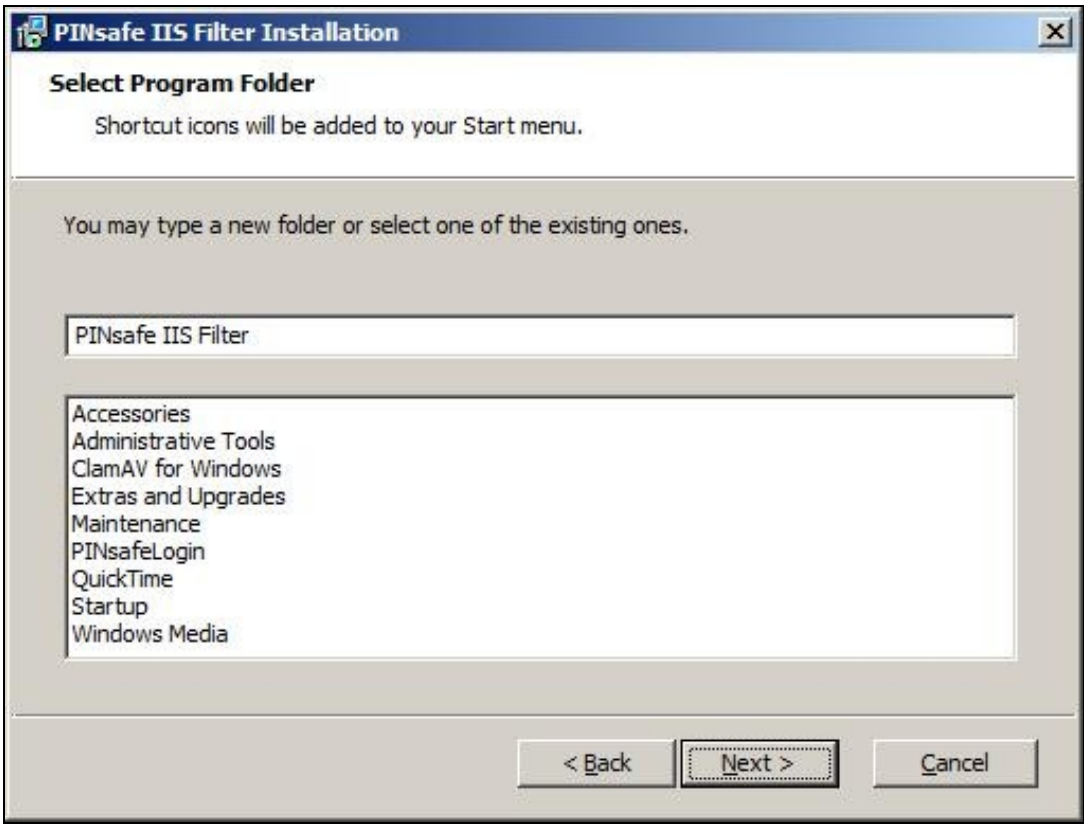
1. On the IIS server run the PINsafeISFilter.exe. The IIS filter may need to be run as an Administrator user (The filter needs to be installed by IIS running as Administrator user but it can run as a normal user).



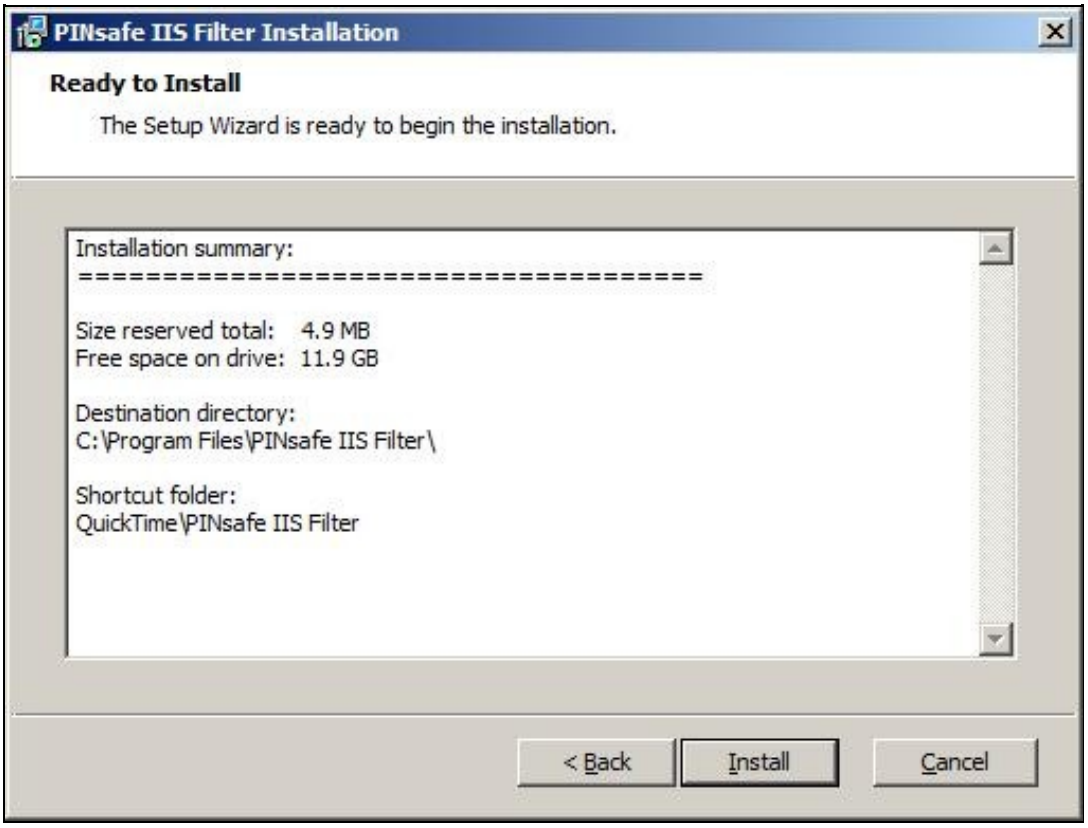
2. Choose the Path to Install to such as C:\Program Files\PINsafe IIS Filter



