EVault Software
Vault API 7.0
User Guide

Revision: This manual has been updated for Version 7.01.

Software Version: 7.01 (September 2012)

EVault, A Seagate Company
c/o Corporation Trust Center
1209 Orange Street
Wilmington, New Castle
Delaware 19801
www.evault.com

Acknowledgements: Two encryption methods, DES and TripleDES, include cryptographic software written by Eric Young. The Windows versions of these algorithms also include software written by Tim Hudson. Bruce Schneier designed Blowfish encryption.

The EVault Software Agent, EVault Software CentralControl, and EVault Software Director applications have the encryption option of AES (Advanced Encryption Standard). Advanced Encryption Standard algorithm (named Rijndael, pronounced “Rain Doll”) was developed by cryptographers Dr. Joan Daemen and Dr. Vincent Rijmen. This algorithm was chosen by the National Institute of Standards and Technology (NIST) of the U.S. Department of Commerce to be the new Federal Information Processing Standard (FIPS).

The EVault Software Agents and EVault Software Director applications also have the added security feature of an over the wire encryption method.

Contents

1 Introduction

This Application Programming Interface (API) guide is intended for Integrators who use Component Object Model (COM) interface technology to create, delete, update, and enumerate customer, location, computer, task, account, and user metadata in a managed vault environment. This guide assumes an intermediate knowledge of EVault Software Director operation and administration.

The COM objects and Director Management Console use the same Vault System Management Protocol (VSMP) to communicate with the vault. After establishing communication with the vault, VSMP creates (with a single call), the customer, location, account, and user hierarchy, and assigns the default WORK (for legacy Vaults) and RAID storage locations.

You can use COM objects and the VSMP protocol to complete these tasks:

- Assign customer plug-in quota
- Create a customer, location, account, user, or location code
- Create an additional user for the same customer, location, and account
- Create a new satellite and generate OTRK
- Create or retrieve a satellite vault configuration
- Verify uniqueness for customer short name and billing code
- Enumerate customer, location, computer, task, and safeset data on a vault and return their properties
- Delete a customer, location, computer, task, or safeset
- Retrieve or remove computer licenses
- Obtain vault properties

Vault API 7.0 is compatible with Director versions 6.22 and later. However, some functionality in Vault API 7.0 is not available in Director versions prior to 7.0.

1.1 What's New in Version 7.01?

- Methods for getting and setting replication and maintenance services have been added to the CManager class. For more information, see getMaintenanceStatus, setMaintenanceStatus, getReplServiceStatus and setReplServiceStatus.
- Billing codes can now range from 5-20 characters in length. Previously, billing codes could only be 5 characters in length. For more information, see CLocation.

1.2 What's New in Version 7.0?

- The following changes have been made to the CAccount class:
    - The OperatingMode property now returns an HR INVALID GETOPMODE/SETOPMODE error if the vault has more than one replication role.
    - A new getOperatingMode method gets a vault's replication role if the vault has more than one replication role.
    - A new setOperatingMode method sets a vault's replication role if the vault has more than one replication role.

    Note: A vault that is version 7.0 or later can have more than one replication role (e.g., both Active and Base).

    - A new Init method initializes an account object. This method is called when a CAccount object is created outside of Vault API.

    For more information, see CAccount.
- The following changes have been made to the CTask class:
    - The OperatingMode property now returns an HR INVALID GETOPMODE/SETOPMODE error if the vault has more than one replication role.
    - A new getOperatingMode method gets a vault's replication role if the vault has more than one replication role.
    - A new setOperatingMode method sets a vault's replication role if the vault has more than one replication role.

    Note: A vault that is version 7.0 or later can have more than one replication role (e.g., both Active and Base).

    For more information, see CTask.
- The following new properties were defined in the CVault class: ReplicationType, ActiveRoleState, PassiveRoleState, SatelliteRoleState, BAVRoleState. For more information, see CVault.
- A CStorageLocation class has been added. For more information, see CStorageLocation.
- Methods for managing primary storage locations have been added to the CManager class. For more information, see setOnlineStorageLocation, removeOnlineStorageLocation, getOnlineStorageLocation, getOnlineStorageLocations, testUNCCustomCredentials and testUNCDefaultCredentials.
- Methods for activating, getting and setting licenses on a vault have been added to the CManager class. For more information, see activateLicenses, getLicenseList and setLicenseList.
- VaultReplicationType Enum now has a Replication_Unknown value. For more information, see VaultReplicationType enum.

