McAfee 9.4 ESM API Guide

Contents

Summary

The following document describes the process involved in using both the SOAP/XML based API as well as its corresponding RESTful form. All calls are available using both methods. It is assumed that the reader is familiar with XML based web services including WSDL (Web Services Descriptor Language) files, and/or RESTful APIs. Only a basic understanding of Java is necessary as the example code is minimal and not complicated.

Currently there are three parts to the API:

- API Coverage area list, providing a general idea of the things that can be done with the API
- A querying tutorial that provides detail around making calls to the API
- Current planned API call list with descriptions

This document should be considered an early rough draft, meant to illustrate how the API will work and what type of functionality will be exposed in the final version.

API Coverage Areas

The following is a breakdown of planned functionality that will be covered in the API. Each bullet point will not necessarily map to a single API call, but rather provides a general indication of the intent of the calls. This list is likely to change during development:

- Queries
  - Get all queries
  - Customize a query's fields, filters, time range, ordering, grouping
  - Execute the query
  - Poll until it's finished, with progress information
  - Get the paged results for the query
- Watchlists
  - Add watchlist
  - Remove watchlist
  - Edit watchlist properties
  - Add/Remove/Update watchlist values – similar to external call today but done through a call as opposed to a file import
- Users
  - Add/remove/edit users
  - Add/remove/edit user access groups
  - Get user list
- Alarms
  - List triggered alarms
- Data sources
  - Add/remove/update data source properties
  - List data sources

- Get extended detail on one data source

gSOAP ESM client setup in Visual Studio 2005

Use gSOAP (http://www.cs.fsu.edu/~engelen/soap.html)
The gSOAP toolkit is a C and C++ software development toolkit for SOAP/XML Web services and generic (non-SOAP) C/C++ XML data bindings. The toolkit analyzes WSDLs and XML schemas (separately or as a combined set) and maps the XML schema types and the SOAP messaging protocols to easy-to-use and efficient C and C++ code. It also supports exposing (legacy) C and C++ applications as SOAP/XML Web services by auto-generating XML serialization code and WSDL specifications.
Getting gSOAP
Download gSOAP from http://sourceforge.net/projects/gsoap2/files/.
For this turtorial use version *gsoap_2.8.17*. This file contains the precompiled Windows binaries and the required source. Unzip the archive to hard drive. Assuming you're running Windows, once uncompressed, gSOAP is ready to run. The precompiled binaries should work with no need to (re-)compile gSOAP. Required tools will be available under directory *<unzipped path>\gsoap-2.8\gsoap\bin*
Compiling WSDL

1) Execute following command:
    a. *wsdl2h.exe -f -k -o esm.h https://<ipaddress_of_your_ESM>/ws/esm?wsdl*

   This command may fail complaining that *"Cannot connect to https site: no SSL support, please rebuild wsdl2h with SSL or download the files and rerun wsdl2h"*
   To solve this either rebuild the gSOAP source with SSL support or see step 2 for a work around.
2) Open the wsdl (https://<ipaddress_of_your_ESM>/ws/esm?wsdl) in a browser and save the file locally. Then run the below command (where esm.htm is the wsdl file you saved locally).
    a. *wsdl2h.exe -f -k -o esm.h esm.htm*

   [It may be required that you provide option "-Igsoap\import" for import files]

   When completed, esm.h is the output header file.

Generating C++ Stubs

1. Run below command to generate the C++ client and server stub file:
    a. *soapcpp2.exe -j -C -Igsoap\import esm.h*

Now we have all the required source files to call the ESM SOAP API in C++. But we need an ssl library for secure communication with ESM. gSOAP depends upon Open SSL for SSL-secured communications.
Get OpenSSL

1. Download the source code from https://www.openssl.org/source/openssl-1.0.1g.tar.gz
2. Unzip the files (e.g. in c:\openssl).