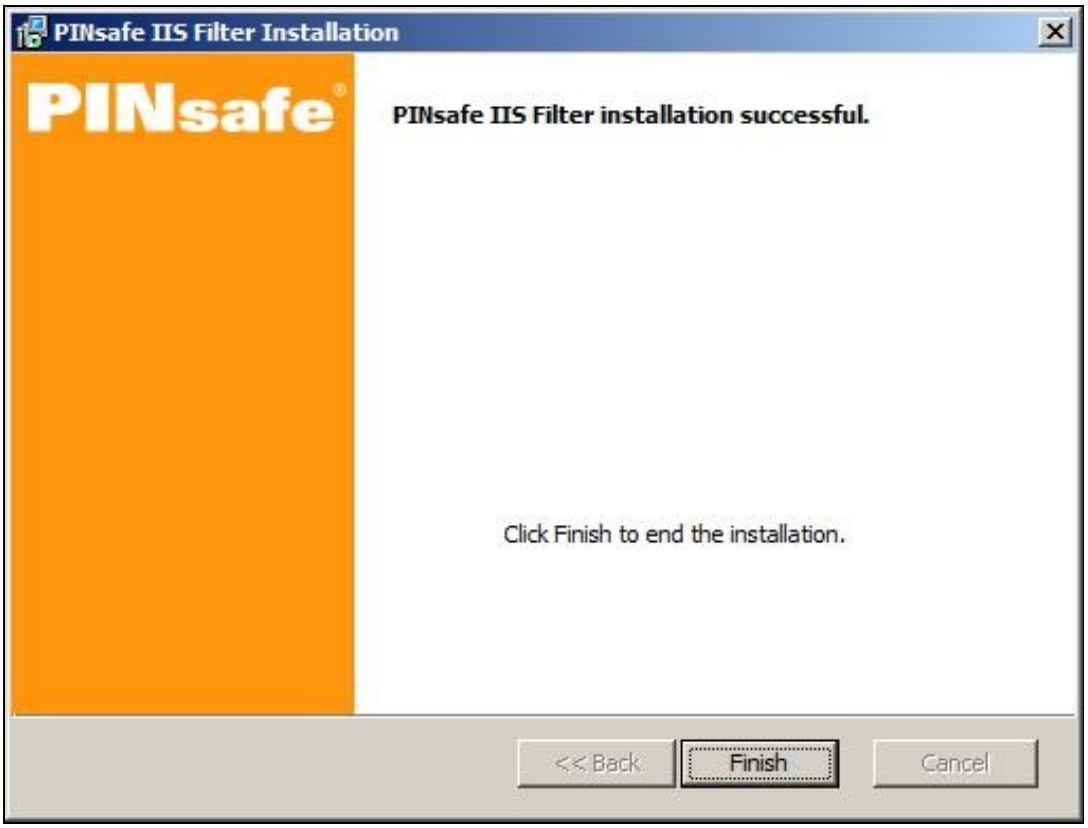
3. Select Start Menu Folder



4. When details are correct click on Install

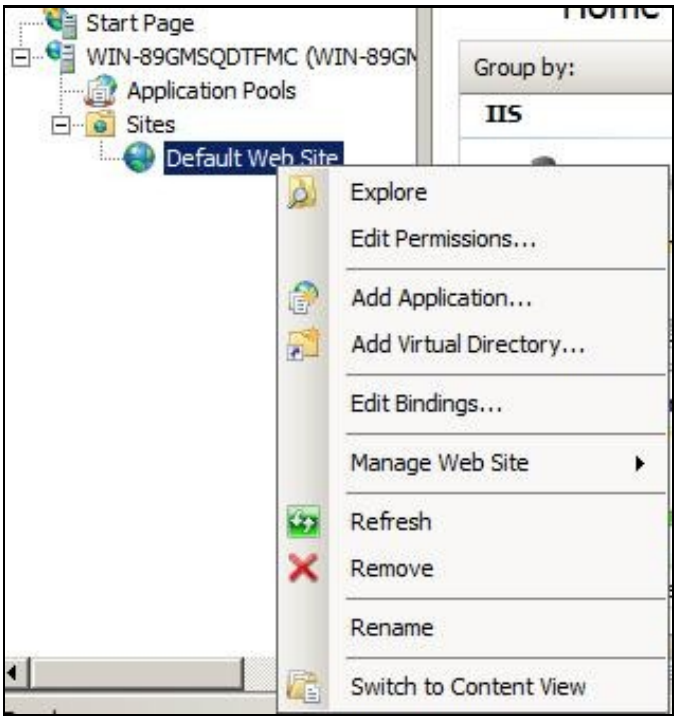


5. If the error ?Incorrect Command Line Parameters? is seen click on OK

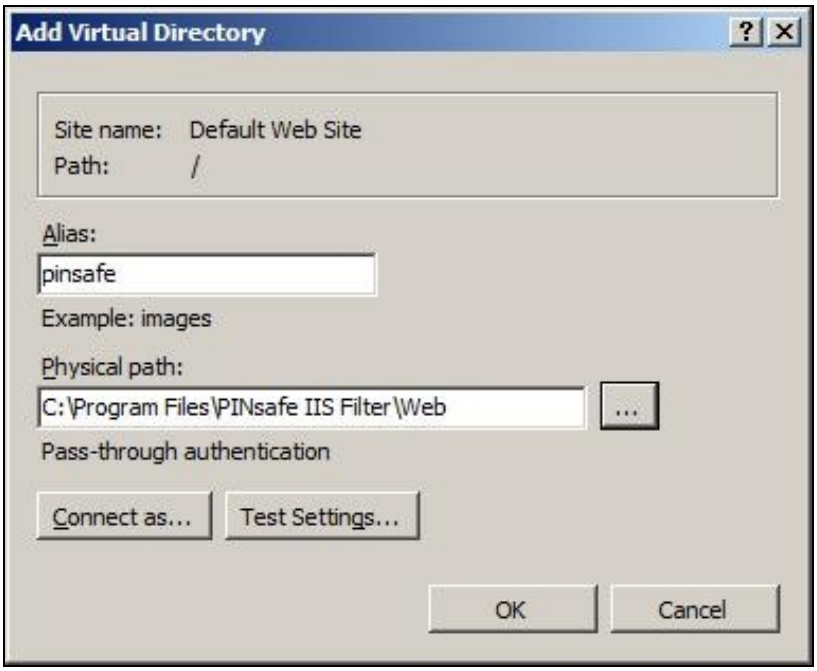


Create a PINsafe virtual directory

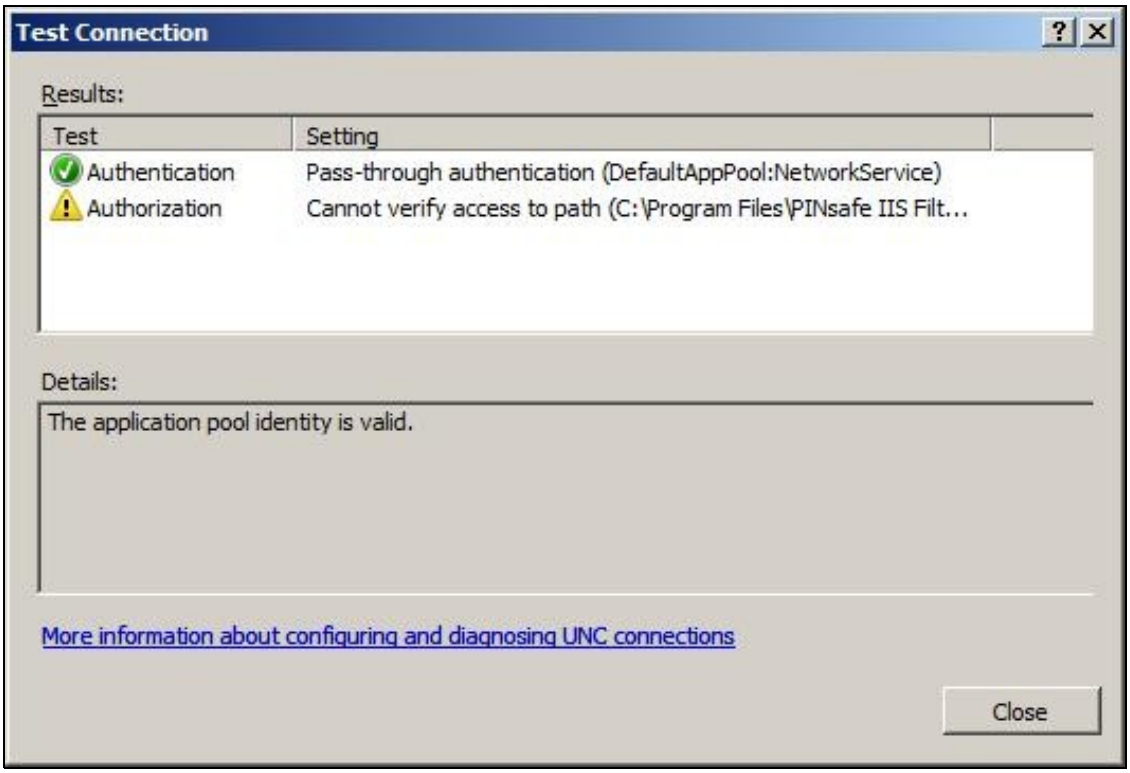
