SOPHOS

simple + secure

SafeGuard Enterprise
Management API

Product version: 5.60
Document date: April 2011

Content

1 SafeGuard Enterprise Management API 2

2 Detailed API Description 14

3 Installation/environment 84

4 Technical Support 85

5 Legal notices 86

1 SafeGuard Enterprise Management API

1.1 Short description

The SafeGuard Enterprise Scripting API contains methods supporting the following areas:

- Users & Computers management
- User-computer assignment (UMA)
- Key generation and assignment
- Certificate assignment
- Token management
- Inventory and status information
- Challenge/Response
- Reporting
- Service Accounts
- Misc

Prior to using the Scripting API Security Officer authentication is mandatory. Security Officer authentication is offered by the Scripting API. However, additional authentication (OTS) is not possible. The authenticated officer must therefore be allowed to perform all required actions on their own. Authentication can be done once centrally and will be valid for the whole scripting session.

Events will be logged the same way as if the user was logged on interactively using the Management Center. Additional events have been defined to reflect the usage of the functionality by the Scripting API. The Scripting API is exposed through COM registration and can be used by common scripting languages like VBS.

1.2 Base

The following methods are available:

```
Initialize()

FreeResources()

CreateDirectoryClassInstance()

CreateUMAClassInstance()

CreateKeysClassInstance()

CreateCertificatesClassInstance()

CreateTokenClassInstance()

CreateInventoryClassInstance()

CreateCRClassInstance()

CreateReportsClassInstance()

CreateMiscClassInstance()

CreateServiceAccountsClassInstance()

GetLastError()

AuthenticateOfficer(string officerName, string pinOrPassword, string
confFilePathName)

AuthenticateWHDOfficer(string OfficerName, string Password)

AuthenticateService()

HasOfficerRightForAction(int action, out int right)
```

1.3 Users and Computers management

The following methods are available:

```
CreateDirectoryConnection(string dsn, string userName, string
password, string serverNameIP, string port, int SSL)

DeleteDirectoryConnection(string dsn)

SynchronizeImportedContainers(string logFileName, int membership, int
accountState)

GetOneObject(string searchName, string adsStartObject, int filter, out
string adsObject, out string type)

GetObjectInitialize(string searchName, string adsStartObject, int
filter, out int hitCount)

GetObjectByIndex(int index, out string adsObject, out string type)

GetObjectFinalize()

GetMemberOfGroupInitialize(string adsGroup, out int hitCount)

GetMemberOfGroupByIndex(int index, out string adsMember, out string
type)

GetMemberOfGroupFinalize()


CreateUser(string userLogonName, string userFullName, string
adsContainer, out string adsUser)

RenameUser(string adsUser, string newUserFullName, out string
newAdsUser)

DeleteUser(string adsUser)

MoveUser(string adsUser, string adsToContainer, out string newAdsUser)

AddUserToGroup(string adsUser, string adsToGroup)

RemoveUserFromGroup(string adsUser, string adsFromGroup)

GetUserProperty(string adsUser, string property, out string value)

SetUserProperty(string adsUser, string property, string value)


CreateMachine(string machineName, string adsContainer, out string
adsMachine)

RenameMachine(string adsMachine, string newMachineName, out string
newAdsMachine)
```

```
DeleteMachine(string adsMachine)

MoveMachine(string adsMachine, string adsToContainer, out string
newAdsMachine)

AddMachineToGroup(string adsMachine, string adsToGroup)

RemoveMachineFromGroup(string adsMachine, string adsFromGroup)

GetMachineProperty(string adsMachine, string property, out string
value)

SetMachineProperty(string adsMachine, string property, string value)


CreateOU(string ouName, string adsParentContainer, out string
newAdsOU)

CreateContainer(string containerName, string adsParentContainer, out
string newAdsContainer)

CreateDomain(string domainName, string distinguishedName, string
domainNetbios)

CreateWorkgroup(string workgroupName, out string newAdsWorkgroup)

RenameContainer(string adsContainer, string newContainerName, out
string newAdsContainer)

DeleteContainer(string adsContainer)

MoveContainer(string adsContainer, string adsToContainer, out string
newAdsContainer)


CreateGroup(string groupName, string adsContainer, out string
adsGroup)

RenameGroup(string adsGroup, string newGroupName, out string
newAdsGroup)

DeleteGroup(string adsGroup)

MoveGroup(string adsGroup, string adsToContainer, out string
newAdsGroup)

AddGroupToGroup(string adsGroup, string adsToGroup)

RemoveGroupFromGroup(string adsGroup, string adsFromGroup)


GetSGDProperty(string adsPath, string property, out string value)

SetSGDProperty(string adsPath, string property, string value)
```

```
SetObjectToAD(string adsPath)

SetObjectToSG(string adsPath)

ConvertADGuidToSGNGuid(object adGuid, out string sgnGuid)

CanSynchronizeDirectory(out int canSynchronize)

CanSynchronizeDirectory(out int canSynchronize)

ResetSynchronizationLock()
```

1.4 User-computer assignment

The following methods are available:

```
CreateUMA(string adsUser, string adsMachine)

DeleteUMA(string adsUser, string adsMachine)

SetUMAProperty(string adsUser, string adsMachine, string property,
string value)

GetUMAProperty(string adsUser, string adsMachine, string property, out
string value)

GetUMAOfUserInitialize(string adsUser, out int hitCount)

GetUMAOfUserByIndex(int index, out string adsMachine)

GetUMAOfUserFinalize()

GetUMAOfMachineInitialize(string adsMachine, out int hitCount)

GetUMAOfMachineByIndex(int index, out string adsUser)

GetUMAOfMachineFinalize()
```

1.5 Key generation and assignment

The following methods are available:

```
CreateKey (string adsObject, string desiredName, string binaryValue,
out string symbolic name, out string keyId)

GetKeyBySymbolicNameInitialize(string symbolicName, out int hitCount)

GetKeyBySymbolicNameByIndex(int index, out string keyId)

GetKeyBySymbolicNameFinalize()

GetKeyByIdInitialize(string keyId, out int hitCount)

GetKeyByIdByIndex(int index, out string keyId, out string keyName)
```

```
GetKeyByIdFinalize()

GetKeyProperty (string keyId, string property, out string value)

SetKeyProperty (string keyId, string property string value)

AssignKey (string adsObject, string keyId)

DetachKey (string adsObject, string keyId)

GetAssignedKeyInitialize(string adsObject, out int hitCount)

GetAssignedKeyByIndex(string int index, out string keyId)

GetAssignedKeyFinalize()

GetAssignedObjectInitialize(string keyId, int out int hitCount)

GetAssignedObjectByIndex(int index, out string adsObject)

GetAssignedObjectFinalize()
```

1.6 Certificate assignment

The following methods are available:

```
ImportAndAssignCertToUser(string adsUser, string pathP12, string
pathP7)

CreateAndAssignCertForUser(string adsUser, string password)

GetCertOfUserInitialize(string adsUser, out int hitCount)

GetCertOfUserByIndex(int index, out string certId)

GetCertOfUserFinalize()

GetUserOfCertInitialize(string subject, string issuer, string serial,
out int hitCount)

GetUserOfCertByIndex(int index, out string adsUser)

GetUserOfCertFinalize()

GetOneCertificate(string subject, string issuer, string serial, out
string certId)

DetachCertFromUser(string adsUser, string certId)

GetCertInfo(string certId, out string subject, out string issuer, out
string serial, out string startDate, out string endDate)

DeleteCertFromDB(string certId)

RenewCertificate(string certId, string password)

ImportCRL(string pathCRL)
```

```
ImportCACert(string pathCACert)

DeleteCRL(string crlFileName)

DeleteAllCRL()

DeleteCACert(string subject, string issuer, string serial)

DeleteAllCACerts()

Int32 Certificates::ImportAndAssignStandbyCertToUser

Int32 Certificates::GetStandbyCertOfUser

Int32 Certificates::DetachStandbyCertFromUser(string adsUser, string
certId)
```

1.7 Token management

The following methods are available:

```
GetSlotIdInitialize(out int hitCount)

GetSlotIdByIndex(int index, out uint SlotId)

GetSlotIdFinalize()

SetUsedSlot(uint SlotId)

GetSlotInfo(out string description, out string manufacturer, out uint
flags, out string hwVersion, out string fwVersion)

GetTokenInfo(out string label, out string manufacturer, out string
model, out string serial, out string hwVersion, out string fwVersion)

TokenPresent(out int isPresent)

InitUserPIN(string soPIN, string newUserPIN)

ChangeUserPIN(string oldUserPIN, string newUserPIN)

ChangeSOPIN(string oldSOPIN, string newSOPIN)

BlockUserPIN()

ForcePINChange(string userPIN)

DeletePINHistory(string userPIN)

WipeToken(string SOPIN, string newUserPIN)

IssueTokenForUser(string adsUser, string userPIN, string SOPIN)

GetSGNCredenials(string userPIN, out string userName, out string
domain)
```

```
SetSGNCredenials(string userPIN, string userName, string password,
string domain)
```

```
GetAssignedUser(out string adsAssignedUser)
```

```
GetAssignedTokensInitialize(string adsUser, out int hitCount)
```

```
GetAssignedTokensByIndex(int index, out string serial)
```

```
GetAssignedTokensFinalize()
```

```
EnableDisableTokenInDB(string tokenSN, int enable)
```

```
RemoveTokenFromDB(string tokenSN)
```

```
P12ToToken(string pathP12, string P12password, string userPIN)
```

```
DeleteCertFromToken(string UserPIN, string subject, string issuer,
string serial)
```

```
GetCertFromTokenInitialize(out int hitCount)
```

```
GetCertFromTokenByIndex(int index, out string subject, out string
issuer, out string serial, out string expiryDate)
```

```
GetCertFromTokenFinalize()
```

```
CreateP12ByToken(string userPIN, string subject, string issuer, string
serial, int keylength)
```

```
P7FromToken(string subject, string issuer, string serial, string
filePathP7)
```

1.8 Inventory and status information

The following methods are available:

```
GetComputerInventory(string adsMachine, sting property, out string
value)
```

```
GetSoftwareInventory(string adsMachine, string softwareId, string
property, out string value)
```

```
GetSoftwareInventoryIdInitialize(string adsMachine, out int hitCount)
```

```
GetSoftwareInventoryIdByIndex(int index, out string softwareId)
```

```
GetSoftwareInventoryIdFinalize()
```

```
GetDriveInventory(string adsMachine, string driveId, string property,
out string value)
```

```
GetDriveInventoryIdInitialize(string adsMachine, out int hitCount)

GetDriveInventoryIdByIndex(int index, out string driveId)

GetDriveInventoryIdFinalize()


GetUserInventory(string adsMachine, string adsUser, string property,
out string value)

GetUserInventoryAdsInitialize(string adsMachine, out int hitCount)

GetUserInventoryAdsByIndex(int index, out string adsUser)

GetUserInventoryAdsFinalize()
```