3. Open INSTALL.w32 for windows 32 bit or install.w64 for windows 64 bit for the installation instruction.
4. Follow the instructions for Visual C++. Perl is a prerequisite.
   a. perl Configure VC-WIN32 --prefix=c:\some\openssl\dir
   b. perl Configure VC-WIN32 no-asm --prefix=c:/some/openssl/dir
   c. ms\do_ms
   d. Then from the VC++ environment at a prompt do:
      i. nmake -f ms\ntdll.mak
   e. Run the test to make sure all is well
      i. nmake -f ms\ntdll.mak test
   f. At this point all the output files are generated into *openssl-1.0.1g\out32dll*
   g. To install OpenSSL to the specified location do:
      i. nmake -f ms\ntdll.mak install

Setup Visual Studio Project

1. Create a C++ Win32 Console Application project. Accept all of the default settings.
2. Add the generated files into the project
   a. *McAfeeEsmApiServiceSoapBinding.nsmap*
   b. *soapC.cpp*
   c. *soapMcAfeeEsmApiServiceSoapBindingProxy.h*
   d. soapMcAfeeEsmApiServiceSoapBindingProxy.cpp
   e. *soapH.h*
   f. *soapStub.h*
3. Add the following files from the gSOAP directory to the project:
   a. stdsoap2.cpp
   b. stdsoap2.h
4. *For soapC.cpp, soapMcAfeeEsmApiServiceSoapBindingProxy.cpp and stdsoap2.cpp files change the precompile option to "Not Using Precompiled Headers".*
5. Right click on the project and select Properties from the menu. Under Configuration Properties, C/C++ and General, add include paths to the gSOAP and OpenSSL include directories.
6. Under Configuration Properties, C/C++ and Preprocessor, add to the list of Preprocessor Definitions, "WITH_OPENSSL" (to enable SSL in gSOAP) and if you choose, "DEBUG" (to enable logging).
7. Under Configuration Properties, Linker and General, add to Additional Library Directories the path to the OpenSSL binary directory, where the compiled lib files reside.
8. Under Configuration Properties, Linker and Input, add to Additional Dependencies, "libeay32.lib" and "ssleay32.lib". These are the Open SSL libraries.
9. Open the console application C++ file. Add three `#include`s to the top of the file just after the *stdafx.h* include:

   a. `#include "soapMcAfeeEsmApiServiceSoapBindingProxy.h"`
   b. `#include "McAfeeEsmApiServiceSoapBinding.nsmap"`
   c. `#include "stdsoap2.h"`

10. Now the project should build fine.

SOAP OpenSSL Configuration

1. Get the public certificate of the ESM server.
   a. echo "" | openssl s_client -connect <*ipaddress_of_your_ESM*>:443 -showcerts 2>/dev/null | openssl x509 -out certfile
2. Run the below command to view the details of the certificate:
   a. openssl x509 -in certfile –noout -text
   b. sample output:

```
openssl x509 -in certfile -noout -text
Certificate:
  Data:
    Version: 1 (0x0)
    Serial Number:
      e9:49:be:a2:e9:95:ca:de
  Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=US, ST=TX, L=Plano, O=McAfee, OU=Enterprise Security Manager,
CN=esm.mcafee.local/emailAddress=support@nitrosecurity.com
    Validity
      Not Before: Apr  9 09:06:16 2014 GMT
      Not After : Apr  8 09:06:16 2024 GMT
    Subject: C=US, ST=TX, L=Plano, O=McAfee, OU=Enterprise Security Manager,
CN=esm.mcafee.local/emailAddress=support@nitrosecurity.com
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
        Public-Key: (2048 bit)
        Modulus:
          00:c8:2e:24:60:ff:c9:be:50:f4:3d:96:11:14:ab:
          f5:d0:33:f6:e2:53:d3:92:4b:5b:1b:b9:27:ab:84:
          42:9a:cf:f1:67:e6:42:09:0d:4c:82:44:75:9d:0c:
          a4:e5:06:3f:f5:24:5a:7d:8a:24:55:03:99:bd:e3:
          c6:ec:0e:8f:26:00:86:82:29:c2:a4:10:d6:95:06:
          10:c5:ce:d6:fe:c8:98:05:a3:5c:48:36:60:3d:9b:
          ad:7c:58:01:57:f9:ed:9b:31:87:67:f0:c7:57:2c:
          d7:1b:3b:2f:51:d5:ff:40:ac:f2:1f:0a:30:be:fc:
          02:9b:df:51:37:38:be:80:32:d2:c3:a2:46:4e:d1:
```