1. On the Internet Information Services Manager right click on the website and select Add Virtual Directory



2. Create an Alias called PINsafe



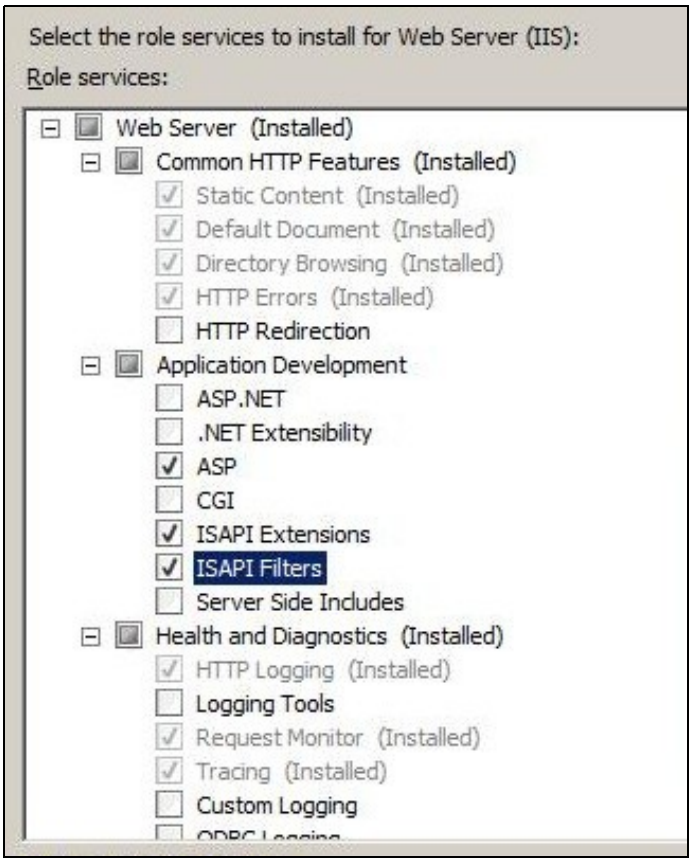
3. Point the path to the PINsafe directory Web folder, by default C:\Program Files\PINsafe IIS Filter\Web. Test Connection verifies the path, and Connect As allows Application User for pass through authentication.