1.9 Challenge/Response

The following methods are available:

```
CheckRecoveryType(string adsMachine, out int isLogon)

GetChallengeFlags(string adsMachine, string challenge, out int
challengeFlags)

CheckChallenge(string challenge, out int challengeErrorPart)

ComputeResponse(string adsUser, string adsMachine, string challenge,
int action, out string response)

SuspensionComputeResponse(string challenge, string dsnUser, uint
suspensionTime, out int opCode, out string response)

BitLockerRecovery(string adsMachine, string drive, out string
response)

OfflineCheckBAKFile(string bakFileName, out int isLogon, out string
drive)

OfflineGetChallengeFlags(string bakFileName, string challenge, out int
challengeFlags)

OfflineComputeResponse(string bakFileName, string challenge, int
action, out string response)

OfflineBitLockerRecovery(string bakFileName, string drive, out string
response)

GetVirtualClientInitialize(string searchName, out int hitCount)

GetVirtualClientByIndex(int index, out string virtualClientName, out
string virtualClientId)

GetVirtualClientFinalize()
```

```
GetVirtualClientKeyFileInitialize(string searchName, out int hitCount)
```

```
GetVirtualClientKeyFileByIndex(int index, out string keyFileName, out
string keyFileComment)
```

```
GetVirtualClientKeyFileFinalize()
```

```
VirtualClientWithKeyRecovery(string virtualClientId, string
virtualClientName, string keyId, string challenge, out string
response)
```

```
VirtualClientWithKeyFileRecovery(string virtualClientId, string
virtualClientName, string keyFileId, string challenge, out string
response)
```

1.10 Reporting

The following methods are available:

```
DeleteAllEvents(string backupFilePathName)
```

```
DeleteEventsLeaveLast(int numberOfEvents, int orderByLogTime, string
backupFilePathName)
```

```
DeleteEventsOlderThan(string date, int orderByLogTime, string
backupFilePathName)
```

1.11 Misc

The following methods are available:

```
ResetCertificateStore(string password)
```

```
ExportCompanyCertificate(string p7Filename, string p12Filename, string
p12Password, int overwriteExisting)
```

```
AddP12ToCertStore(string certStoreName, string certStorePassword,
string p12Filename, string password)
```

```
AddP7ToCertStore(string certStoreName, string p7Filename)
```

```
ExportOfficerCertificate(string p7Filename, string p12Filename, string
p12Password, int overwriteExisting )
```

```
SignFileForLCImport(string toSignFileName, string signedFileName
```

1.12 Service Accounts

The following methods are available:

```
RenameServiceAccount (string listName, string oldName, string newName,
string oldDomain, string newDomain)
```

```
DeleteServiceAccount (string listName, string name, string domain)
```

```
ServiceAccountExists (string listName, string name, string domain, out
int exists)
```

```
CreateServiceAccountList (string listName)
```

```
RenameServiceAccountList (string listName, string newListName)
```

```
DeleteServiceAccountList (string listName)
```

```
AddServiceAccountToServiceAccountList (string listName, string
userName, string domain)
```

```
RemoveServiceAccountFromList (string listName, string userName, string
domain)
```

```
ServiceAccountListExists (string listName, out int exists)
```

```
InitializeServiceAccountLists (out int hitCount)
```

```
GetServiceAccountListsByIndex (int index, out string name)
```

```
GetServiceAccountMembersByIndex (string listName, int index, out
string username, out string domain)
```

```
GetServiceAccountMembersCount (string listName, out int count)
```

1.13 Local Self Help

```
GetLocalSelfHelpParameter(out int minCountAnswers, out int
drawnCountQuestions)
```

```
SetLocalSelfHelpParameter(int minCountAnswers, int drawnCountQuestions)
```

1.14 POA group to container assignment

```
AssignPOAGroupToContainer(string dsnContainerObject, string dsnPOAGroup)
```

```
UnassignPOAGroupToContainer(string dsnContainerObject, string
dsnPOAGroup)
```

```
GetAssignedPoaGroupInitialize(string dsnContainerObject, out int
hitCount)
```

```
GetAssignedPoaGroupByIndex(int index, out string dsnPOAGroup)
```

```
GetAssignedPoaGroupByIndex(int index, out string dsnPOAGroup)
```

```
GetAssignedPoaGroupFinalize()
```

2 Detailed API Description

2.1 General

2.1.1 Scripting API and 64 bit

The SafeGuard Enterprise Scripting API only runs as a 32-bit application. Due to the fact that Win32 base components were not ported to 64 bit, the Scripting API cannot be run as a 64-bit application. You can call the Scripting API from a 32-bit shell or with %SYSTEMROOT%\SysWOW64\wscript.exe.

2.1.2 Authentication

A security officer has to authenticate against SafeGuard Enterprise Scripting.

2.1.3 General methods

Each class in the SafeGuard Enterprise Scripting API has an Initialize() and a FreeResource() method which needs to be called before and after using the object. The main module ("Base") implements a GetLastError method which can be used for advanced error processing.

2.1.4 Enumerating result lists using wildcard search methods

The wildcard search methods are intended to be used for enumerating result lists with multiple entries.

The initialize function "Get*Name*Initialized(*searchparams*, out int hitCount)" creates a result set of search parameters and a hit count.

The by-index-function "Get*Name*ByIndex(int index, *outputparams)*" delivers the result at position index. To show all results, start a loop with an index of 0 (zero) and call the method repeatedly with an index increased by 1. If the method returns `NO_MORE_DATA`, the end of the list has been reached.

The finalize function "Get*Name*Finalize()" deletes the result set and a new search can be started. A new initialize function without a finalize function of the old wildcard search returns an error.

2.1.5 Additional permission “Execute Script”

An additional permission for security officers (initially MSO and SO) “Execute Script” has been introduced. A security officer who is logged on, but does not have this permission, will not be able to perform any operations using the Scripting API (except Initialize and FreeResources). Like all other permissions, “Execute Script” can be configured to use additional authentication. In contrast to operations in the GUI, configuring additional authentication for “Execute Script” will result in the fact that the script cannot be executed at all as additional authentication is not supported by the Scripting API!

2.1.6 “Execute Script” for SafeGuard Enterprise Servers

Scripts can be executed unattendedly on each SafeGuard Enterprise Server. In order to prevent unauthorized script execution, a SafeGuard Enterprise Server will not have the right “Execute Script” by default. The MSO needs to explicitly allow script execution for a registered SafeGuard Enterprise Server.

To allow script execution for a registered SafeGuard Enterprise Server, the MSO has to proceed as follows:

1. In the menu bar of the SafeGuard Management Center, select **Tools** > **Configuration Package Tool**.
2. In tab **Register Server**, select check box **Scripting allowed** for the registered SafeGuard Enterprise Server.
3. Click button **Close**.

The SafeGuard Enterprise Server now has the right “Execute Script“.

2.2 API for Base

2.2.1 Int32 <each Class>::Initialize()

Initialize() has to be called prior to any other method calls of the object concerned.

2.2.2 Int32 <each Class>::FreeResources()

FreeResources() releases resources needed by the API. Calling FreeResources() on the main module (“Base”) will release the Security Officer session.

2.2.3 Int32 Base::CreateDirectoryClassInstance ()

Creates a new instance for Directory. This is necessary for using the API for Users & Computers management and assignment.

2.2.4 Int32 Base::CreateUMAClassInstance()

Creates a new instance for UMA. This is necessary for using the API for user-computer assignment (UMA).

2.2.5 Int32 Base::CreateKeysClassInstance()

Creates a new instance for Keys. This is necessary for using the API for key generation and assignment.

2.2.6 Int32 Base::CreateCertificatesClassInstance()

Creates a new instance for Certificates. This is necessary for using the API for certificate assignment.

2.2.7 Int32 Base::CreateTokenClassInstance()

Creates a new instance for Token. This is necessary for using the API for token management.

2.2.8 Int32 Base::CreateInventoryClassInstance()

Creates a new instance for Inventory. This is necessary for using the API for topic inventory and status information.

2.2.9 Int32 Base::CreateCRClassInstance()

Creates a new instance for CR. This is necessary for using the API for Challenge/Response.

2.2.10 Int32 Base::CreateReportsClassInstance()

Creates a new instance for Reports. This is necessary for using the API for reporting.

2.2.11 Int32 Base::CreateMiscClassInstance()

Creates a new instance for Misc. Contains miscellaneous functions required to set up and maintain the system.

2.2.12 Int32 Base::CreateServiceAccountsClassInstance()

Creates a new instance for service accounts.

2.2.13 Int32 Base::CreateLocalSelfHelpClassInstance()

Creates a new instance for Local Self Help. This is necessary to use the API for Local Self Help (LSH).

2.2.14 Int32 Base::CreatePOAGroupAssignmentClassInstance()

Creates a new instance for group to container assignment. This is necessary to use the API for POA group to container assignment.

2.2.15 Uint32 Base::GetLastError(out string errorText)

Retrieves a textual representation of the last (internal) error occurred.

errorText	Error message belonging to the internal error code.

Note: In contrast to other API methods this method returns an unsigned integer which holds the internal error code.

2.2.16 Int32 Base::AuthenticateOfficer(string officerName, string pinOrPassword, string confFilePathName)

Authenticates the currently logged on user against the SafeGuard Enterprise Scripting API. Authentication is mandatory prior to using the API.

officerName	Name of the officer to be authenticated.
pinOrPassword	Either the password for the certificate store of the logged on user in case of "no Token" or "Token optional", or the PIN of the plugged in token in case of "Token optional" or "Token mandatory". The private key required for authentication must be available in the relevant location (token or certificate store).
confFilePathName	Path and file name of the .conf file to be used [**optional**].

2.2.17 Int32 Base::AuthenticateOfficerMT(string officerName, string pinOrPassword, string configurationName)

Authenticates an officer in a multi tenancy environment. The desired configuration name has to be provided. Refer to the documentation of Multi Tenancy for further information concerning configurations. Authentication is mandatory prior to using the API.

officerName	Name of the officer to be authenticated.
pinOrPassword	Either the password for the certificate store of the logged on user in case of "no Token" or "Token optional", or the PIN of the plugged in token in case of "Token optional" or "Token mandatory". The private key required for authentication must be available in the relevant location (token or certificate store).
configurationName	Name of the configuration to be used. **Note:** SafeGuard Enterprise saves the configurations available on the system at:<CSIDL_LOCAL_APPDATA>\Utimaco\SafeGuard Enterprise\Configuration. The configurationName can be found in SGNDatabases.xml file at <string>Configuration ...</string>. It is not indicated by the file names of the *.xml files located in the Configuration folder.

2.2.18 Int32 Base::AuthenticateWHDOfficer(string OfficerName, string Password)

Authenticates Web Helpdesk officers by their P12 file stored in the database.

officerName	Name of the officer to be authenticated.
Password	Password provided

2.2.19 Int32 Base::AuthenticateService()

Authenticates a service account against the SafeGuard Enterprise Scripting API. Authentication is mandatory prior to using the API. This method can be used when a script is executed on a SafeGuard Enterprise Server, where a logged on user is not required.

Note: This method will work on activated SafeGuard Enterprise Servers only. Therefore scripts using this authentication method will only run on SafeGuard Enterprise Servers, *see Installation/ environment*, (page 84).
All actions performed by the script when using AuthenticateService will be done with the SafeGuard Enterprise Server SO account which is a Master Security Officer (same user as used by the SafeGuard Enterprise Web Service).