3. Open *soapMcAfeeEsmApiServiceSoapBindingProxy.cpp.* Just after the `soap_new()` call *(approximately line 15)*, add the following code:

   ```
   soap_ssl_init();
   if (soap_ssl_client_context(soap, SOAP_SSL_DEFAULT, NULL, NULL,
        "C:\\Path\\To\\Certs\\File\\certfile ", NULL, NULL ))
   {
         soap_print_fault(soap, stderr);
   }
   ```

4. Disable host authentication (if you are using the default self-signed certificate) by commenting out the below lines in stdsoap2.cpp

   ```
   if (!ok)
   {
      soap_set_sender_error(soap, "SSL/TLS error", "SSL/TLS certificate
                 host name mismatch in tcp_connect()", SOAP_SSL_ERROR);
      soap->fclosesocket(soap, sk);
      return SOAP_INVALID_SOCKET;
   }
   ```

Finally Login to ESM server

```
McAfeeEsmApiServiceSoapBindingProxy esmProxy;

ns1__userLogin req;
ns1__userLoginResponse resp;

std::string sUsername("username");
req.username = &sUsername;

std::string sPassword("password");
req.password = &sPassword;

int err = esmProxy.userLogin( &req, &resp );

if( err == SOAP_OK )
{
      std::cout << "Login success" << std::endl;
}
```

Maintain session after login
To maintain session, we need to enable cookies by adding a complie option; -DWITH_COOKIES. In VS, go to Project Properties > Configuration Properties > C/C++ > Preprocessor. Add "WITH_COOKIES" in preprocessor definitions.

Setting up the SOAP client Test Suite for C# in Visual Studio 2012

1. Create a Visual C# project.
2. Right click on the solution explorer and **add Service reference** and provide the **WSDL** URL. If you are using the self-signed certificate, Visual Studio will raise a certificate warning. Just except the warning, provide a namespace for the web service client stubs.
3. Go to advanced option and select your preference for making changes to the generated client like choosing Array or List for collection types, generating asynchronous methods etc.
4. Click the **OK** button which will generate the SOAP clients and place it inside **Service References** directory.
5. In **App.config** file add '**allowCookies="true"**' to all the bindings. This will allow you to maintain the login state once logged in to ESM. For example:

```
<binding name="McAfeeEsmApiServiceSoapBinding" allowCookies="true">
        ..........
<binding name="McAfeeEsmApiServiceSoapBinding1" allowCookies="true">
```