4. Set the permissions to Read and Run Scripts

6.5.2 Installing the ISAPI Filters, extensions and ASP on IIS

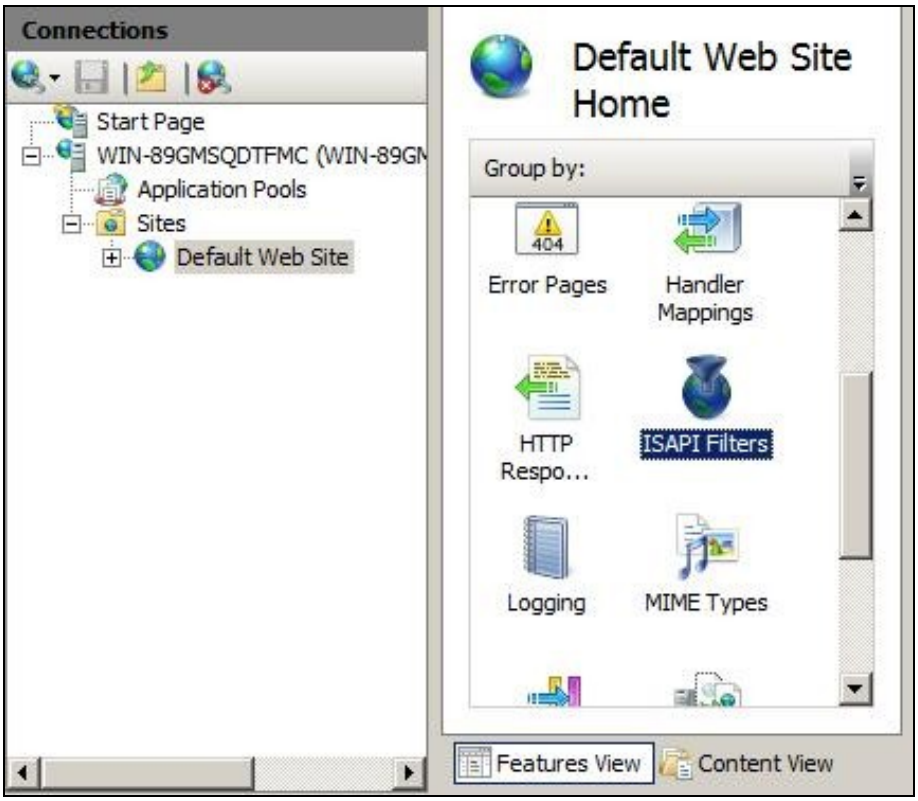
This requires the ISAPI filters, ISAPI extensions and ASP to be installed. To verify or install these, for Windows 2008, on the IIS server start the Server Manager by selecting Start/Administrative Tools/Server Manager then expand the tab for Roles, click on the Web Server (IIS), then look under Role Services to ensure that the *ISAPI Filters*, *ISAPI Extensions* and ASP are installed. If it is not click on Add Role Services and add them.



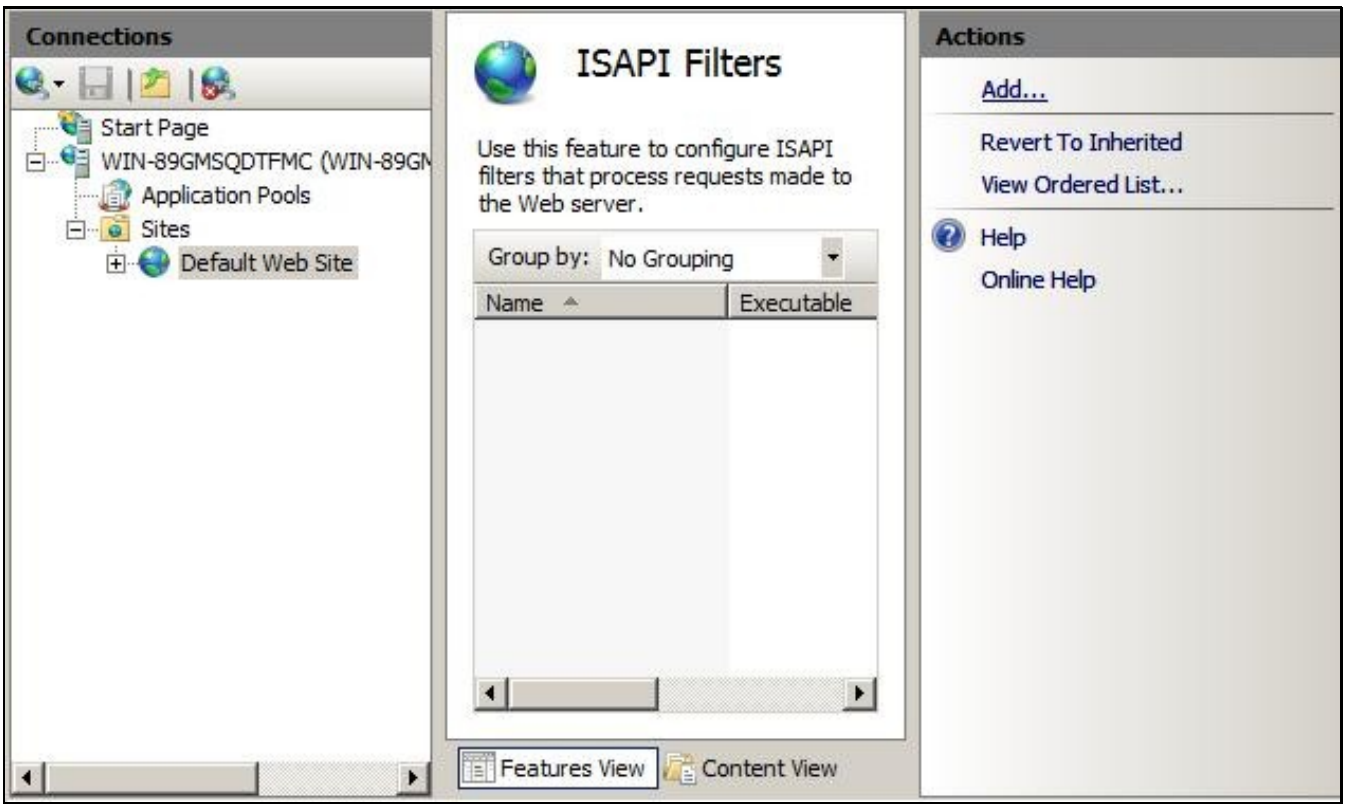
6.5.3 Install the Swivel ISAPI Filter

1. On the Internet Information Services Manager Select the website

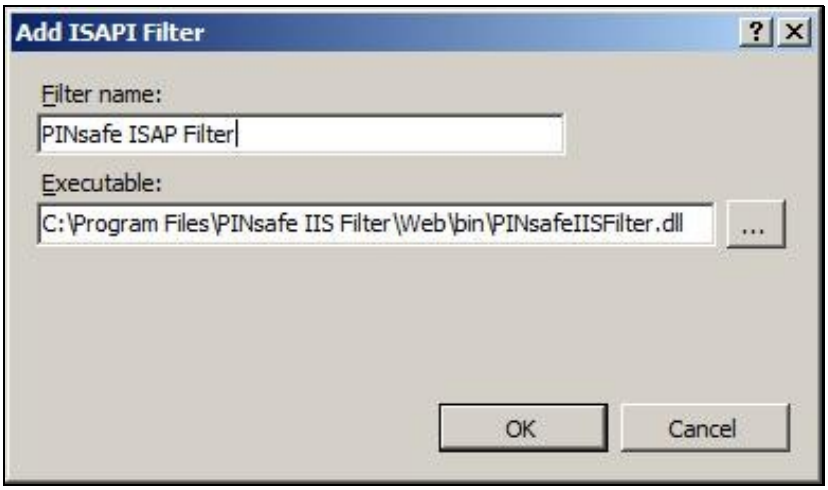
2. Select ISAPI filters by double clicking on the ISAPI filters icon



3. Under Actions select Add



4. Select the Path to the Swivel ISAPI filter. Note that the actual file you require will be PINsafeIISFilter.dll, located in the sub-folder Web\bin of the installation folder. Enter a name for the Filter such as *PINsafe ISAPI Filter*. When information is complete click on Ok.