2.3 API for Users & Computers management

2.3.1 Int32 Directory::CreateDirectoryConnection (string dsn, string userName, string password, string serverNameIP, string port, int SSL)

Creates a new connection to a directory. The connection will be tested immediately and the process will fail if the connection does not work. The connection will be stored in the SafeGuard Enterprise database and will also appear in the GUI.

dsn	Distinguished name of the directory service to connect to.
userName	User to be used for connecting to the directory service.
password	Password of the user.
serverNameIP	Server name or IP number.
port	Port number to be used.
SSL	0 = No SSL connection. 1 = SSL connection.

2.3.2 Int32 Directory::DeleteDirectoryConnection(string dsn)

Deletes an existing directory connection from the SafeGuard Enterprise database.

dsn	Distinguished name of the directory connection.

2.3.3 Int32 Directory::SynchronizeImportedContainers(string logFileName, int membership, int accountState)

After you have imported a directory structure into the SafeGuard Enterprise Databse you can use this method to synchronize all existing containers with the SafeGuard Enterprise Database. This method does not add new containers.

logFileName	Full path and name of the log file. This parameter is mandatory. If it is invalid or empty, the process fails and an error message is returned. If the file already exists, new logs are appended to the end of the log file.
membership	0 = Memberships are synchronized too. 1 = Memberships are not synchronized.
accountState	0 = User-enabled state is synchronized too. 1 = User-enabled state is only synchronized at first synchronization.

2.3.4 Int32 Directory::GetOneObject(string searchName, string adsStartObject, int filter, out string adsObject, out string type)

Returns an object (machine, user, etc.) in the SafeGuard Enterprise directory. For more than one object hit, GetOneObject returns an error code.

searchName	Search name (wildcards * are not allowed).
adsStartObject	Full ADS path to the object where the search should start from recursively, an empty string for a search in the whole tree (not for **filter** = 3).

filter	The type of search, i.e. the type of the result object. Possible values for **filter** are: 0 All objects (except local user) 1 Computer 2 User (except local user) 3 Local user 4 Logon name (except local user) 5 Groups 6 Container 7 OU 8 Workgroups 9 Domains 10 Object GUID **Note:** When searching for local users (**filter** = 3), you have to specify a start node for **adsStartObject**.
[**out**] **adsObject**	The ADS path of the object found.
[**out**] **type**	The type of the object found at the specified index, an empty string, if no object was found. Possible values of **type** are: user — Denotes a user object. group — Denotes a group object. computer — Denotes a machine object. container — Denotes a container (not an OU). organizationalunit — Denotes an OU. domaindns — Denotes a domain node. sgroot — DenotestheSafeGuard Enterprise specific root element (exists only once). sgauthmachinegroup — Denotes the SafeGuard Enterprise specific virtual group “.Authenticated machines”. sgauthusergroup — Denotes the SafeGuard Enterprise specific virtual group “.Authenticated users”. sgcontainerunknownobjects — Denotes the SafeGuard Enterprise specific virtual group “.Autoregistered”.

2.3.5 Int32 Directory::GetObjectInitialize(string searchName, string adsStartObject, int filter, out int hitCount)

Creates a result set of the wildcard search for the object (machine, user, etc.) in the SafeGuard Enterprise directory.

searchName	Search name (wildcards * are allowed).
adsStartObject	Full ADS path to the object where the search should start from recursively, an empty string for a search in the whole tree (not for **filter** = 3).
filter	The type of search, i.e. the type of the result object. Possible values for **filter** are: 0 All objects (except local user) 1 Computer 2 User (except local user) 3 Local user 4 Logon name (except local user) 5 Groups 6 Container 7 OU 8 Workgroups 9 Domains 10 Object GUID
[**out**] **hitCount**	Count of all hits in the result set.

2.3.6 Int32 Directory::GetObjectByIndex(int index, out string adsObject, out string type)

Returns the object (machine, user, ...) found at a specified index in the SafeGuard Enterprise directory.

index	Index of the child object, zero-based.
[**out**] **adsObject**	The ADS path of the object found at the specified index, an empty string, if no object was found.
[**out**] **type**	The type of the object found at the specified index, an empty string, if no object was found. Possible values of **type** are: user — Denotes a user object. group — Denotes a group object. computer — Denotes a machine object. container — Denotes a container (not an OU). organizationalunit — Denotes an OU. domaindns — Denotes a domain node. sgroot — Denotes the SafeGuard Enterprise specific root element (exists only once). sgauthmachinegroup — Denotes the SafeGuard Enterprise specific virtual group “.Authenticated machines”. sgauthusergroup — Denotes the SafeGuard Enterprise specific virtual group “.Authenticated users”. sgcontainerunknownobjects — Denotes the SafeGuard Enterprise specific virtual group “.Autoregistered”.

2.3.7 Int32 Directory::GetObjectFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.3.8 Int32 Directory::GetMemberOfGroupInitialize(string adsGroup, out int hitCount)

Creates a result set of the wildcard search for the member object of a group in the SafeGuard Enterprise directory.

adsGroup	ADS path of the group to retrieve the member objects from.
[**out**] **hitCount**	The count of all hits in the result set.

2.3.9 Int32 Directory::GetMemberOfGroupByIndex(int index, out string adsMember, out string type)

Returns the member object at a specified index of a group in the SafeGuard Enterprise directory. This method can be used to enumerate group members.

index	Index of the child object, zero-based.
[**out**] **adsMember**	The ADS path of the member object found at the specified index. If no member was found, this is an empty string.
[**out**] **type**	The type of object found at the specified index. If no object was found, this is an empty string. Possible values of **type** are: **user** Denotes a user object. **group** Denotes a group object. **computer** Denotes a machine object.

2.3.10 Int32 Directory::GetMemberOfGroupFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.3.11 Int32 Directory::CreateUser(string userLogonName, string userFullName, string adsContainer, out string adsUser)

Creates a new user object in the SafeGuard Enterprise directory. Creates a local user object when the parent is a machine.

userLogonName	User logon name (as in ADS).
userFullName	Full name of the new user (as in ADS).
adsContainer	ADS path of the container to create the new user in.
[**out**] **adsUser**	ADS path of the created user object.

2.3.12 Int32 Directory::RenameUser(string adsUser, string newUserFullName, out string newAdsUser)

Renames an existing user or a local user in the SafeGuard Enterprise directory.

adsUser	ADS path of the user object to be renamed.
newUserFullName	The new user name.
[**out**] **newAdsUser**	ADS path of the renamed user object.

2.3.13 Int32 Directory::DeleteUser(string adsUser)

Deletes an existing user or a local user from the SafeGuard Enterprise directory.

adsUser	ADS path of the user object to be deleted.

2.3.14 Int32 Directory::MoveUser(string adsUser, string adsToContainer, out string newAdsUser)

Moves an existing user object to a different container.

adsUser	ADS path of the user object to be moved.
adsToContainer	ADS path of the container object the user object is to be moved to.
[**out**] **newAdsUser**	ADS path of the moved user object.

2.3.15 Int32 Directory::AddUserToGroup(string adsUser, string adsToGroup)

Adds an existing user or a local user to an existing group.

adsUser	ADS path of the user object to be added.
adsToGroup	ADS path of the group object the user object is to be added to.

2.3.16 Int32 Directory::RemoveUserFromGroup(string adsUser, string adsFromGroup)

Removes an existing user or a local user from a group.

adsUser	ADS path of the user object to remove.
adsFromGroup	ADS path of the group object from which the user object is to be removed.

2.3.17 Int32 Directory::GetUserProperty(string adsUser, string property, out string value)

Retrieves a property of an existing user object.

adsUser	ADS path of the user object.
property	Identifier of the required property. Possible values of **property** are: USR_FIRST_NAME First name of user (as in ADS) . USR_LAST_NAME Last name of user (as in ADS). USR_INITIALS User initials (as in ADS). USR_LOGON_NAME User logon name (as in ADS). USR_DOMAIN_NAME User domain name (as in ADS). USR_LOGON_NAME_PRE2000 Pre-W2k logon name (as in ADS). USR_LOGON_DOMAIN Logon domain user name (as in ADS). USR_DOMAIN_NAME_PRE2000 Pre-W2k domain name (as in ADS). USR_EMAIL E-mail address of user (as in ADS). USR_TOKEN_USER_LOGON Logon name used for issuing a token. USR_TOKEN_PASSWORD Password used for issuing a token. USR_TOKEN_DOMAIN Domain name used for issuing a token. USR_TOKEN_ISSUE_CREDENTIALS 1 = Issuing token possible. 0 = Issuing token not possible.
[**out**] **value**	Value of the property (string representation).

2.3.18 Int32 Directory::SetUserProperty(string adsUser, string property, string value)

Sets a property of an existing user object.

adsUser	ADS path of the user object.
property	Identifier of the property. Possible values of **property** are: USR_FIRST_NAME — First name of user (as in ADS). USR_LAST_NAME — Last name of user (as in ADS). USR_INITIALS — User initials (as in ADS). USR_LOGON_NAME — User logon name (as in ADS). USR_DOMAIN_NAME — User domain name (as in ADS). USR_LOGON_NAME_PRE2000 — Pre-W2k logon name (as in ADS). USR_DOMAIN_NAME_PRE2000 — Pre-W2k domain name (as in ADS). USR_LOGON_DOMAIN — Logon domain user name (as in ADS). USR_EMAIL — E-mail address of user (as in ADS). USR_TOKEN_USER_LOGON — Logon name used for issuing a token. USR_TOKEN_PASSWORD — Password used for issuing a token. USR_TOKEN_DOMAIN — Domain name used for issuing a token. USR_TOKEN_ISSUE_CREDENTIALS — 1 = Issuing token possible. 0 = Issuing token not possible.
value	Value of the property (string representation).

2.3.19 Int32 Directory::CreateMachine(string machineName, string adsContainer, out string adsMachine)

Creates a new machine object in the SafeGuard Enterprise directory.

machineName	Machine name (as in ADS).
adsContainer	ADS path of the container to create the new machine in.
[**out**] **adsMachine**	ADS path of the created machine object.

2.3.20 Int32 Directory::RenameMachine(string adsMachine, string newMachineName, out string newAdsMachine)

Renames an existing machine object in the SafeGuard Enterprise directory.

adsMachine	ADS path of the machine to be renamed.
newMachineName	New name of the machine.
[**out**] **adsMachine**	New ADS path of the renamed machine object.

2.3.21 Int32 Directory::DeleteMachine(string adsMachine)

Deletes an existing machine object from the SafeGuard Enterprise directory.

adsMachine	ADS path of the machine to be deleted.

2.3.22 Int32 Directory::MoveMachine(string adsMachine, string adsToContainer, out string newAdsMachine)

Moves an existing machine object to a different container.

adsMachine	ADS path of the machine to be moved.
adsToContainer	ADS path of the destination container.
[**out**] **newAdsMachine**	New ADS path of the machine.

2.3.23 Int32 Directory::AddMachineToGroup(string adsMachine, string adsToGroup)

Adds an existing machine object to an existing group.

adsMachine	ADS path of the machine to be added.
adsToGroup	ADS path of the group the machine is to be added to.