1.3 Installing COM Objects

As a prerequisite, you must install Microsoft .NET Framework 4.0.

You can install the COM objects locally on the vault, or remotely. If you are installing the COM objects remotely, use a secure VPN connection.

To install the COM objects:

1. Double-click the VaultAPISetup.exe file. To obtain the self-extracting installation file, contact your licensed service provider of EVault Software products, or visit http://www.evault.com/.
2. Complete the **EVault Software Vault API Setup** wizard.
3. Click **Finish**.

1.4 Upgrading COM Objects

You can upgrade the COM objects locally on the vault, or remotely. If you are upgrading the COM objects remotely, use a secure VPN connection.

To upgrade the COM objects:

1. Double-click the VaultAPISetup.exe file. To obtain the self-extracting installation file, contact your licensed service provider of EVault Software products, or visit http://www.evault.com/.
2. Select **Upgrade**.
3. Click **Next**.

1.5 Uninstalling COM Objects

To uninstall the COM objects:

1. Double-click the VaultAPISetup.exe file. To obtain the self-extracting installation file, contact your licensed service provider of EVault Software products, or visit http://www.evault.com/.
2. Select **Uninstall**.
3. Click **OK**.
4. Click **Finish**.

1.6 Running the Test Application

You can run a C# based test application to test the COM objects. The test application stores its log files in the c:\temp directory.

To run the test application:

Double-click the AccountServiceTestApp.exe file. To obtain this self-extracting file, contact your licensed service provider of EVault Software products, or visit http://www.evault.com/.

2 EVault COM Objects

This section describes the EVault COM objects you can implement.

2.1 CVaultConnection

Implements the following interface:

CVaultConnection

↑

IVaultConnection

If you are connecting to a version 6.21 or later vault, you can use Windows authentication instead of user credentials.

This object represents a vault:

- **UserName** – account name to be used when connecting to the vault
- **Password** – account password to be used when connecting to the vault
- **Domain** – account domain to be used when connecting to the vault
- **Description** – (optional) description on the vault connection
- **Address** – network address of the vault (Make sure if you are using a vault name instead of the IP address that there is name resolution for the name.)
- **Port** – (optional) port to be used to connect to the vault
- **DefaultWorkArea** – for legacy vaults, work area name to be used when creating account and user (e.g., VV_WORK1)
- **DefaultRaidArea** – raid area name to be used when creating account and user (e.g., SG01)

2.2 CManager

Implements the following interfaces:

CManager

↑

IManager5

↑

IManager4

↑

IManager3

↑

IManager2

↑

IManager

Properties

- **LogFileName** – The name of the file in an existing log directory where the COM object logs important activities. For example, C:\VaultCOM\Logs\Execution.log.
- **AuditFileName** – The name of the file in an existing log directory where the COM object logs all successfully created customers. For example, C:\VaultCOM\Logs\Audit.csv.
- **ReqestId** – A string that associates the external application call with a customer in the audit log.

2.2.1 billingCodeExists

Determines if the billing code already exists on the vault.

Syntax

```
HRESULT billingCodeExists(IDispatch* vaultInfo, [in] BSTR billingCode,
[out] VARIANT_BOOL* codeExists);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **billingCode** – The billing code to be checked

Remarks

The call returns:

- **codeExists** – true means the billing code exists
- **codeExists** – false means the billing code does not exist

2.2.2 GetCustomerQuota

Returns the quota for a specific customer and type.

Syntax

```
HRESULT GetCustomerQuota([in]IDispatch* vaultInfo, [in]IDispatch*
customer, [in]BSTR quotaType, [out, retval] ULONG* quotaValue);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **customer** – CCustomer object
- **quotaType** – The type of quota

Remarks

The call returns:

- **quotaValue** – The quota for a specific customer and type

2.2.3 SetCustomerQuota

Sets a new quota for a specific customer and type.

Syntax

```
HRESULT SetCustomerQuota([in]IDispatch* vaultInfo, [in]IDispatch*
customer, [in]BSTR quotaType, [in] ULONG quotaValue);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **customer** – CCustomer object
- **quotaType** – The type of quota
- **quotaValue** – The quota for a specific customer and type

2.2.4 Create

Creates a customer, location, account, and user.

Syntax

```
HRESULT Create(IDispatch* vaultInfo, IDispatch* customer, IDispatch*
location, IDispatch* account, IDispatch* user);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **customer** – CCustomer object
- **location** – CLocation object
- **account** – CAccount object
- **user** – CUser object

2.2.5 CreateUser

Creates a user in a specific account.