6. The C# client generator has issues with how it handles the 'rows' property in the **esmQueryResult** class. It creates this property with a type '**string [][]**' instead of '**esmQueryRow[]**'. Look for this issue in the '**Reference.cs**' file (the generated SOAP client file) and if it exists, then change the **row** property from a type of **string[][]** to **esmQueryRow[]** in **esmQueryResult** generated class. You will also need to change the type on the property accessor. The final **esmQueryResult** class should look something like the following:

```
  /// <remarks/>
   [System.CodeDom.Compiler.GeneratedCodeAttribute("System.Xml", "4.0.30319.33440")]
   [System.SerializableAttribute()]
   [System.Diagnostics.DebuggerStepThroughAttribute()]
   [System.ComponentModel.DesignerCategoryAttribute("code")]
   [System.Xml.Serialization.XmlTypeAttribute(Namespace="http://mcafee.com/siem/api/v1")]
   public partial class esmQueryResults : object,
System.ComponentModel.INotifyPropertyChanged {

    private esmQueryColumn[] columnsField;

    private esmQueryRow[] rowsField;

    /// <remarks/>
    [System.Xml.Serialization.XmlElementAttribute("columns", IsNullable=true, Order=0)]
    public esmQueryColumn[] columns {
      get {
        return this.columnsField;
      }
```

```
            set {
                this.columnsField = value;
                this.RaisePropertyChanged("columns");
            }
        }

        /// <remarks/>
        [System.Xml.Serialization.XmlElementAttribute("rows", IsNullable = true, Order = 1)]
        public esmQueryRow[] rows
        {
            get
            {
                return this.rowsField;
            }
            set
            {
                this.rowsField = value;
                this.RaisePropertyChanged("rows");
            }
        }

        public event System.ComponentModel.PropertyChangedEventHandler PropertyChanged;

        protected void RaisePropertyChanged(string propertyName) {
            System.ComponentModel.PropertyChangedEventHandler propertyChanged =
    this.PropertyChanged;
            if ((propertyChanged != null)) {
                propertyChanged(this, new
    System.ComponentModel.PropertyChangedEventArgs(propertyName));
            }
        }
    }
```

Then add a new class **esmQueryRow** to **Reference.cs** as follows:

```
     /// <remarks/>
    [System.CodeDom.Compiler.GeneratedCodeAttribute("System.Xml", "4.0.30319.33440")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlTypeAttribute(Namespace = "http://mcafee.com/siem/api/v1")]
    public partial class esmQueryRow : object, System.ComponentModel.INotifyPropertyChanged
    {

        private string[] valuesField;

        /// <remarks/>
        [System.Xml.Serialization.XmlElementAttribute(Order = 0)]
        public string[] values
        {
            get
            {
                return this.valuesField;
            }
            set
            {
                this.valuesField = value;
                this.RaisePropertyChanged("values");
            }
        }

        public event System.ComponentModel.PropertyChangedEventHandler PropertyChanged;

        protected void RaisePropertyChanged(string propertyName)
        {
            System.ComponentModel.PropertyChangedEventHandler propertyChanged =
this.PropertyChanged;
            if ((propertyChanged != null))
            {
                propertyChanged(this, new
System.ComponentModel.PropertyChangedEventArgs(propertyName));
            }
        }
    }
```

7. If you face **hostname verification** issue then add and call the following method before calling the login service. This will bypass the hostname verification by trusting all hosts.

```
      private static void BypassCertificateError()
        {
            ServicePointManager.ServerCertificateValidationCallback +=
                delegate(Object sender1, X509Certificate certificate, X509Chain chain, SslPolicyErrors
sslPolicyErrors)
                {
                    return true;
                };
        }
```

8. For some of the properties there will be a **'<property>Specified'** bool field in the generated client classes. For example in **esmQueryConfig** class, for property '**timeRange**' there is a property called '**timeRangeSpeified**' of type bool. If timeRange is provided then this property has to be set to true otherwise the 'timeRange' value provided will be ignored in the service call.
9. If you want to point to a different server in the future, change the address of the ESM server in the app.config.

10. Increase the maxReceivedMessageSize to a higher value (i.e. 262144) in the App.config file. By default, C# Service reference accepts http response of size up to 65536 bytes and many ESM service responses easily exceed this size limit. For e.g.

```
<configuration>
  <startup>
    ...
  </startup>
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="McAfeeEsmApiServiceSoapBinding" allowCookies="true"
maxBufferSize="262144" maxBufferPoolSize="262144" maxReceivedMessageSize="262144">

                ...
        </binding>

      </basicHttpBinding>
    </bindings>
    <client>
                ...
    </client>
  </system.serviceModel>
</configuration>
```

Querying Tutorial

The following is a two part step-by-step tutorial showing how to issue queries through the 9.4 SIEM API. Both parts use Java as the example language, but of course Web Services can be used with just about any language. Part 1 will detail the steps using the WSDL for SOAP/XML based interactions. The second part shows how to do the same this with the REST interface.