2.3.24 Int32 Directory::RemoveMachineFromGroup(string adsMachine, string adsFromGroup)

Removes an existing machine object from an existing group.

adsMachine	ADS path of the machine to be removed.
adsFromGroup	ADS path of the group from which the machine is to be removed.

2.3.25 Int32 Directory::GetMachineProperty(string adsMachine, string property, out string value)

Retrieves a property of an existing machine.

adsMachine	ADS path of the machine
property	Identifier of the property to retrieve Possible value of **property**: MAS_WOL_CMD Wake On Lan command, possible values are "Start", "Stop" and "Nothing".
[**out**] **value**	Value of the property (string representation).

2.3.26 Int32 Directory::SetMachineProperty(string adsMachine, string property, string value)

Sets a property of an existing machine.

adsMachine	ADS path of the machine.
property	Identifier of the property to set. Possible value of **property**: MAS_WOL_CMD Wake On Lan command, possible values are "Start", "Stop" and "Nothing".
value	Value of the property (string representation).

2.3.27 Int32 Directory::CreateOU(string ouName, string adsParentContainer, out string newAdsOU)

Creates a new OU object in the SafeGuard Enterprise directory.

ouName	Name of the OU to be created.
adsParentContainer	ADS path of the container to create the OU in.
[**out**] **newAdsOU**	ADS path of the OU created.

2.3.28 Int32 Directory::CreateContainer(string containerName, string adsParentContainer, out string newAdsContainer)

Creates a new container object in the SafeGuard Enterprise directory.

containerName	Name of the container to be created.
adsParentContainer	ADS path of the container to create the container in.
[**out**] **newAdsContainer**	ADS path of the container created.

2.3.29 Int32 Directory::CreateDomain(string domainName, string distinguishedName, string domainNetbios)

Creates a new domain object in the SafeGuard Enterprise directory.

domainName	Name of the domain to be created.
distinguishedName	Distinguished name (ADS path) of the domain.
domainNetbios	Domain NetBIOS of the domain.

2.3.30 Int32 Directory::CreateWorkgroup(string workgroupName, out string newAdsWorkgroup)

Creates a new workgroup object in the SafeGuard Enterprise directory.

workgroupName	Name of the workgroup to be created.
[**out**] **newAdsWorkgroup**	ADS path of the workgroup created.

2.3.31 Int32 Directory::RenameContainer(string adsContainer, string newContainerName, out string newAdsContainer)

Renames an existing container, OU, domain or workgroup object in the SafeGuard Enterprise directory.

adsContainer	ADS path of the container, OU, domain or workgroup object to be renamed.
newContainerName	New name of the container, OU, domain or workgroup object.
[**out**] **newAdsContainer**	New ADS path of the container, OU, domain or workgroup object.

2.3.32 Int32 Directory::DeleteContainer(string adsContainer)

Deletes an existing container, OU, domain or workgroup object from the SafeGuard Enterprise directory.

adsContainer	ADS path of the container, OU, domain or workgroup object to be deleted.

Note: All objects in the container, subcontainers, groups etc., will be deleted recursively.

2.3.33 Int32 Directory::MoveContainer(string adsContainer, string adsToContainer, out string newAdsContainer)

Moves an existing container or OU in the SafeGuard Enterprise directory.

adsContainer	ADS path of the container to be moved.
adsToContainer	ADS path of the container to move the container to.
[**out**] **newAdsContainer**	New ADS path of the moved container.

Note: All objects in the container, subcontainers, groups etc., will also be moved and they will receive new ADS path values.

2.3.34 Int32 Directory::CreateGroup(string groupName, string adsContainer, out string adsGroup)

Creates a new group object in the SafeGuard Enterprise directory.

groupName	Group name (as in ADS).
adsContainer	ADS path of the container to create the new group in.
[**out**] **adsGroup**	The ADS path of the group object created.

2.3.35 Int32 Directory::RenameGroup(string adsGroup, string newGroupName, out string newAdsGroup)

Renames an existing group object in the SafeGuard Enterprise directory.

adsGroup	ADS path of the group to be renamed.
newGroupName	New name of the group.
[**out**] **newAdsGroup**	ADS path of the renamed group object.

2.3.36 Int32 Directory::DeleteGroup(string adsGroup)

Deletes an existing group object from the SafeGuard Enterprise directory.

adsGroup	ADS path of the group to be deleted.

2.3.37 Int32 Directory::MoveGroup(string adsGroup, string adsToContainer, out string newAdsGroup)

Moves an existing group object to a different container in the SafeGuard Enterprise directory.

adsGroup	ADS path of the group to be moved.
adsToContainer	ADS path of the container to move the group to.
[**out**] **newAdsGroup**	New ADS path of the group moved.

2.3.38 Int32 Directory::AddGroupToGroup(string adsGroup, string adsToGroup)

Adds an existing group object to an existing group in the SafeGuard Enterprise directory.

adsGroup	ADS path of the group to be added.
adsToGroup	ADS path of the group to add the group to.

2.3.39 Int32 Directory::RemoveGroupFromGroup(string adsGroup, string adsFromGroup)

Removes an existing group object from a group in the SafeGuard Enterprise directory.

adsGroup	ADS path of the group to be removed.
adsFromGroup	ADS path of the group from which the group is to be removed.

2.3.40 Int32 Directory::GetSGDProperty(string adsPath, string property, out string value)

Retrieves a property from the SafeGuard Enterprise directory.

adsPath	ADS path of the directory object.
property	Identifier of the required property. Possible values of **property** are: SGD_DESCRIPTION (all objects) Description SGD_IMP_ID (all objects) The GUID of the object defined in Active Directory (not SID!). SGD_CREATE_DATE (all objects) The creation date of the objects in the SafeGuard Enterprise directory (string representation yyyy-mm-ddThh:mm:ss, e.g. 2001-04-10T15:51:24) SGD_BLOCK_INHERITANCE Indicates whether inheritance will be (container only) blocked on the container or not. SGD_ACCOUNT_STATE (user/ users/computers) Indicates whether user/local machine is disabled or not. 0 = unlocked, 1 = locked
[**out**] **value**	Value of the property (string representation).

2.3.41 Int32 Directory::SetSGDProperty(string adsPath, string property, string value)

Sets a property in the SafeGuard Enterprise directory.

adsPath	ADS path of the directory object.
property	Identifier of the required property. Possible values of **property** are: SGD_DESCRIPTION (all objects) Description SGD_IMP_ID (all objects) — The GUID of the object defined in Active Directory (not SID!). SGD_BLOCK_INHERITANCE — Indicates whether the inheritance will be (container only) blocked on the container or not. SGD_ACCOUNT_STATE — Indicates whether user/local (user/users/computers) machine is disabled or not. 0 = unlocked, 1 = locked.
value	Value of the property (string representation).

2.3.42 Int32 Directory::SetObjectToAD(string adsPath)

Sets a SafeGuard Enterprise object (only machine, user, local user, group) to an AD object for synchronization with the Active Directory.

adsPath	ADS path of the directory object.

2.3.43 Int32 Directory::SetObjectToSG(string adsPath)

Sets an AD Object (only machine, user local user, group) to a SafeGuard Enterprise object.

adsPath	ADS path of the directory object.

2.3.44 Int32 Directory::ConvertADGuidToSGNGuid(object adGuid, out string sgnGuid)

Converts an AD GUID into a SafeGuard Enterprise GUID, e.g. as input for set properties SGD_IMP_ID.

adGuid	GUID in Active Directory form.
sgnGuid	GUID in SafeGuard Enterprise form.

2.3.45 Int32 Directory::CanSynchronizeDirectory(out int canSynchronize)

Checks, if synchronization can be performed by the API. In case a synchronization process is currently running (e.g. in a Management Center or in another instance of the API) an attempt to synchronize the directory will fail.

[**out**] **canSynchronize**	1 = synchronization allowed, 0 = synchronization forbidden.

2.3.46 Int32 Directory::ResetSynchronizationLock()

If the synchronization lock is not reset by the synchronization algorithm automatically (e.g. due to network problems or system crashes), this function can be used to reset the flag to make synchronization possible again.

Note: Do NOT use while a synchronization is running.

2.4 API for user-computer assignment (UMA)

2.4.1 Int32 UMA::CreateUMA(string adsUser, string adsMachine)

Creates a new user-machine assignment in the SafeGuard Enterprise database. For **adsUser** or **adsMachine** a group can also be specified. In this case all direct members (users or computers) of the relevant group are assigned to the other object.

adsUser	ADS path of the user or group.
adsMachine	ADS path of the machine or group.

2.4.2 Int32 UMA::DeleteUMA(string adsUser, string adsMachine)

Deletes an existing user-machine assignment from the SafeGuard Enterprise database. For **adsUser** or **adsMachine** a group can also be specified. In this case the user-machine assignments for all direct members (users or computers) of the relevant group are deleted.

adsUser	ADS path of the user.
adsMachine	ADS path of the machine.

2.4.3 Int32 UMA::SetUMAProperty(string adsUser, string adsMachine, string property, string value)

Sets a UMA property in the SafeGuard Enterprise database.

adsUser	ADS path of the user object.
adsMachine	ADS path of the machine object.
property	Identifier of the desired property. Possible values of **property** are: UMA_LOCK_DOWN 1 = true, 0 = false. UMA_IS_OWNER 1 = true, 0 = false. UMA_DENY_OWNERSHIP 1 = true, 0 = false. **Note:** If you use UMA_IS_OWNER to set an owner for the machine, it is checked whether a different owner has already been set for this machine. If so, an error message is displayed. To change the owner, set the UMA_IS_OWNER property for the old owner to false first. If you try to set an owner, for whom the UMA_DENY_OWNERSHIP property is set to true, an error message is displayed.
value	Value of the property (string representation).

2.4.4 Int32 UMA::GetUMAProperty(string adsUser, string adsMachine, string property, out string value)

Gets a UMA property from the SafeGuard Enterprise database.

adsUser	ADS path of the user object.
adsMachine	ADS path of the machine object.
property	Identifier of the required property. Possible values of **property** are: UMA_LOCK_DOWN 1 = true, 0 = false. UMA_IS_OWNER 1 = true, 0 = false. UMA_DENY_OWNERSHIP 1 = true, 0 = false.
[**out**] **value**	Value of the property (string representation).

2.4.5 Int32 UMA::GetUMAOfUserInitialize(string adsUser, out int hitCount)

Creates a result set of the wildcard search for the machine which is assigned to a user from the SafeGuard Enterprise directory.

adsUser	ADS path of the user.
[**out**] **hitCount**	Count of all hits in the result set.

2.4.6 Int32 UMA::GetUMAOfUserByIndex(int index, out string adsMachine)

Returns the machine at the specified index which is assigned to a user from the SafeGuard Enterprise database. This method can be used for enumerating the machines of a user.

index	Index of the machine object, zero-based.
[**out**] **adsMachine**	ADS path of the machine found at the specified index. If no object was found, this is an empty string.

2.4.7 Int32 UMA::GetUMAOfUserFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.4.8 Int32 UMA::GetUMAOfMachineInitialize(string adsMachine, out int hitCount)

Creates a result set of the wildcard search for the user who is assigned to a machine from the SafeGuard Enterprise directory.

adsMachine	ADS path of the machine.
[**out**] **hitCount**	Count of all hits in the result set.