5. Ensure the Swivel ISAPI filter is the top filter by selecting the 'View Ordered List...'

6.5.4 Configure the ISAPI filter

1. Select Start Programs/PINsafe IIS Filter/Filter Configuration. This will look for the location of config.xml, this will be created when first used and this must be located in web/bin.

Note: If the Swivel Filter Configuration does not exist in the Start Menu, it can be started by running it from its install location. The default install location is C:\Program Files\PINsafe IIS Filter\Web\bin\ConfigApp.exe

2. You will be asked to select the location of the configuration file. It is important that you select the right location. It should be in the same folder as the ISAPI filter. Initially, this file will not exist, but will be created as a result of running this configuration application.

PINsafeIISFilter Options

PINsafeServer: The PINsafe Server tab contains settings which define the Swivel server which will be used to authenticate users.

Hostname/IP: The name or IP address of the Swivel server.

Port: The port number used by the Swivel server (normally 8080, or 8443 for HTTPS).

Context: The context (i.e. web application name) of the Swivel instance on that server

Secret: The common secret used to communicate with the Swivel server. This value must be the same as the secret defined for the Swivel agent configured earlier.

SSL enabled: Tick this box to require SSL (HTTPS) communication with the Swivel server.

Permit self-signed certificates: Tick this box to allow SSL certificates to be self-signed. This also ignores other certificate errors, such as site names not matching.

Authentication: The Authentication tab contains the following settings:

Idle time (s): The time (in seconds) for which the authentication cookie will be valid if the web page is not used.

Username header: The name of a cookie which will pass the username of the authenticated Swivel user. If this value is blank, no cookie will be provided.

Single: Indicates that single channel security strings (i.e. TURing image) are permitted.

Dual: Indicates that dual channel security strings (i.e. via e-mail, SMS) are permitted.

On-demand dual: Indicates that the login page should display a button to request dual-channel security strings.

Display password fields: Indicates that the login page should show a field for Swivel password as well as OTC.

Permit self-reset: Indicates that the user self-reset page should be enabled.

Exclusions and Inclusions: Use the inclusion and exclusion tabs to enter which paths should be protected by Swivel:

Included paths: This is a list of paths within the current website which require Swivel authentication. If this list is empty, the entire website will be protected except as indicated by the Exclusions tab. Paths should be one per line.

Excluded paths: This is a list of paths within the current website which should be exempt from Swivel authentication. This is only relevant if the included paths list is empty, in which case all paths not on this list will be protected by Swivel.

Excluded addresses: This is a list of IP addresses which are exempt from Swivel authentication. All requests from these addresses are passed through without authentication.

Misc: On the Misc tab, edit any custom paths as follows:

Default path: This is the path to which authenticated requests are directed if the login page is targeted directly. If a user tries to access a protected page, she is redirected to the login page, and after authentication, back to the page she was trying to access. If the user requests the login page directly, she will be redirected to this location after authentication.

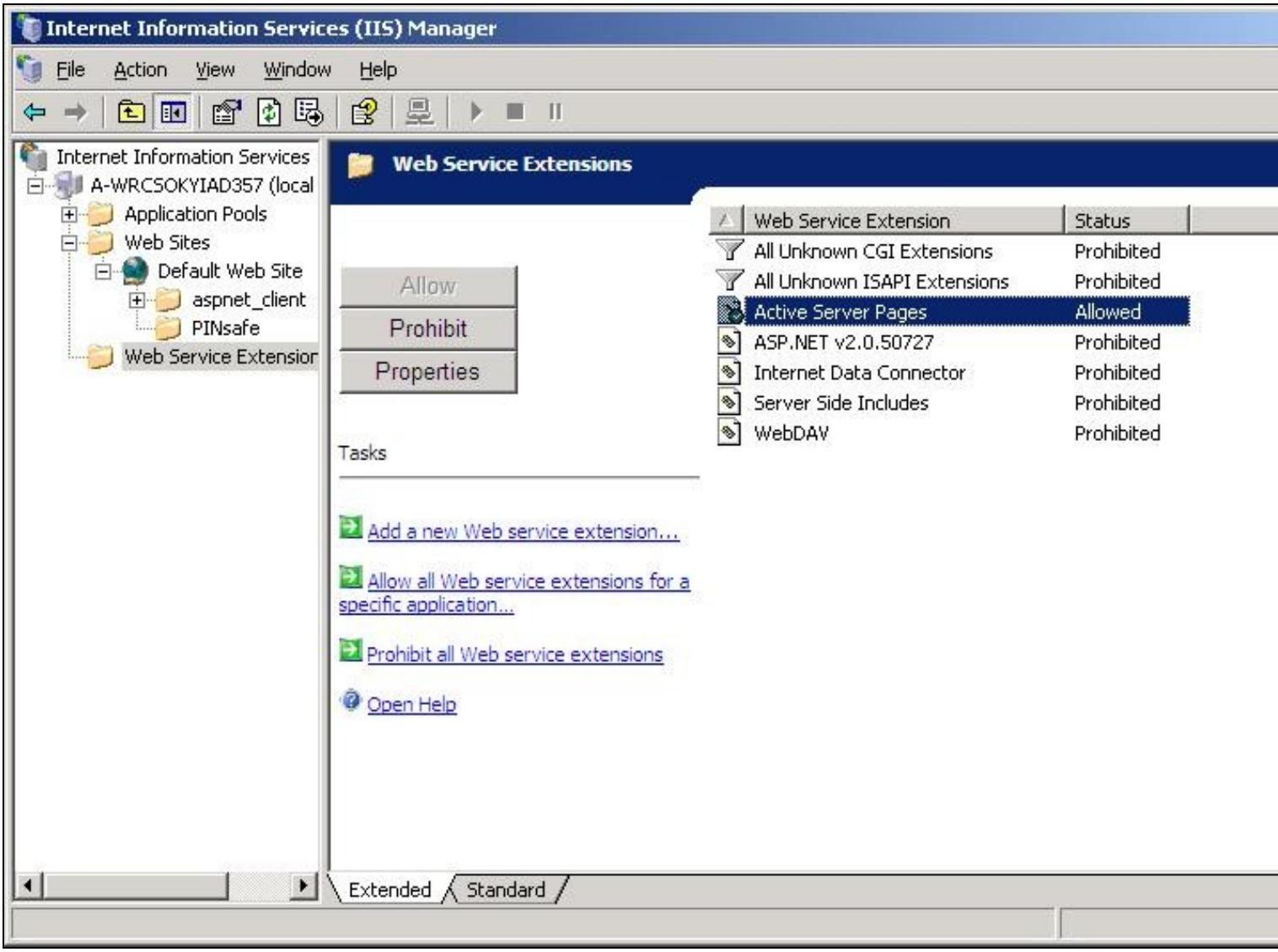
Logout path: Requesting this path will result in the user being logged out. Subsequent requests will require re-authentication, if relevant. If this path is empty, users can only be logged out by closing the browser, or if the authentication times out. Note: The logout path must be an included file location.