Documentation on all currently exposed calls is available at the following url:

https://<esm>/rs/esm/help/commands

A list of commands is presented and information describing enum values, complex types, etc can be viewed from this basic interface. It is recommended that this online interface be used for reference during development of an API client.

SOAP/XML Querying

The first step in connecting via SOAP/XML would be to build a client from the WSDL file. This file can be located at the following URL (*NOTE: this url will change as we get integrate with Apache on the ESM*):

https://<esm>/ws/esm?wsdl

As an example command that pulls in the WSDL and generates client source, an ant target using wsimport follows:

```
<wsimport
destdir="${build.dir}/client"
sourcedestdir="${build.dir}/client"
keep="true"
extension="true"
verbose="false"
wsdl="${com.mcafee.siem.api.base.url}/ws/esm?wsdl"
package="com.mcafee.siem.api.v1">
<xjcarg value="-cp"/>
<xjcarg file="${basedir}/lib/jaxb2-value-constructor.jar" />
<xjcarg value="-Xvalue-constructor"/>
<binding dir="${basedir}/webapp" includes="bindings.xml"/>
</wsimport>
```

The remainder of this tutorial will work off of the assumption that the client code has been successfully generated from the WSDL and is ready to be used.

The SOAP based interface is stateful, as would be expected. This means that when connecting to the API, session tracking needs to be enabled so that the login state can be maintained for the duration of the session. In Java, this is done as follows:

```
EsmApi api = new McAfeeEsmApiService().getMcAfeeEsmApiPort();
((BindingProvider)api).getRequestContext().put(BindingProvider.SESSION
_MAINTAIN_PROPERTY, true);
```

Once the api is set up to maintain state, we can login with our credentials:

```
api.userLogin(username, password);
```

If any problems occur during processing of a call for any reason including invalid login credentials, an EsmException will be thrown with a description of the reason why the call failed. Once the login succeeds, other calls can be made to the API.

For this querying tutorial, the first thing we'll do is retrieve the query templates from the system:

```
List<EsmQueryTemplate> queries =
api.qryGetQueryList(EsmQueryOrigin.PREDEFINED, EsmQueryType.BOTH,
true, Collections.singletonList(EsmQueryUse.VIEWS));
```

They are returned in a list. These templates are named so the client can easily locate the query template that is needed. Once found, it would be possible to retain this query's EsmQueryID object to avoid having to make this call in the future, assuming the query template wasn't deleted in the interim. These kinds of details are up to the client developer.

```
EsmQueryTemplate query = (some query from the above list);
```

Assuming the template that is needed is located, its id property can be used to reference the query in future calls. Before executing the query template, however, we may want to customize the query's columns, time range, filters, etc. This is done via the EsmQueryConfig type. An EsmQueryConfig object must be provided as part of the call to execute the query. If no modifications are needed, the default config object can be obtained as follows:

```
EsmQueryConfig info = api.qryGetDefaultConfig(query.getId());
```

If custom fields are needed from the query, all available select fields for the given query type can be obtained with the following call:

```
List<EsmSelectField> selectFields =
api.qryGetSelectFields(query.getType());
```

The following snippet shows some modifications being made to the query config object:

```
config.getFields().add(importantSelectField); //add some other field
you want returned in the results
config.setTimeRange(EsmTimeRange.CUSTOM); //set the time range of the
query to some custom time period
config.setCustomStart(DatatypeFactory.newInstance().newXMLGregorianCal
endar(new GregorianCalendar(2013, 9, 5))); //from oct 5th
config.setCustomEnd(DatatypeFactory.newInstance().newXMLGregorianCalen
dar(new GregorianCalendar(2013, 9, 25)));  //to oct 25th
```