2.4.9 Int32 UMA::GetUMAOfMachineByIndex(int index, string adsUser)

Returns the user at the specified index who is assigned to a machine from the SafeGuard Enterprise database. This method can be used for enumerating the users of a machine.

index	Index of the user object, zero-based.
[**out**] **adsUser**	ADS path of the user found at the specified index. If no object was found, this is an empty string.

2.4.10 Int32 Directory::GetUMAOfMachineFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.5 API for key generation and assignment

2.5.1 Int32 Keys::CreateKey (string adsObject, string desiredName, string binaryValue, out string symbolicName, out string keyId)

Creates a new key in the SafeGuard Enterprise database. The user has to provide a desired name, which will be converted and potentially changed to the symbolic name of the key.

adsObject	ADS path of the object (no computer allowed) to assign the new key to.
desiredName	Desired/initial name for the key (e.g. OU_SGEnterprise).
binaryValue	[optional] Binary value of the key, random key if empty (").
[**out**] **symbolicName**	Symbolic name created (e.g. OU_SGEnterprise_8).
[**out**] **keyId**	ID of the key created (e.g. 0x2E87A…).

2.5.2 Int32 Keys::GetKeyBySymbolicNameInitialize(string symbolicName, out int hitCount)

Creates a result set of the wildcard search for keys in the database which match the symbolic name provided.

symbolicName	Symbolic name to search for.
[**out**] **hitCount**	Count of all hits in the result set.

2.5.3 Int32 Keys::GetKeyBySymbolicNameByIndex(int index, out string keyId)

Searches for keys in the database which match the symbolic name provided. Returns the key ID from the specified index.

index	Index in hit list.
[**out**] **keyId**	ID of the key found at the specified index.

2.5.4 Int32 Keys::GetKeyBySymbolicNameFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.5.5 Int32 Keys::GetKeyByIdInitialize(string keyId, out int hitCount)

Creates a result set of the wildcard search for keys in the database which match the key ID provided.

keyId	Key ID to search for.
[**out**] **hitCount**	Count of all hits in the result set.

2.5.6 Int32 Keys::GetKeyByIdByIndex(int index, out string keyId, out string keyName)

Searches for keys in the database which match the symbolic name provided. Returns the key ID and name at the specified index.

index	Index in hit list.
[**out**] **keyId**	ID of the key found at the specified index.
[**out**] **keyName**	Name of the key found at the specified index.

2.5.7 Int32 Keys::GetKeyByIdFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.5.8 Int32 Keys::GetKeyProperty (string keyId, string property, out string value)

Gets the specified property of the specified key.

keyId	ID of the key.
property	Property identifier. Possible values of **property** are: KIN_DESCRIPTION KIN_EXPIRY_DATE (mm/dd/yyyy)
[**out**] **value**	Value of the desired property (string representation).

2.5.9 Int32 Keys::SetKeyProperty (string keyId, string property, string value)

Sets the specified property of the specified key

keyId	ID of the key.
property	Property identifier. Possible values of **property** are: KIN_DESCRIPTION KIN_EXPIRY_DATE (mm/dd/yyyy)
value	Value of the desired property (string representation).

2.5.10 Int32 Keys::AssignKey (string adsObject, string keyId)

Assigns a key to an object.

adsObject	ADS path of the object (no computer allowed) to assign the key to.
keyId	ID of the key to assign.

2.5.11 Int32 Keys::DetachKey (string adsObject, string keyId)

Detaches an assigned key from an object.

adsObject	ADS path of the object to detach the key from.
keyId	ID of the key to detach.

2.5.12 Int32 Keys::GetAssignedKeyInitialize(string adsObject, out int hitCount)

Creates a result set of the wildcard search for keys in the database which are assigned to a specific object.

adsObject	ADS path of the object.
[**out**] **hitCount**	The count of all hits in the result set.

2.5.13 Int32 Keys::GetAssignedKeyByIndex(int index, out string keyId)

Searches for keys of a type in the database which are assigned to a specific object. Returns the key ID from the specified index.

index	Index in hit list.
[**out**] **keyId**	ID of the key found at the specified index.

2.5.14 Int32 Keys::GetAssignedKeyFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.5.15 Int32 Keys::GetAssignedObjectInitialize(string keyId, out int hitCount)

Creates a result set of the wildcard search for objects in the database to which a specific key is assigned.

keyId	ID of the key.
[**out**] **hitCount**	Count of all hits in the result set.

2.5.16 Int32 Keys::GetAssignedObjectByIndex(int index, out string adsObject)

Searches for objects in the database to which a specific key is assigned. Returns the ADS path of the object from the specified index

index	Index in hit list.
[**out**] **adsObject**	ADS path of the object at the specified index.

2.5.17 Int32 Keys::GetAssignedObjectFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.6 API for certificate assignment

2.6.1 Int32 Certificates::ImportAndAssignCertToUser(string adsUser, string pathP12, string pathP7)

Imports a certificate and (optionally) a key file to the SafeGuard Enterprise database and assigns them to an existing user.

adsUser	ADS path of the user.
pathP12	Path and file name of the P12 file to import (optional, can be empty).
pathP7	Path and file name of the P7 file to import (mandatory).

2.6.2 Int32 Certificates::CreateAndAssignCertForUser(string adsUser, string password)

Internally creates a certificate and assigns it to an existing user.

adsUser	ADS path of the user.
password	The user's password.

Note: For this function, the user's password has to be provided. Typically, this is the Windows password which may not be available to the caller.

2.6.3 Int32 Certificates::GetCertOfUserInitialize(string adsUser, out int hitCount)

Creates a result set of the wildcard search for the certificate which is assigned to a user from the SafeGuard Enterprise database.

adsUser	ADS path of the user.
[**out**] **hitCount**	Count of all hits in the result set.

2.6.4 Int32 Certificates::GetCertOfUserByIndex(int index, out string certId)

Returns the certificate at the specified index which is assigned to a user from the SafeGuard Enterprise database. This method can be used to enumerate the certificates of users.

Index	Index of the certificate object, zero-based.
[**out**] **certId**	Certificate identifier (at the specified index). Can be used as input parameter for GetCertInfo().

2.6.5 Int32 Certificates::GetCertOfUserFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.6.6 Int32 Certificates::GetUserOfCertInitialize(string subject, string issuer, string serial, out int hitCount)

Creates a result set of the wildcard search for the user to whom the certificate is assigned in the SafeGuard Enterprise database.

subject	Subject of the certificate to query for.
issuer	Issuer of the certificate to query for.
serial	Serial number of the certificate to query for.
[**out**] **hitCount**	Count of all hits in the result set.

2.6.7 Int32 Certificates::GetUserOfCertByIndex(int index, out string adsUser)

Returns the user at the specified index to whom the certificate is assigned in the SafeGuard Enterprise database. This method can be used to enumerate users to whom a specific certificate is assigned.

index	Index of the user object.
[**out**] **adsUser**	User object at the specified index.

2.6.8 Int32 Certificates::GetUserOfCertFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.6.9 Int32 Certificates::GetOneCertificate(string subject, string issuer, string serial, out string certId)

Searches for a specific certificate in the database and returns the certificate ID.

subject	Subject of the certificate to query for.
issuer	Issuer of the certificate to query for.
serial	Serial number of the certificate to query for.
[**out**] **certId**	Certificate identifier (at the specified index). Can be used as input parameter for GetCertInfo().

2.6.10 Int32 Certificates::DetachCertFromUser(string adsUser, string certId)

Removes the assignment of a certificate to a user in the database.

adsUser	ADS path of the desired user.
certId	ID of the certificate.

2.6.11 Int32 Certificates::GetCertInfo(string certId, out string subject, out string issuer, out string serial, out string startDate, out string endDate)

Retrieves certificate properties of a certificate specified by its certificate ID.

certId	Certificate ID to query for.
subject	Subject of the certificate (empty string, if not found).
issuer	Issuer of the certificate (empty string, if not found).
serial	Serial number of the certificate (empty string, if not found).
startDate	Start date of the certificate (string representation mm/dd/yyyy).
endDate	Expiry date of the certificate (string representation mm/dd/yyyy).

2.6.12 Int32 Certificates::DeleteCertFromDB(string certId)

Deletes a certificate from the database.

certId	Certificate ID of the certificate to be deleted.

Note: Certificates currently assigned to a user cannot be deleted. In this case the function will fail. Certificates have to be detached from a user beforehand.
For information on detaching certificates, *see Int32 Certificates::DetachCertFromUser(string adsUser, string certId)*, (page 50).

2.6.13 Int32 Certificates::RenewCertificate(string certId, string password)

Renews a certificate in the database by the time it is configured in the SafeGuard Management Center (**Tools menu** > **Options**). Only certificates created by SafeGuard Enterprise - either automatically by the server or by CreateAndAssignCertForUser() - can be renewed. Otherwise, this function will fail.

For this function, the user's password has to be provided. Typically, this is the Windows password which may not be available to the caller.

certId	Certificate ID of the certificate to be renewed.
password	Password of the certificate to be renewed.

2.6.14 Int32 Certificates::ImportCRL(string pathCRL)

Imports a Certificate Revocation List to the SafeGuard Enterprise database.

pathCRL	Path and file name of the CRL file to import. The CRL will only be placed in the database and not (as in the GUI) in the trusted root store.

2.6.15 Int32 Certificates::ImportCACert(string pathCACert)

Imports CA certificates to the SafeGuard Enterprise database.

pathCACert	Path and file name of the CA certificate to import.

2.6.16 Int32 Certificates::DeleteCRL(string crlfile name)

Deletes a Certificate Revocation List from the database.

crlFileName	File name of the binary blob containing the CRL.

2.6.17 Int32 Certificates::DeleteAllCRL()

Deletes all Certificate Revocation Lists from the database.

2.6.18 Int32 Certificates::DeleteCACert (string subject, string issuer, string serial)

Deletes a CA certificate from the database.

subject	Subject of the certificate to be deleted.
issuer	Issuer of the certificate to be deleted.
serial	Serial number of the certificate to be deleted.

2.6.19 Int32 Certificates::DeleteAllCACerts()

Deletes all CA certificates from the database.

2.6.20 Int32 Certificates::ImportAndAssignStandbyCertToUser(string adsUser, string pathP12, string pathP7)

adsUser	ADS path of the user.
pathP12 (optional	Path and filename of the P12 file to import and assign.
pathP7	Path and filename of the P7 file to import and assign.

2.6.21 Int32 Certificates::GetStandbyCertOfUser(string adsUser, out string certId)

adsUser	ADS path of the user.
[**out**] **certId**	Certificate ID found.

2.6.22 Int32 Certificates::DetachStandbyCertFromUser(string adsUser, string certId)

adsUser	ADS path of the user.
certId	Standby certificate to delete.

2.7 API for token management

2.7.1 Int32 Token::GetSlotIdInitialize(out int hitCount)

Creates a result set of the wildcard search for the slot ID of the token slot.

A slot ID is required for SetUsedSlot, which has to be called in order to be able to perform token operations.

[**out**] **hitCount**	Count of all hits in the result set.

2.7.2 Int32 Token::GetSlotIdByIndex(int index, out int slotId)

Returns the slot ID of the token slot at the specified index. This function can be used to enumerate token slots.

index	Index of the slot list.
[**out**] **slotId**	Slot ID from specified index.

