AdminAPI

Contents

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.

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.

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.

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.

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.

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.

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.

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>
```

Delete

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

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

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.

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>
```

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>
```

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>
```

Purge

Purge deleted users from the repository:

```
<PurgeDeleted />
```

No content is permitted.

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>
```

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.

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>
```

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>
```

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>
```

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>
```

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.

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>
```

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>
```

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.
```

Error Handling

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.

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.

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

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".

.