Syntax

```
HRESULT CreateUser(IDispatch* vaultInfo, IDispatch* account, IDispatch*
user);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **account** – CAccount object to specify the account
- **user** – CUser object

2.2.6 AddCustomerQuota

Adds a quota of a specific type to a specific customer.

Syntax

```
HRESULT AddCustomerQuota([in]IDispatch* vaultInfo, [in]IDispatch*
customer, [in]BSTR quotaType, [in] ULONG quotaValue);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected

- **customer** – CCustomer object to specify the customer
- **quotaType** – quota type to be added
- **quotaValue** – The quota for a specific customer and type

2.2.7 customerExists

Checks if a customer exists based on the short name.

Syntax

```
HRESULT customerExists(IDispatch* vaultInfo, [in] BSTR
customerShortName, [out] VARIANT_BOOL* exists);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **customerShortName** – The short name of the customer to be checked

Remarks

The call returns:

- **exists** – true means the customer exists
- **exists** – false means the customer does not exist

2.2.8 accountAndUserExists

Checks if a customer and account exist based on the short name.

Syntax

```
HRESULT (IDispatch* vaultInfo, [in] BSTR accountName, [in] BSTR
userName, [out] VARIANT_BOOL* exists);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **accountName** – The name of the account to check
- **userName** – The user name in the specified account

Remarks

The call returns:

- **exists** – true means the account and user exist
- **exists** – false means the account or user does not exist

2.2.9 enableObject

Enables or disables the object that is associated with the object ID.

Syntax

```
HRESULT enableObject([in]IDispatch* vaultInfo, [in] OBJECT_TYPE_ENUM
objectType, [in] ULONG objectId, [in] VARIANT_BOOL enable);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **objectType** – The type of object. Only account and user are supported.
- **objectId** – The unique ID for the object
- **enable** – true means enable, false means disable

2.2.10 getCustomerList

Returns a list of vault customers.

Syntax

```
HRESULT getCustomerList([in]IDispatch* vaultInfo, [out, retval]
VARIANT* customers);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected

Remarks

The call returns:

- **customers** – a collection of CCustomer objects

2.2.11 getLocationList

Returns all the CLocation objects for a customer.

Syntax

```
HRESULT getLocationList([in] IDispatch* vaultInfo, [in] ULONG
customerId, [out, retval] ICollection** locList);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **customerId** – Identifies the unique customer ID

Remarks

The call returns:

- **locList** – a collection of CLocation objects

2.2.12 getAccountList

Returns all CAccount objects for a location.

Syntax

```
HRESULT getAccountList([in] IDispatch* vaultInfo, [in] ULONG locationId,
[out, retval] ICollection** acctList);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **locationId** – Identifies a unique location

Remarks

The call returns:

- **acctList** – a collection of CAccount objects

2.2.13 getUserList

Returns all CUser objects for an account.

Syntax

```
HRESULT getUserList([in] IDispatch* vaultInfo, [in] ULONG accountId,
[out, retval] ICollection** userList);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **accountId** – Identifies a unique location

Remarks

The call returns:

- **userList** – a collection of CUser objects

2.2.14 getComputerList

Returns all CComputer objects for a location.

Syntax

```
HRESULT getComputerList([in] IDispatch* vaultInfo, [in] ULONG
locationId, [out, retval] ICollection** computerList);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **locationId** – Identifies a unique location

Remarks

The call returns:

- **computerList** – a collection of CComputer objects

2.2.15 getTaskList

Returns all the CTask objects for a computer.

Syntax

```
HRESULT getTaskList([in] IDispatch* vaultInfo, [in] ULONG computerId,
[out, retval] ICollection** taskList);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **computerId** – Identifies a unique computer

Remarks

The call returns:

- **taskList** – a collection of CTask objects

2.2.16 getSafesetList

Returns all CSafeset objects for a task.

Syntax

```
HRESULT getSafesetList([in] IDispatch* vaultInfo, [in] ULONG taskId,
[out, retval] ICollection** safesetList);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **taskId** – Identifies a unique task

Remarks

The call returns:

- **safesetList** – a collection of CSafeset objects

2.2.17 getBillingCodeList

Returns all the CBillingCode objects for a customer.