Once the config is setup as needed, the query template can be run using this config:

```
EsmRunningQuery runningQuery = api.qryExecute(query.getId(), info,
false);
```

Note the return value for the execute call is an EsmRunningQuery. This object will provide the result ID needed to reference the running query when looking for progress and/or result data. Before the results can be retrieved, the progress must reach 100%. This is checked by asking the ESM for the query's status like so:

```
EsmQueryStatus status = api.qryGetStatus(runningQuery.getResultID());
```

The "percentComplete" property will provide a progress value of between 0 and 100. To simply check whether the query is done running, the complete flag can be checked:

```
while (!status.isCompleted()) {
    //do something
    ...
    //check again
    status = api.qryGetStatus(runningQuery.getResultID());
}
```

When the query has completed, its results can now be retrieved. The status object contains the total row count, so a paged approach to reading the results can be achieved using qryGetResults call:

```
long pos = 0;
while (pos < status.getTotalRecords()) {

    //get the next batch of rows
    EsmQueryResults results =
api.qryGetResults(runningQuery.getResultID(), pos, PAGE_SIZE, false);

    //update our row counter
    pos += results.getRows().size();

    //do something with the results

    ...

}
```

Note that in the above example code, PAGE_SIZE is just a statically defined value on the client side. This value should not be higher than 10000. Once the query results have been processed, the query should be closed:

```
api.qryCloseResults(runningQuery.getResultID());
```

At this point a query template has been retrieved, customized, executed, and the results retrieved from the server. This concludes the SOAP/XML querying tutorial.

REST Querying

The same API calls are available over a RESTful interface, but some of the mechanics around login and logout are different. In a pure RESTful environment there would be no concept of a client session. The SIEM API requires state to be maintained between requests in cases like long running queries that must be monitored and processed with subsequent calls. For this reason, a login POST call will create a session, and this created session ID will be returned in the Location response header. Subsequent calls need to set the authorization header to use the custom ESM Session scheme: Session <sessionId> Where <sessionId> is the plain text session provided in the Location response header after a successful login.

The following is a description of the HTTP calls for logging in and performing all of the above actions using the REST interface.

- To login, set the HTTP authentication header to Basic with encoded username:password, which is pretty standard. The URL for the POST would be: http://localhost:8080/rs/esm/login
- Read the Location response header from this call to get the created session, which will look something like this, in plain text: 1234567890
- Subsequent calls are then made with the “authorization” header set to “Session 1234567890”

To make a call, you can use HTTP GET or POST, but generally GET would be used for calls that aren't sending complex JSON. The content of a POST of complex content would consist of simple query parameters and complex data. For simple objects like IDs or primitives, the values can be provided in the query string or as part of the content of the request. Complex objects must always be posted as json in the request body. RESTful examples of every call are provided in the documentation mentioned above. Here is what the URL and JSON content might look like for a qryExecute call:

The RESTful URL (with request header authorization: Session <sessionId>):

```
http://localhost:8080/rs/esm/qryExecute?id=123&reverse=false
```

The JSON content:

```
{"config": {
    "timeRange": "CUSTOM",
    "customStart": "2013-09-08T15:55:43.557-04:00",
    "customEnd": "2013-10-08T15:55:43.558-04:00",
    "includeTotal": false,
    "fields": [
        {
            "name": "DstPort",
            "table": "Alert"
        },
        {
            "name": "DSID",
            "table": "Alert"
        },
        {
            "name": "EventCount",
            "table": "Alert"
        }
    ],
    "limit": 0
}
}
```

In an attempt to remain as RESTful as possible, a DELETE request is then issued to log the user out effectively deleting the session, again assuming the authorization header has been set appropriately: https://<esm>/rs/esm/logout

Using this information, the explanations of the calls in the SOAP/XML tutorial, and the examples in the online documentation it should be possible to write a REST based API client.