Virtual web path: This is the path to the Swivel authentication pages. See the next section for details on setting this up. You should normally set this to be ?/pinsafe?, unless you have a particular reason not to.

Help URL: The URL for Swivel IIS filter help. The filter does not come with help pages as standard, so this should only be filled in if help pages have been provided by the reseller or end user.

After configuration is complete Apply the settings and restart the World Wide Web Publishing Services.

Allow Active Server Pages: to verify this, select the Internet Information Services Manager expand the required server then click on Web Service Extension.



6.6 Installing the Filter on Multiple Websites

Most of the instructions above assume that you are installing the IIS filter on the single default website of a web server. However, if you want to use the filter on multiple websites on the same server, you need to carry out a few extra steps.

Firstly, if all the settings on all the websites are exactly the same, you can configure it once, and then carry out the same steps to activate the filter on each website.

If, however, you need different settings for each website, you will need to do the following for all except the first website:

1. Make a copy of the entire Web folder, including the bin sub-folder. You can copy it as a sub-folder of Swivel IIS filter, with a different name, or you can put it somewhere completely different. It is important, however, that you keep the structure the same ? all the .asp files must remain intact, and the filter DLL must be in a sub-folder called ?bin?.

2. When you come to configure each filter, navigate to the bin sub-folder of the copy you have just made. Otherwise, configuration is exactly as before.

3. When selecting the IIS filter to install, and also when defining the virtual directory for Swivel web pages, you should use the copy you have just created, rather than the original. This is important, as both the filter and the web pages look for the configuration file relative to their own location.

6.7 Testing

Browse to a web page that has been configured for protection. This should display a Swivel login dialogue:

Enter the Username.

For dual channel, enter the One Time Code:

Or click start session to enter a single channel OTC. The Swivel log will record that a single channel session has started.

If authentication is successful it should redirect to the login page. If failed an error message will appear. The Swivel log will record any successful log attempt for the agent.

6.8 Uninstalling the filter

To remove the Filter, remove role services that are not required by other applications, to do this for Windows 2008, on the IIS server start the Server Manager by selecting Start/Administrative Tools/Server Manager then expand the tab for Roles, click on the Web Server (IIS), then look under Role Services to remove the *ISAPI Filters*, *ISAPI Extensions* and ASP are installed. The system will require a restart to complete.

From the IIS Manager right click on the Swivel Virtual Directory, then select Remove, Click on Yes to Confirm.

To uninstall the Swivel IIS Filter, choose Start/All Programs/PINsafe IIS Filter/PINsafe IIS Filter Uninstaller, then click Yes on the confirmation to uninstall.

The Swivel Filter config may be left after uninstalling, so to completely remove this, remove the folder Program Files\PINsafe IIS Filter.

6.9 Troubleshooting

Verify the correct filter version has been installed for the operating system i.e. 32 bit or 64 bit.

Check for error messages in the Swivel log

Check the IIS log messages

Filter Install issues: The IIS service needs to run as an administrator in order to install the filter. Once the filter is installed, it will run as the normal user. Open the Services applet and locate World Wide Web Publishing. Select properties, then go to the Log On tab. Assuming you are logged in as an administrator, select Local System account. Then restart the service. Hopefully, this will start the filter. You can then switch back to the default account and restart again.

If you are not redirected to the Swivel login page when trying to access a protected page, open IIS manager and check that the ISAPI filter is installed and has loaded properly (there should be a green arrow next to it, and the priority should be ?High?). If this is not the case, restart IIS, if you have not already done so, and try again. The filter doesn't try to install until you try to access a web page from that website, so try that before assuming it hasn't worked.

If the filter is listed, but showing a red cross, then it failed to start. In this case, check the error messages in the event log showing.

If the filter shows as green, with priority unknown, it may not be installed. In this case, you need to check. You can do this with a debug viewer, such as the one supplied by SysInternals (http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx). Download this and run it (there's no need to install it) on the server, then restart IIS. If you don't see any messages relating to the Swivel IIS filter, then it's not installed properly.

If the page is still not redirected, try the following:

1. Check which application pool your web application is running as, then go to the properties page for that application pool.

2. On the Identity tab, change the user to ?Local System?. You will be warned that this is a potential security risk, but don?t worry ? it won?t be left like this.

3. Restart IIS.

4. Try accessing a protected page again. Hopefully this time you will be redirected.

5. You can now go back to the application pool and change the identity back to what it was originally: it would appear that it is only necessary to run as an administrator to get the filter to register initially.

If you do not see a Turing image when using start session then in a web browser test the following link from the IIS server. If an image is not seen, then there is a problem either with communicating with the Swivel server or the Allow Image request by username may be set to No.

For a virtual or hardware appliance Install

https://<pinsafe_server_ip>:8443/proxy/SCImage?username=<username>

For a software only install see Software Only Installation

If the web page is redirected to the /PINsafe/login.asp page but an error message appears then ensure that the Active Server Pages are allowed. To verify this select the Internet Information Services Manager expand the required server then click on Web Service Extension.