2.7.3 Int32 Token::GetSlotIdFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.7.4 Int32 Token::SetUsedSlot(int slotId)

Selects the slot for further token operations.

Note: The slot ID has to be selected prior to any other calls which affect a token

slotId	Slot ID to be used for follow-up operations.

2.7.5 Int32 Token::GetSlotInfo(out string description, out string manufacturer, out int flags, out string hwVersion, out string fwVersion)

Retrieves information about the currently selected token slot.

[out] **description**	Description of the slot (e.g. reader name).
[out] **manufacturer**	Manufacturer (e.g. Omnikey).
[out] **flags**	Flags according to PKCS11.
[out] **hwVersion**	Hardware version number.
[out] **fwVersion**	Firmware version number.

2.7.6 Int32 Token::GetTokenInfo(out string label, out string manufacturer, out string model, out string serial, out string hwVersion, out string fwVersion)

Retrieves information about the token in the currently selected token slot.

[out] **label**	Label of the token.
[out] **manufacturer**	Manufacturer of the token.
[out] **model**	Token model information string.
[out] **serial**	Token serial number.
[out] **hwVersion**	Hardware version number.
[out] **fwVersion**	Firmware version number.

2.7.7 Int32 Token::TokenPresent(out int isPresent)

Checks, if a token has been plugged in the slot currently selected.

[out] **isPresent**	1 = Token plugged in. 0 = No token.

2.7.8 Int32 Token::InitUserPIN(string soPIN, string newUserPIN)

Initializes the user PIN of the token (setting it to a new value without knowing the original value). This call will also unblock a blocked token.

soPIN	Security Officer PIN of the plugged in token.
newUserPIN	New user PIN.

2.7.9 Int32 Token::ChangeUserPIN(string oldUserPIN, string newUserPIN)

Changes the user PIN of the plugged in token.

oldUserPIN	Current user PIN of the token.
newUserPIN	New user PIN of the token.

2.7.10 Int32 Token::ChangeSOPIN(string oldSOPIN, string newSOPIN)

Changes the Security Officer PIN of the plugged in token.

oldSOPIN	Current SO PIN of the token.
newSOPIN	New SO PIN of the token.

2.7.11 Int32 Token::BlockUserPIN()

Blocks the user PIN of the plugged in token. The token can be unblocked using InitUserPIN().

2.7.12 Int32 Token::ForcePINChange(string userPIN)

Sets a flag on the token which will force the user to change the user PIN during logon.

userPIN	Current user PIN of the token.

2.7.13 Int32 Token::DeletePINHistory(string userPIN)

Deletes the PIN history stored on the token.

userPIN	Current user PIN of the token.

2.7.14 Int32 Token::WipeToken(string SOPIN, string newUserPIN)

Completely deletes the SafeGuard Enterprise data stored on the token. Certificates will not be affected by this function. When Security Officer PIN and user PIN are provided, the user PIN will be set to newUserPIN. If no Security Officer PIN is provided, the value given in newUserPIN is used for token logon.

SOPIN	Security Officer PIN [optional].
newUserPIN	Either the current user PIN of the token or the new user PIN for the token.

2.7.15 Int32 Token::IssueTokenForUser(string adsUser, string userPIN, string SOPIN)

Issues a token for a user in the SafeGuard Enterprise directory. User credentials must have been configured for the user and USR_TOKEN_ISSUE_CREDENTIALS must be set to 1. If Security Officer PIN and user PIN are provided, the user PIN will be set to newUserPIN. If no Security Officer PIN is provided, the value given in newUserPIN is used for token logon.

adsUser	ADS path of the user for whom the token is to be issued.
userPIN	Either the current user PIN of the token or the new user PIN for the token.
SOPIN	Security Officer PIN [optional].

2.7.16 Int32 Token::GetSGNCredentials(string userPIN, out string userName, out string domain)

Retrieves the SafeGuard Enterprise credentials currently stored on the plugged in token. SafeGuard Enterprise credentials will be used for logon in case of non-cryptographic tokens.

userPIN	User PIN of the token.
[**out**] **userName**	User name stored on the token.
[**out**] **domain**	Domain information stored on the token.

2.7.17 Int32 Token::SetSGNCredentials(string userPIN, string userName, string password, string domain)

Writes the SafeGuard Enterprise credentials to the plugged in token. SafeGuard Enterprise credentials will be used for logon in case of non-cryptographic tokens.

userPIN	User PIN of the token.
userName	User name.
password	(Windows) password.
domain	Domain information.

Note: It is not recommended to use this function as the user/token assigment is not stored in the database.
In this case, use IssueTokenForUser() instead.
All parameters can be empty. This will cause the SafeGuard Enterprise credentials to be removed from the token.

2.7.18 Int32 Token::GetAssignedUser(out string adsAssignedUser)

Retrieves the user the token was issued for using IssueTokenForUser().

[**out**] **adsAssignedUser**	ADS path of the user for whom the token was issued.

2.7.19 Int32 Token::GetAssignedTokensInitialize(string adsUser, out int hitCount)

Creates a result set of the wildcard search for serial numbers of tokens which were issued for a specific user.

adsUser	ADS path of the user to query for issued tokens.
[**out**] **hitCount**	Count of all hits in the result set.

2.7.20 Int32 Token::GetAssignedTokensByIndex(int index, out string serial)

Retrieves serial numbers of tokens at the specified index which were issued to a specific user.

index	Index in list of assigned tokens.
[**out**] **serial**	Serial number from index.

2.7.21 Int32 Token::GetAssignedTokensFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.7.22 Int32 Token::EnableDisableTokenInDB(string tokenSN, int enable)

Enables or disables a token which is stored in the issued tokens table. All disabled tokens form the "Token Blacklist" which will be replicated to the clients. Disabled tokens cannot be used to log on.

tokenSN	Serial number of the token to be modified.
enable	1 = Enable token/remove from blacklist. 0 = Disable token/put on blacklist.

2.7.23 Int32 Token::RemoveTokenFromDB(string tokenSN)

Removes a token from the issued tokens table.

tokenSN	Serial number of the token to be removed.

2.7.24 Int32 Token::P12ToToken(string pathP12, string P12password, string userPIN)

Imports a key file to the plugged in token. Note that the plugged in token must be able to handle RSA key pairs with the desired key length.

pathP12	Path and file name of the P12 file holding the key pair(s) to import.
P12password	Password for the P12 file.
userPIN	User PIN for the plugged in token.

2.7.25 Int32 Token::DeleteCertFromToken(string UserPIN, string subject, string issuer, string serial)

Searches for a certificate on the token and (if found) deletes it from the token.

userPIN	User PIN for the plugged in token.
subject	Certificate subject to search for.
issuer	Certificate issuer to search for.
serial	Certificate serial number to search for.

2.7.26 Int32 Token::GetCertFromTokenInitialize(out int hitCount)

Creates a result set of the wildcard search for properties of certificates found on the plugged in token.

[**out**] **hitCount**	Count of all hits in the result set.

2.7.27 Int32 Token::GetCertFromTokenByIndex(int index, out string subject, out string issuer, out string serial, out string expiryDate)

Returns properties of certificates found on the plugged in token at the specified index.

index	Index of the certificate on the token.
[**out**] **subject**	Certificate subject.
[**out**] **issuer**	Certificate issuer.
[**out**] **serial**	Certificate serial number.
[**out**] **expiryDate**	Certificate expiry date (dd/mm/yyyy), string representation.

2.7.28 Int32 Token::GetCertFromTokenFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.7.29 Int32 Token::CreateP12ByToken(string userPIN, string subject, string serial, int keylength)

Creates a key pair/certificate by the plugged in token.

The private key cannot be exported. The key pair is stored in a self-signed certificate. The token must be able to handle keys with the desired key length.

userPIN	User PIN of the plugged in token.
subject	Certificate subject.
serial	Certificate serial number.
keylength	Desired key length of the RSA key pair (allowed values: 1024, 2048, 4096).

2.7.30 Int32 Token::P7FromToken(string subject, string issuer, string serial, string filePathP7)

Searches for a certificate on the token and (if found) exports the certificate (P7) to a file.

subject	Certificate subject to search for.
issuer	Certificate issuer to search for.
serial	Certificate serial number to search for.
filePathP7	Path and file name for the certificate file to be created.

Note: Existing files will be overwritten without warning.

2.8 API for inventory and status information

2.8.1 Int32 Inventory::GetComputerInventory(string adsMachine, string property, out string propertyValue, out string propertyString)

Retrieves an inventory property for a machine from the database (machine inventory).

adsMachine	ADS path of the machine to retrieve the inventory property from.
property	Property name.
[**out**] **propertyValue**	Inventory value from database. Values for **property** are defined as: MachineName LastPolicyReceived EncryptedDrives UnencryptedDrives OperatingSystem POA POAType WOL Attack ModificationDate ParentDSN
[**out**] **propertyString**	Value of the desired property (string representation).

2.8.2 Int32 Inventory::GetDriveInventory(string adsMachine, string driveId, string property, out string propertyValue, out string propertyString)

Retrieves a drive inventory property for a machine from the database.

adsMachine	ADS path of the machine to get the drive inventory property from.
driveId	Drive ID (from GetDriveInventoryIdByIndex()).
property	Property name.
[**out**] **propertyValue**	Inventory value from database, property values are defined as: DriveName Type State Algorithm
[**out**] **propertyString**	Value of the desired property (string representation).

2.8.3 Int32 Inventory::GetDriveInventoryIdInitialize(string adsMachine, out int hitCount)

Creates a result set of the wildcard search for drive inventory IDs.

adsMachine	ADS path of the machine.
[**out**] **hitCount**	Count of all hits in the result set.

2.8.4 Int32 Inventory::GetDriveInventoryIdByIndex(int index, out string driveId)

Gets a drive inventory ID from the specified index. This function can be used to enumerate drive IDs of the inventory of a specific machine.

index	Index in hit list.
[**out**] **driveId**	Software ID at the specified index.

2.8.5 Int32 Inventory::GetDriveInventoryIdFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.8.6 Int32 Inventory::GetUserInventory(string adsMachine, string adsUser, string property, out string propertyValue, out string propertyString)

Retrieves a user inventory property for a machine from the database.

adsMachine	ADS path of the machine to get the software inventory property from.
adsUser	ADS path of the user.
property	Property name.
[**out**] **propertyValue**	Inventory value from database, property values are defined as: UserName UserIsOwner
[**out**] **propertyString**	Value of the required property (string representation).

2.8.7 Int32 Inventory::GetUserInventoryAdsInitialize(string adsMachine, out int hitCount)

Creates a result set of the wildcard search for the user ADS path of a machine's user inventory.

adsMachine	ADS path of the machine.
[**out**] **hitCount**	Count of all hits in the result set.

2.8.8 Int32 Inventory::GetUserInventoryAdsByIndex(int index, out string adsUser)

Gets a user ADS path from the specified index of a machine's user inventory. This function can be used to enumerate the users of the inventory of a specific machine.

index	Index in hit list.
[**out**] **adsUser**	User ADS path at the specified index.