Call List

What follows is a current list of planned calls and a short description of each. Please note that this list is subject to change, as the API is still in its early development stages:

`caseAddCase`

Add a case to the system.

`caseEditCase`

Edit an existing case.

`caseGetCaseDetail`

Get detail on an existing case.

`caseGetCaseUsers`

Get users associated with a particular case.

`devGetDeviceList`

Get a list of all devices defined in the system.

`dsAddDataSource`

Add a data source.

`dsDeleteDataSources`

Delete a data source.

`dsEditDataSource`

Edit a data source's properties.

`dsGetDataSourceList`

Get a list of defined data sources.

dsGetDataSourceTypes

Get all data source types.

dsGetUserDefinedDataSources

Get user defined data sources.

dsSetUserDefinedDataSources

Set user defined data sources.

geoGetGeoLocRegionList

Get the top level geo locations

geoGetGeoLocs

Get geo locations within the given location

getVersion

Get the version information for this ESM

grpGetDeviceTree

Gets the basic device tree structure with only basic properties loaded. Each entry in the returned list is a root node in the tree.

grpGetDeviceTreeEx

Gets the basic device tree structure with only basic properties loaded. Each entry in the returned list is a root node in the tree. This version of the call returns more detail per device than getDeviceList, wrapped in an esmDeviceList object.

alarmGetTriggeredAlarms

Get triggered alarms

plcyGetPolicyList

Get the list of all policies defined in the ESM.

plcyGetVariableList

Get all variables defined in the system

qryCloseResults

Closes the query results, must be called after a query's results have been processed. If no exception is thrown, the close operation completed normally.

qryExecute

Execute a query.

qryGetColumns

Get detail on query column indexes for a particular query.

qryGetDefaultConfig

Get extended information on a query template.

qryGetFilterFields

Get all fields that can be used in query filters.

qryGetQueryList

Gets a list of all queries, based on the constraints passed in.

qryGetResults

Get the results for a query.

qryGetSelectFields

Get the fields available for selecting in queries

qryGetStatus

Get the status for a query that has been executed.

sysAddWatchList

Add a watchlist to the system.

sysEditWatchList

Edit properties of a watchlist.

sysGetWatchlistDetails

Get detailed information about a watchlist.

sysGetWatchlists

Return basic information on all watchlists in the system

sysRemoveWatchList

Remove a watchlist from the system.

sysUpdateWatchlistData

Update data for a watchlist (add/remove values)

userAddAccessGroup

Add an access group

userAddUser

Add a user to the system.

userDeleteAccessGroup

Delete an access group.

userDeleteUser

Delete a user from the system.

userEditAccessGroup

Edit properties of an access group.

userEditUser

Used by the master user to update information about another user.

userGetAccessGroupInfo

Get extended information about an access group.

userGetAccessGroupList

Get all user access groups defined in the system.

userGetRightsList

Get all rights defined in the system.

userGetUserList

Get a list of all users.

userLogin

Log into the SIEM with the given username and password.

userLogout

Log the user out of their SIEM session

userModifyUser

Allows a user to modify basic information about them including their password.

zoneAddSubZone

Add a new subzone under a zone

zoneAddZone

Create a new zone.

zoneDeleteSubZone

Delete the sub zone

zoneDeleteZone

Delete the zone

zoneEditSubZone

Edit the given sub zone. Note that ID must be set to an existing sub zone for this to work properly. The ID value will be set if the zone was gotten from zoneGetSubZone().

zoneEditZone

Edit the given zone. Note that ID must be set to an existing zone for this to work properly. The ID value will be set if the zone was gotten from zoneGetZone().

zoneGetSubZone

Get detailed information on a sub zone

zoneGetZone

Get extended detail on a zone.

zoneGetZoneTree

Get the full tree of zones defined in the ESM.