No authentication on main page

Open IIS Manager and disable Anonymous authentication for the root folder. Refresh the browser to prevent caching and try again.

You may need to ensure that Anonymous authentication is enabled for the PINsafe folder, though, so you don't run into problems showing the TURing image.

Authentication working internally but not externally

If it is working internally, but not externally, ensure that there is no caching by openine a new browser. Also specify the default redirect URL as "/default.htm", rather than "./default.htm". The latter will redirect to default.htm within the pinsafe folder.

6.9.1 Error Messages

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

User fails to authenticate with the above error message in the Swivel log. An Agent on Swivel server has not been defined for the IIS server. Go to Server/Agents in the Swivel admin console, and add a new entry, using the IP address of the IIS server. Make sure the agent secret is the same as on the IIS filter configuration.

This installation package is not supported on this processor type. Contact your product vendor

7 PHP Integration

7.1 Introduction

This article provides example files for integrating Swivel with a PHP-based website. The solution provided here is just an example: you will need to modify it according to your needs to produce a working solution.

7.2 Prerequisites

This link provides an example PHP filter - see below for more details. This version has been updated to include PINpad support, and has had limited testing on PHP version 8.1.

This link is the previous version. It was tested using PHP 5.3, but does not include PINpad support, only TURing.

The following links are for older PHP solutions. Going forward, it is recommended that you use the first link. The following libraries will not be maintained further.

This link provides a PINsafe API library and a basic example login page.

This link points to the previous version of the library, which uses the HTTP library. As not all PHP implementations include this, and the functionality can be reproduced using just the cUrl library, this version will not be maintained in the future. It has been tested with PHP version 5.3 on a Linux server (Ubuntu), but to use it in PHP 5.3 under Windows, you will need to get hold of the appropriate libraries. Unfortunately, a previous link to the relevant library is no longer valid, so we cannot provide a suitable link at present.

7.2.1 PHP Requirements

PHP version 5.3 or later. Version 5.2 may also work, but has not been tested. Versions earlier than 5.2 are known not to be compatible. The latest version has only been tested in PHP version 8.1.

The latest solution uses the PHP modules DOM and cUrl. The earlier version also uses the HTTP module, as described above.

The following setting is required in php.ini:

```
allow_url_fopen = On
```

7.3 Example PHP Filter

The example code above shows how you might use the PHP library to protect a PHP-based website. It contains the following files:

- swivel_client.php - An enhanced version of the PHP API library.
- config.php - this file is used to read in the configuration settings.
- swivel_filter.php - this file should be included in every PHP page you want to protect.
- image.php - A TURing image proxy.
- pinpad.php - A PINpad image proxy.
- login.php - an example login page with no image support.
- loginTuring.php - an example login page with TURing support.
- loginPinpad.php - an example login page with PINpad support.
- loginPush.php - an example login page with Push support.
- logout.php - an example logout page.
- testPage.php - an example web page that uses the filter, demonstrating how it should be used.
- login.css - the stylesheet for the login page.
- swivel_push.js - JavaScript to support Push login.
- config.xml - the Swivel server settings file.

- Copy all of these files to a subdirectory on your web site, for example "/swivel".
- Edit config.xml and enter the correct settings for your Swivel server - see below for more details. You should also enter a random string for the Cookie secret - the longer and more random the better.
- Optional: move config.xml to a location outside your website. Edit swivel_client.php and change CONFIG_DOC to the full path of this file. If you leave config.xml on the website with the other files, it can be read by browsers, which might be considered a security risk.
- Edit swivel_filter.php and set $swivelPath to the relative URL of the swivel subdirectory ('/swivel' if you are using the location suggested above). Change the name of the login page as well, depending on which one you want to use.
- Edit every PHP page that you want to protect with Swivel authentication, and add the line

```
<?php require('../swivel/swivel_filter.php'); ?>
```

Note that the exact URL above depends on whereabouts in the website your file is. This example assumes your page is in a subdirectory off the root website. If it is several levels deep, you will need to prepend more '../' entries to get to the right directory.

7.3.1 Filter Configuration

The following is an example config.xml file. Each value will be described below:

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
 <server>localhost</server>
 <port>8080</port>
 <context>pinsafe</context>
 <secret>secret</secret>
 <secure>1</secure>
 <passwords>0</passwords>
```

```
 <ignoreCertErrors>1</ignoreCertErrors>
 <cookieSecret>sdf9087d2345fv89hn!</cookieSecret>
 <cookieTimeout>30</cookieTimeout>
 <caInfoFile></caInfoFile>
</config>
```

- **server** - the (internal) host name or IP address of the Swivel appliance
- **port** - the port used to communicate with the Swivel application - typically it will be 8080
- **context** - the application context for the Swivel application - this will invariably be "pinsafe"
- **secure** - 1 if you are using https to communicate with the Swivel application (the default), or 0 if it is http.
- **ignoreCertErrors** - 1 if you are using https, and the certificate on the Swivel appliance is not fully trusted, or the certificate subject does not match the **server** parameter given above. Typically, this will be required if you are using https, unless you set a trust store as described below.
- **passwords** - 1 if you want to show a password field on the login page as well as the one-time code. Normally, you only want this if you are using Swivel passwords as well as PINs, or you are using the option to check repository password.
- **secret** - the shared secret for the Swivel Agent. See here for more information.
- **cookieSecret** - the seed used to encrypt the authentication cookie. Setting this to a large, random value will improve security.
- **cookieTimeout** - the length of idle time (in minutes) before the authentication cookie expires and the user has to re-authenticate. The default is 10.
- **caInfoFile** - this is the full path to a trust store containing the CA root certificates in PEM format. PHP does not provide such a store by default, so you will need to create one: see PHP documentation on curl for details. Alternatively, specify ignoreCertErrors as 1, in which case you do not need to provide this (and it will be ignored anyway).

8 Swivel Combined Client

8.1 Overview

This document describes a combined authentication and administration client for Swivel Server APIs which can be used in web servers using ASP.Net. It is a convenient wrapper around the XML-based APIs described here.

The client can also be used in non-web applications using the .Net Framework. See configuration below.

8.2 Prerequisites

The client DLL is available from here. This is version 1.3 of the client, which includes support for user attributes and for TLS protocol versions 1.1 and 1.2. This requires version 3.9.7 or later of the Swivel server. For TLS 1.1/1.2 support, version 3 of the Swivel appliance is required.

An example website demonstrating some of the features of the client is available from here. This download includes the DLL above, so you don't need to download both. Again, this test server uses the new client.

This version of the client requires Microsoft.Net Framework version 4.5 or later.

8.3 Using the Client

8.3.1 Configuring The Swivel Server

In order to use the client with a particular Swivel Server, you must define the computer running your web application as an Agent on that server. To do this:

- Log into the Swivel Server Administration Console
- Go to Server -> Agents
- In version 3.8 or later, expand the entry at the bottom labelled "New Entry". In earlier versions, this will already be visible.
- Enter a name for the new Agent. Note that any users created using the API will be added to a repository with this name, so the name should not be the same as an existing repository if you intend to manage users through this Agent.
- Enter the IP address of the web server under Host/IP.
- If you will be managing users through this Agent, set "Can act as Repository" to Yes.
- The remaining entries can be left as default for now. Click Apply.

Depending on how you intend to use the client, you may like to create a new Repository Group for users created by the Agent, or you may prefer to use an existing Group. You can create a new Group under Repository -> Groups. All you need to add is a name in the blank entry at the bottom. There is no need to add definitions for other repositories, or to specify rights for this Group, as you will be specifying user rights explicitly when the user is created. You may also like to go back to the Agent definition and set the Group for this Agent to the Group you have just created. If you do this, only users created by the Agent will be able to authenticate through the Agent.

If you have created a new Group, you should also set up Transports for this Group. You should at least set up an Alert Transport, and if you are using dual channel authentication, a Strings Transport as well. Note that if you want to use an existing Transport, you will need to copy the class name to a new Transport and give it a new name. Select the Group you have just created as the Strings or Alert Group as appropriate.

8.3.2 Configuring Your ASP.Net Application

In order to use the client, you need to add certain entries to the web.config file in your ASP.Net application. The following is an example:

```
<appSettings>

  <add key="PINsafeServer" value="myserver"/>
  <add key="PINsafePort" value="8080"/>
  <add key="PINsafeContext" value="sentry"/>
  <add key="PINsafeSecret" value="secret"/>
  <add key="PINsafeSecure" value="True"/>
  <add key="PINsafeAcceptSelfSigned" value="False"/>
  <add key="AllowNonPINsafeUsers" value="False"/>
  <add key="IgnoreDomainPrefix" value="True"/>
  <add key="IgnoreDomainSuffix" value="False"/>
  <add key="PINsafeAgentVersion" value="3.97" />
  <add key="PINsafeTlsProtocols" value="Tls12" />

</appSettings>
```

The comments at the start and end are not necessary - they are for clarity - but there is a method in the client to remove these settings which requires these comments. The <appSettings> element should be contained within the main <configuration> element, and may already exist, depending on your application.

If you are using the client in an executable (i.e. non-web) application, you can use exactly the same settings, but these must be in a file named *Application*.exe.config, where *Application* is the name of the executable application. This must be in the same directory as the program executable.

The meanings of the key names are shown below:

- **PINsafeServer** - The host name or IP address of the Swivel server
- **PINsafePort** - The port used by the Swivel server, normally 8080
- **PINsafeContext** - The context (application URL) of the Swivel server, normally "pinsafe"
- **PINsafeSecret** - The shared secret as configured in the Agent entry previously
- **PINsafeSecure** - "True" if HTTPS is to be used for communication with, "False" otherwise
- **PINsafeAcceptSelfSigned** - "True" if SSL certificate errors should be ignored
- **AllowNonPINsafeUsers** - "True" if users unknown to PINsafe should be authenticated automatically, "False" if they should be rejected
- **IgnoreDomainPrefix** - "True" if the domain prefix in usernames such as domain\user should be stripped off before checking with Swivel server
- **IgnoreDomainSuffix** - "True" if the domain suffix in usernames such as user@domain should be stripped off before checking with Swivel server
- **PINsafeAgentVersion** - Sets the *Version* attribute sent with the API request. The default is "3.4". To support user attributes, this should be set to "3.97".

- **PINsafeTlsProtocols** - Specifies which TLS protocols are supported. You can specify Ssl3, Tls1, Tls11 or Tls12. Separate supported protocols with commas. The default is "Tls1,Tls11,Tls12".

NOTE: on an appliance, you should not use the proxy application (and port 8443), as functionality in the proxy is deliberately limited. You must use the pinsafe application to get the full functionality out of the client.

If it is not practical to provide the settings in a configuration file, you can create an instance of the SwivelSettings class, and initialise it as follows:

```
SwivelSettings swivelSettings = SwivelSettings.EmptySettings;
swivelSettings.Server = "myserver";
swivelSettings.Port = 8080;
swivelSettings.Context = "sentry";
swivelSettings.Ssl = true;
swivelSettings.Secret = "secret";
swivelSettings.AcceptSelfSigned = false;
swivelSettings.TlsProtocols = SecurityProtocolType.Tls12;
```

You can then use this instance when creating a request - see the class documentation for details.

8.3.3 Implementing the Swivel Client in ASP.Net Applications

In order to use the client DLL, you need to copy it to the Bin folder of your ASP.Net application.

Technical documentation for the client is in progress. Please see the example application.

The namespace for all classes in the client is **swivelsecure.client**.

Documentation of the authentication class can be found here

The user management classes fall into two categories: administrator and helpdesk. The administrator methods are more extensive, allowing you to create, update and delete users. However, they are limited to users belonging to a single repository, named after the Agent corresponding to the computer the application is running on. The helpdesk methods are more limited, typically read-only methods, but with some update functionality. However, these methods can apply to users in other repositories. See the two documents for details.

9 Website Integration

10 Website Authentication

There are a number of ways of authenticating custom-built websites and web based applications, depending on the nature of the application and how much development you can do or are prepared to do.

- A XML-based API for authentication is provided, see Agent-XML.
- For ASP.Net Swivel has a DLL wrapper around this API. See, Swivel_Combined_Client.
- For IIS, see: Microsoft_IIS_version_7_Integration.
- For IIS forms based authentication see: Microsoft_IIS_version_7_ASP.NET_Forms_Integration.
- For a PHP wrappers for the API: see PHP_Integration.
- For a Java wrappers for the API: see Java_Client.