2.8.9 Int32 Inventory::GetUserInventoryAdsFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.8.10 Int32 Inventory::GetFeatureInventory(string adsMachine, string moduleName, string property, out string propertyValue, out string propertyString)

Retrieves a feature inventory property for a machine from the database.

adsMachine	ADS path of the machine to retrieve the feature inventory property from.
moduleName	Module name of the feature.You can retreive a module name from the inventory of a specific name by using **Int32 Inventory::GetFeature InventoryModuleByIndex(int index, out string moduleName**. For details *see Int32 Inventory::GetFeatureInventoryModuleByIndex(int index, out string moduleName)*, (page 65).
property	Property name
[**out**] **propertyValue**	Inventory value from the database. The values for **property** are defined as: ModuleName Version

[**out**] **propertyString**	Inventory value from the database as string representation (with no special string representation identical to **propertyValue**)

2.8.11 Int32 Inventory::GetFeatureInventoryModuleInitialize(string adsMachine, out int hitCount)

Creates a result set of the wildcard search for the module name of a machine's feature inventory.

adsMachine	ADS path of the machine.
[**out**] **hitCount**	Count of all hits in the result set.

2.8.12 Int32 Inventory::GetFeatureInventoryModuleByIndex(int index, out string moduleName)

Retrieves a module name from the specified index of a machine's feature inventory. This function can be used to enumerate features of the inventory of a specific machine.

index	Index in hit list.
[**out**] **moduleName**	Module name from the specified index.

2.8.13 Int32 Inventory::GetFeatureInventoryModuleFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.9 API for Challenge/Response

2.9.1 Int32 CR::CheckRecoveryType(string adsMachine, out int isLogon)

Checks the recovery type for a machine (logon recovery or BitLocker).

adsMachine	ADS path for the machine from which the challenge was sent.
[**out**] **isLogon**	1 = logon recovery, 0 = BitLocker.

2.9.2 Int32 CR::GetChallengeFlags(string adsMachine, string challenge, out int challengeFlags)

Extracts the action flags from a challenge and returns them.

adsMachine	ADS path for the machine from which the challenge was sent.
challenge	Challenge received from user.
[**out**] **challengeFlags**	Flags contained in the challenge. Flags are defined as: 1 Boot SafeGuard Enterprise client without user logon requested. 2 Flush of SafeGuard Enterprise local cache expected. 4 Boot from external medium or floppy disk requested 8 Crypto token requested Flag combinations in **challengeFlags** are possible.

2.9.3 Int32 CR::CheckChallenge(string challenge, out int challengeErrorPart)

Checks the challenge and returns the state in **challengeErrorPart**.

challenge	Challenge received from user.
[**out**] **challengeErrorPart**	State of the challenge. The values for **challengeErrorPart** are defined as: 1 Error in the first part of the challenge (character 1 to 10). 2 Error in the second part of the challenge (character 11 to 20). 3 Error in the third part of the challenge (character 21 - 30). 9 Wrong challenge length.

2.9.4 Int32 CR::ComputeResponse(string adsUser, string adsMachine, string challenge, int action, out string response)

Calculates and returns a response for a given challenge.

adsUser	ADS path of the user requesting the response [optional].
adsMachine	ADS path of the machine on which the challenge was created.
challenge	Challenge string.
action	Flags defining which action will be performed on the client. Actions are defined as: 0 Boot SafeGuard Enterprise client with user logon (without showing user password). 1 Boot SafeGuard Enterprise client with user logon (showing user password). 2 Boot SafeGuard Enterprise client without user logon. 3 Allow booting from external medium.
[**out**] **response**	Response code string.

2.9.5 Int32 CR::SuspensionComputeResponse(string challenge, string dsnUser, uint suspensionTime, out int opCode, out string response)

Calculates a suspension reponse string and a suspension opCode for a user.

challenge	A 10-character challenge string.
dsnUser	The DSN user name (for example CN=john,CN=Users,DC=ps,DC=utimaco,DC=de).
suspensionTime	The suspension time in minutes.
[**out**] **opCode**	The suspension opCode (for example suspend Configuration Protection...).
[**out**] **response**	A 10-character response code.

2.9.6 Int32 CR::BitLockerRecovery(string adsMachine, string drive, out string response)

Calculates the response for BitLocker.

adsMachine	ADS path of the machine on which the challenge was created.
drive	Drive requesting the response, empty for BitLocker Boot.
[**out**] **response**	Response code string.

2.9.7 Int32 CR::OfflineCheckBAKFile(string bakFileName, out int isLogon, out string drive)

Checks the BAK file of an offline client (Logon recovery or BitLocker).

bakFileName	Path of the BAK file of the offline client.
[**out**] **isLogon**	1 = logon recovery, 0 = BitLocker.
[**out**] **drive**	Requesting the response for BitLocker drive, empty for BitLocker BOOT.

2.9.8 Int32 CR::OfflineGetChallengeFlags(string bakFileName, string challenge, out int challengeFlags)

Extracts the action flags from a challenge and returns them for an offline client.

bakFileName	Path of the BAK file of the offline client.
challenge	Challenge received from user
[**out**] **challengeFlags**	Flags contained in the challenge. Flags are defined as: 1 Boot SafeGuard Enterprise client without user logon requested. 2 Flush of SafeGuard Enterprise local cache expected. 4 Boot from external medium or floppy disk requested 8 Crypto token requested Flag combinations in **challengeFlags** are possible.

2.9.9 Int32 CR::OfflineComputeResponse(string bakFileName, string challenge, int action, out string response)

Calculates and returns a response for a given challenge for an offline client.

bakFileName	Path of the BAK file of the offline client.
challenge	Challenge string.
action	Flags defining which action will be performed on the client. Actions are defined as: 0 Boot SafeGuard Enterprise client with user logon (without showing user password). 1 Boot SafeGuard Enterprise client with user logon (showing user password). 2 Boot SafeGuard Enterprise client without user logon. 3 Allow booting from external medium.
[**out**] **response**	Response code string.

2.9.10 Int32 CR::OfflineBitLockerRecovery(string bakFileName, string drive, out string response)

Calculates the response for BitLocker for an offline client.

bakFileName	Path of the BAK file from the offline client.
drive	Drive requesting the response, empty for BitLocker BOOT.
[**out**] **response**	Response code string.

2.9.11 Int32 CR::GetVirtualClientInitialize(string searchName, out int hitCount)

Creates a result set of the wildcard search for virtual clients in the SafeGuard Enterprise database.

searchName	Search name of the virtual client.
[**out**] **hitCount**	The count of all hits in the result set.

2.9.12 Int32 CR::GetVirtualClientByIndex(int index, out string virtualClientName, out string virtualClientId)

Returns the virtual client at the specified index from the SafeGuard database.

index	Index of the certificate object, zero-based.
[**out**] **virtualClientName**	Name of the virtual client.
[**out**] **virtualClientId**	ID of the virtual client

2.9.13 Int32 CR::GetVirtualClientFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.9.14 Int32 CR::GetVirtualClientKeyFileInitialize(string searchName, out int hitCount)

Creates a result set of the wildcard search for virtual client key files in the SafeGuard Enterprise database.

searchName	Search name of the virtual client key files.
[**out**] **hitCount**	Count of all hits in the result set.

2.9.15 Int32 CR::GetVirtualClientKeyFileByIndex(int index, out string keyFileName, out string keyFileComment)

Returns the virtual client key file at the specified index from the SafeGuard Enterprise database.

index	Index of the certificate object, zero-based
[**out**] **keyFileName**	Name of the virtual client key file.
[**out**] **keyFileComment**	Comment of the virtual client key file.

2.9.16 Int32 CR::GetVirtualClientKeyFileFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

Note: You have to call the finalize function to be able to start a new search. Calling a new initialize function without calling the finalize function of the old wildcard search first, results in error code ACTION_NOT_FINALIZED.

2.9.17 Int32 CR::VirtualClientWithKeyRecovery(string virtualClientId, string virtualClientName, string keyId, string challenge, out string response)

Calculates and returns a response for a given challenge for virtual client with a key.

virtualClientId	ID of the virtual client.
virtualClientName	Name of the virtual client.
keyId	ID of the key.
challenge	Challenge string.
[**out**] **response**	Response code string.

2.9.18 Int32 CR::VirtualClientWithKeyFileRecovery(string virtualClientId, string virtualClientName, string keyFileId, string challenge, out string response)

Calculates and returns a response for a given challenge for a virtual client with a key file. Deletes the key file after sucessful creation of the response.

virtualClientId	ID of the virtual client.
virtualClientName	Name of the virtual client.
keyId	ID of the key.
challenge	Challenge string.
[**out**] **response**	Response code string.

2.10 API for reporting

2.10.1 Int32 DeleteAllEvents(string backupFilePathName)

Deletes all events currently stored in the central database.

backupFilePathName	Path and file name of the backup file (if exists, data will be appended). **Note:** If **backupFilePathName** is empty, no backup file will be written.

2.10.2 Int32 DeleteEventsLeaveLast(int numberOfEvents, int orderByLogTime, string backupFilePathName)

Deletes all events currently stored in the central database but leaves the latest n events in the database.

numberOfEvents	Number of events to retain (latest).
orderByLogTime	1 = Order by log time. 0 = Order by creation time.
backupFilePathName	Path and file name of the backup file (if exists, data will be appended). **HINT:** If **backupFilePathName** is empty, no backup file will be written.

2.10.3 Int32 DeleteEventsOlderThan(string date, int orderByLogTime, string backupFilePathName)

Deletes all events older than <date> from the database.

date	Timestamp (mm/dd/yyyy). All older events will be deleted.
orderByLogTime	1 = Order by log time. 0 = Order by creation time.
backupFilePathName	Path and file name of the backup file (if exists, data will be appended). **Note:** If **backupFilePathName** is empty, no backup file will be written.

2.10.4 Int32 CleanEventTable(int maxDuration, int maxCount, int keepBackup)

Deletes events from the EVENT table in the database.

Note: This method uses a stored procedure to delete the events. It is therefore essential that event chaining is switched off!

maxDuration	Events older than <maxDuration> days from now will be deleted. If zero (0) is provided, the parameter is ignored.
maxCount	The system will leave <maxCount> events in the database.
keepBackup	1 = Events will be deleted from the EVENT table, but a backup will be created th the EVENT_BACKUP table. The stored procedure will automatically create the EVENT_BACKUP table if it does not exist. 0 = Events are deleted without a backup. **Note:** The EVENT_BACKUP table can grow dramatically. Parameters **maxCount** and **maxDuration** can be combined. First **maxCount** will be checked, then **maxDuration**.

2.11 API for Misc

2.11.1 Int32 ResetCertificateStore(string password

Resets the CBI certificate store on the local machine

password	(New) password for the local certificate store.

Note: Be aware theat all certificates and all private keys will be removed from the local certificate store.

2.11.2 Int32 ExportCompanyCertificate(string p7Filename, string p12Filename, string p12Password, int overwriteExisting)

Retrieves the company certificate stored in the central database and exports it to two files:

- a password encrypted P12 file containing the public and private key
- a P7 file containing the public key only

p7Filename	Path and file name of the certificate (P7)
p12Filename	Path and file name of the key file
p12Password	Password for encrypting the P12 file
overwriteExisting	1 = Overwrite file(s) if they already exist. Otherwise, do not overwrite (will return an error)

Note: The currently logged on officer has to be a Master Security Officer to use this method.