Syntax

```
HRESULT getBillingCodeList([in] IDispatch* vaultInfo, [in] ULONG
customerId, [out, retval] ICollection** codeList);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **customerId** – Identifies a unique customer

Remarks

The call returns:

- **codeList** – a collection of CBillingCode objects

2.2.18 deleteSafeSet

Deletes a safeset.

Syntax

```
HRESULT deleteSafeSet([in] IDispatch* vaultInfo, [in] ULONG taskId, [in]
ULONG synchNum);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **taskId** – Identifies the unique task to which the safeset belongs
- **synchNum** – Identifies a unique safeset under a task

2.2.19 deleteComputer

Deletes a computer under a location.

Syntax

```
HRESULT deleteComputer([in] IDispatch* vaultInfo, [in] ULONG parentId,
[in] ULONG objectId);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **parentId** – Identifies a unique location for the computer. You can use 0 if the vault version is newer than 5.2.
- **objectId** – Identifies a unique computer under a location

2.2.20 deleteTask

Deletes a task under a computer.

Syntax

```
HRESULT deleteTask([in] IDispatch* vaultInfo, [in] ULONG parentId, [in]
ULONG objectId);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **parentId** – Identifies a unique location for the computer. You can use 0 if the vault version is newer than 5.2.
- **objectId** – Identifies a unique task under a computer

2.2.21 deleteAccount

Deletes an account under a location.

Syntax

```
HRESULT deleteAccount([in] IDispatch* vaultInfo, [in] ULONG parentId,
[in] ULONG objectId);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **parentId** – Identifies a unique location for the account. You can use 0 if the vault version is newer than 5.2.
- **objectId** – Identifies a unique account under a location

2.2.22 deleteUser

Deletes a user under an account.

Syntax

```
HRESULT deleteUser([in] IDispatch* vaultInfo, [in] ULONG parentId, [in]
ULONG objectId);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **parentId** – Identifies a unique account for the user. You can use 0 if the vault version is newer than 5.2.
- **objectId** – Identifies a unique user under an account

2.2.23 deleteCustomer

Deletes a customer.

Syntax

```
HRESULT deleteCustomer([in] IDispatch* vaultInfo, [in] ULONG objectId);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **objectId** – Identifies a unique customer

2.2.24 deleteLocation

Deletes a location under a customer.

Syntax

```
HRESULT deleteLocation([in] IDispatch* vaultInfo, [in] ULONG parentId,
[in] ULONG objectId);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **parentId** – Identifies a unique customer for the location. You can use 0 if the vault version is newer than 5.2.
- **objectId** – Identifies a unique location under a customer

2.2.25 deleteBillingCode

Deletes a billing code.

Syntax

```
HRESULT deleteBillingCode([in]IDispatch* vaultInfo, [in]ULONG objectId);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **objectId** – Identifies a unique billing code

2.2.26 getCustomerQuotaList

Returns all quota types for a customer.

Syntax

```
HRESULT getCustomerQuotaList([in]IDispatch* vaultInfo, [in]IDispatch*
customer, [out, retval] SAFEARRAY** quotaTypes);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **customer** – CCustomer object, identifies a customer

Remarks

The call returns:

- **quotaTypes** – an array of quota types

2.2.27 getSecondayGroupList

Returns all secondary storage groups for a vault.

Syntax

```
HRESULT getSecondayGroupList([in]IDispatch* vaultInfo,[out, retval)]
SAFEARRAY** secondaryGroups);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected

Remarks

The call returns:

- **secondaryGroups** – an array of secondary storage group names

2.2.28 getArchiveGroupList

Returns all archive storage groups for a vault.

Syntax

```
HRESULT getArchiveGroupList([in]IDispatch* vaultInfo, [out, retval]
SAFEARRAY** archiveGroups);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected

Remarks

The call returns:

- **archiveGroups** – an array of archive storage group names

2.2.29 getOnlineGroupList

Returns all primary storage groups for a vault.

Syntax

```
HRESULT getOnlineGroupList([in]IDispatch* vaultInfo,[out, retval]
SAFEARRAY** primaryGroups);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected

Remarks

The call returns:

- **primaryGroups** – an array of primary storage group names

2.2.30 getWorkareaGroupList

Returns all work area storage groups for a vault. This syntax can only be used on EVault Software Director version 6.04 or older.

Syntax

```
HRESULT getWorkareaGroupList([in]IDispatch* vaultInfo, [out, retval]
SAFEARRAY** workareaGroups);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected

Remarks

The call returns:

- **workareaGroups** – an array of work area group names

2.2.31 getSatelliteQuotaList

Returns all the unused satellite licenses classified in storage quota type on a Base vault.