2.11.3 Int32 AddP12ToCertStore(string certStoreName, string certStorePassword, string p12Filename, string password)

Adds the certificates(s) along with the private keys contained in the P12 provided to a specified certificate store.

certStoreName	Name of the certificate store (e.g., "MY")
certStorePassword	Password of the certificate store
p12Filename	Path and file name of the P12 file to be imported
password	Password of the P12 file to be imported

2.11.4 Int32 AddP7ToCertStore(string certStoreName, string p7Filename)

Adds a certificate to a specified certificate store.

certStoreName	Name of the certificate store (e.g., "MY")
p7Filename	Path and file name of the P7 file to be imported

2.11.5 Int32 ExportOfficerCertificate(string p7Filename, string p12Filename, string p12Password, int overwriteExisting)

Exports the certificate (P7) and the key file (P12) of the certificate that the officer is currently logged on with.

p7Filename	Path and file name of the certificate (P7)
p12Filename	Path and file name of the key file
p12Password	Password for encrypting the P12 file
overwriteExisting	1 = Overwrite file(s) if they already exist. Otherwise, do not overwrite (will return an error)

2.11.6 Int32 SignFileForLCImport(string toSignFileName, string signedFileName)

Signs a file with the company certificate to allow import to the SafeGuard Enterprise policy cache.

toSignFileName	Path and file name of the file to be signed.
signedFileName	Path and file name of the file to store the signed file in. A new file is created. If the file already exists, this will fail.

2.12 API for service accounts

2.12.1 Int32 CreateServiceAccountList(string listName)

Creates a service account list

listName	Name of the service account list

2.12.2 Int32 AddServiceAccountToServiceAccountList(string listName, string userName, string domain)

Adds a service account to a service account list.

listName	Name of the service account list
userName	Name of the service account
domain	Name of the domain

2.12.3 Int32 DeleteServiceAccount(string listName, string name, string domain)

Deletes a service account list.

listName	Name of the service account list

2.12.4 Int32 InitializeServiceAccountLists(out int hitCount)

Initializes the service account lists.

hitCount	The number of service account lists.

2.12.5 Int32 GetServiceAccountListsByIndex(int index, out string name)

Retrieves the service account lists from the result set generated by InitializeServiceAccountLists.

index	Index of the service account, zero-based
[**out**] **name**	Name of the service account.

2.12.6 GetServiceAccountMembersByIndex(string listName, int index, out string username,out string domain)

Retrieves the service account list members by index.

listName	Name of the service account list
index	The index to retrieve
[**out**] **username**	User name of the service account list member
[**out**] **domain**	Domain of the service account list member

The index starts with 0.

2.12.7 Int32 GetServiceAccountMembersCount(string listName,out int count)

Supplies a count of the members associated with a service account list.

listName	Name of the list to inquire
count	The number of members of the relevant service account list.

2.12.8 Int32 RemoveServiceAccountFromList(string listName,string userName, string domain)

Removes a service account from a service account list.

listName	Name of the service account list
userName	Name of the service account
domain	Name of the domain

2.12.9 Int32 RenameServiceAccount (string listName,string oldName, string newName, string oldDomain, string newDomain)

Renames a service account.

listName	Name of the service account list
oldName	The old name of the service account
newName	The new name of the service account
oldDomain	The old domain
newDomain	The new domain

2.12.10 Int32 RenameServiceAccountList(string listName,string newListName)

Renames a service account list.

listName	The old service account list name
newListName	The new service account list name

2.12.11 Int32 ServiceAccountExists(string listName, string name, string domain, out int exists)

Checks whether a service account exists or not.

listName	Name of the service account list
name	The name of the service account that is to be checked.
domain	The domain of the service account
[**out**] **exists**	1 = The service account exists. 0 = The service account does not exist.

2.12.12 Int32 ServiceAccountListExists(string listName, out int exists)

Checks whether a service account list exists or not.

listName	The name of the service account list that is to be checked.
[**out**] **exists**	1 = The service account list exists. 0 = The service account list does not exist.

2.13 API for Local Self Help

2.13.1 Int32 LocalSelfHelp::GetLocalSelfHelpParameter(out int minCountAnswers, out int drawnCountQuestions)

Gets Local Self Help parameters from the database.

[**out**] **minCountAnswers**	The mimimum number of answers a user must to enter to activate Local Self Help.
[**out**] **drawnCountQuestions**	The number of questions a user must answer correctly to log on at the POA.

2.13.2 Int32 LocalSelfHelp::SetLocalSelfHelpParameter(int minCountAnswers, int drawnCountQuestions)

Writes Local Self Help parameters to the database

[**out**] **minCountAnswers**	The mimimum number of answers a user must to enter to activate Local Self Help.
[**out**] **drawnCountQuestions**	The number of questions a user must answer correctly to log on at the POA.

2.14 API for POA group to container assignment

2.14.1 Int32 GroupContainerAssignment::AssignPOAGroupToContainer(string dsnContainerObject, string dsnPOAGroup)

Adds a POA group to the assigment list of a container.

dsnContainerObject:	Distinguished name of the container
dsnPOAGroup	Distinguished name of the group

2.14.2 Int32 GroupContainerAssignment::UnassignPOAGroupToContainer(string dsnContainerObject, string dsnPOAGroup)

Removes a POA group from the assignment list of a container.

dsnContainerObject:	Distinguished name of the container
dsnPOAGroup	Distinguished name of the group

2.14.3 Int32 GroupContainerAssignment::GetAssignedPoaGroupInitialize(string dsnContainerObject, out int hitCount)

Initializes a list of all groups assigned to a container.

dsnContainerObject:	Distinguished name of the container
[**out**] **hitCount**	Count of all hits in the result set

2.14.4 Int32 GroupContainerAssignment::GetAssignedPoaGroupByIndex(int index, out string dsnPOAGroup)

Retrieves the group assigned to the container form the result set. This method can be used to enumerate the groups assigned to a container.

dsnContainerObject:	Distinguished name of the container
[**out**] **hitCount**	Count of all hits in the result set

2.14.5 Int32 GroupContainerAssignment::GetAssignedPoaGroupFinalize()

Finalizes the wildcard search and deletes the result set. A new search can be started.

2.15 Error codes returned by the API methods

`OBJECT_OWN_MEMBER = -15`	Object cannot be its own member. For example: a group cannot be its own member.
`ACTION_NOT_FINALIZED = -14`	Action (e.g. wildcard search) not finalized.
`ACTION_NOT_INITIALIZED = -13`	Action (e.g. wildcard search) not initialized.
`RESULT_NOT_UNIQUE = -12`	Result set is not unique.
`INVALID_CHALLENGE_CODE = -11`	Wrong challenge code entered for Challenge/ Response.
`NO_MORE_DATA = -10`	End of data in any wildcard search method.
`INSUFFICIENT_RIGHTS = -9`	Current Security Officer has insufficient rights.
`CONFIG_FILE_ERROR = -8`	.conf file could not be found or is invalid.
`TOKEN_INVALID_SLOT = -7`	Invalid token slot ID.
`NOT_AUTHENTICATED = -6`	Security Officer has not authenticated.
`OBJECT_NOT_FOUND = -5`	Object not found in the database.
`OBJECT_ALREADY_EXISTS = -4`	Object already exists.
`TOKEN_NOT_PRESENT = -3`	No token in the slot.
`NOT_INITIALIZED = -2`	API was not initialized.
`FAILURE = -1`	General failure.
`OK = 0`	Success.

Note: As soon as an error is returned by the API, the user can call `GetLastError` to receive a more detailed error message along with the internal error code. Note that the error message string will be localized.

2.16 Additional log events

Additional log events have been defined. The events are logged when the Scripting API is used:

- ADMIN_SCRIPTING_AUTHENTICATION_SUCCESS (OfficerName)
- ADMIN_SCRIPTING_AUTHENTICATION_FAILED (OfficerName, reason)
- ADMIN_SCRIPTING_DOMAIN_INIT (OfficerName)

2.17 Format of the log file created during synchronization

Basically, the log file created during synchronization will be written in comma-separated values (*.csv).

2.18 Sample scripts

The SafeGuard Enterprise Administration Product CD contains sample scripts for all main areas. You will find the sample scripts in directory \samples on this CD. The following sample scripts are available:

Area	Sample script file
Synchronization	Synchronize.vbs
Users & Computers management	AddObjectToSGN.vbs
User-computer assignment (UMA)	UmaAPI.vbs
Key generation and assignment	KeyGenerationAssignmentAPI.vbs
Certificate assignment	CertificatesAPI.vbs
Token management	TokenSampleScript.vbs
Inventory and status information	InventoryAPI.vbs
Challenge/Response	ChallengeResponseByScript.vbs
Reporting	ReportsAPI.vbs

For running WSH scripts you should always use Cscript.exe Interpreter (by default scripts are run using Wscript.exe). Using Cscript Interpreter, the WSH script is run within a command shell (CMD). This avoids for example a message box being displayed at Wscript echo output, which you have to close by mouse click. Furthermore, using Cscript facilitates the detection of programming errors. You can cancel scripts simply by closing the command shell or pressing Ctrl + C. Cscript Interpreter also facilitates running simple scripts (e.g. Schedule) automatically at a specified point in time.

Following is an example for running scripts via Cscript:

1. Select **Start** > **Run** and enter command `cmd`.
2. In the window displayed, use command `cd` (change directory) to change to the directory, where the WHS is stored.
3. Call the script using command `cscript` (e.g. `cscript Synchronize.vbs`).

3 Installation/environment

A script which is executed in a user context can be run on the Security Officer's machine, where the SafeGuard Enterprise Management Center is installed. The Management Center has to be configured (Initial Configuration Wizard) and operable.

When a script is executed unattended on a SafeGuard Enterprise Server, this requires an activated SafeGuard Enterprise Server (Server Package installed). However, the Web Service itself can be deactivated after it has been tested for successful operation.

The API DLL is called "Utimaco.SafeGuard.AdministrationConsole.Scripting.dll". It is installed in the .NET Global Assembly Cache (GAC) along with the Management Center Setup.

4 Technical Support

You can find technical support for Sophos products in any of these ways:

- Visit the SophosTalk forum at *http://community.sophos.com/* and search for other users who are experiencing the same problem.
- Visit the Sophos support knowledgebase at *http://www.sophos.com/support/*
- Download the product documentation at *http://www.sophos.com/support/docs/*
- Send an email to *support@sophos.com*, including your Sophos software version number(s), operating system(s) and patch level(s), and the text of any error messages.

5 Legal notices

Copyright © 1996 - 2011 Sophos Group. All rights reserved. SafeGuard is a registered trademark of Sophos Group.

Sophos is a registered trademark of Sophos Limited, Sophos Group and Utimaco Safeware AG, as applicable. All other product and company names mentioned are trademarks or registered trademarks of their respective owners.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise unless you are either a valid licensee where the documentation can be reproduced in accordance with the licence terms or you otherwise have the prior permission in writing of the copyright owner.

You find copyright information on third party suppliers in the file entitled Disclaimer and Copyright for 3rd Party Software.rtf in your product delivery.