Syntax

```
HRESULT getSatelliteQuotaList([in]IDispatch* vaultInfo, [out, retval]
ICollection** quotaList);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the Base vault to be connected
- **quotaList** – A collection of storage quota objects. Storage quota object include the storage quota type, and unused license count. Storage quota type is a string. For example, "REPLICATIONSV", "REPLICATIONSV250", or "REPLICATIONSV500". The suffix number indicates the storage quota in GB on satellite. If you do not specify a number, it means "Unlimited".

2.2.32 createSatellite

Creates a satellite vault on the Base vault and returns an OTRK key.

Syntax

```
HRESULT createSatellite([in]IDispatch* vaultInfo, [in]BSTR
customerShortName, [in]BSTR quotaType, [in]ULONG otrkExpiryHours,
[in]SatelliteMode satelliteMode, [out, retval]BSTR* otrk);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the Base vault to be connected
- **customerShortName** – The customer short name
- **quotaType** – The name of quota from getSatelliteQuotaList
- **otrkExpiryHours** – The OTRK expiry time in hours
- **satelliteMode** – satellite mode (enum SatelliteMode)

Remarks

The call returns:

- **otrk** – The OTRK key for the satellite vault

2.2.33 getSatelliteConfiguration

Retrieves satellite configuration information from the Base vault or satellite vault.

Syntax

```
HRESULT getSatelliteConfiguration([in]IDispatch* vaultInfo, [in]ULONG
customerId, [out, retval] IDispatch** satelliteConfig);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the Base vault to be connected
- **customerId** – The customer ID which has associated satellite. If the call is executed from the satellite, this parameter is ignored.

Remarks

The call returns:

- **satelliteConfig** – CReplicationSatelliteConfig object

2.2.34 deleteSatellite

Deletes satellite Vaults from the Base vault.

Syntax

```
HRESULT deleteSatellite([in]IDispatch* vaultInfo, [in]UINT satelliteId);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the Base vault to be connected
- **satelliteId** – The ID number of the satellite vault

2.2.35 getTaskPath

Finds all paths used by given task.

Syntax

```
HRESULT getTaskPath([in] IDispatch* vaultInfo, [in] ULONG taskId, [out,
retval] ICollection** taskPath);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the Base vault to be connected
- **taskId** – ID of the task

Remarks

The call returns:

- **taskPath** – a collection of paths for a task

2.2.36 getTaskByGuid

Finds the task associated with a specific globally unique identifier (GUID).

Syntax

```
HRESULT getTaskByGuid([in] IDispatch* vaultInfo, [in] BSTR taskGuid,
[out, retval] IDispatch** taskObject);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the Base vault to be connected
- **taskGuid** – GUID of the task

Remarks

The call returns:

- **taskObject** – CTask object

2.2.37 getComputerByGuid

Finds the computer associated with a specific globally unique identifier (GUID).

Syntax

```
HRESULT getComputerByGuid([in] IDispatch* vaultInfo, [in] BSTR
computerGuid, [out, retval] IDispatch** computerObject);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the Base vault to be connected
- **computerGuid** – GUID of the computer

Remarks

The call returns:

- **computerObject** – CComputer object

2.2.38 getComputerLicenseList

Returns the license list claimed on a computer with specific ID.

Syntax

```
HRESULT getComputerLicenseList([in] IDispatch* vaultInfo, [in] ULONG
computerId,[out, retval,satype(BSTR)] SAFEARRAY** licenses);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **computerId** – The ID number of the computer

Remarks

The call returns:

- **Licenses** – an array of license types claimed by a specific computer

2.2.39 removeComputerLicenseList

Removes the specified license list claimed on a computer with the specified ID.

Syntax

```
HRESULT removeComputerLicenseList([in] IDispatch* vaultInfo, [in] ULONG
computerId, [in,satype(BSTR)] SAFEARRAY* Licenses);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **computerId** – The ID of computer

Remarks

The call returns:

- **Licenses** – An array of license types that will be removed from a given computer

2.2.40 getFirstVault

Returns vault records for the vault with the lowest Vault ID.

Syntax

```
HRESULT getFirstVault([in] IDispatch* vaultInfo, [out, retval]
IDispatch** vaultObject);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected

Remarks

The call returns:

- **vaultObject** – CVault object

2.2.41 getCustomerByShortName

Returns the customer based on the short name provided.

Syntax

```
HRESULT getCustomerByShortName([in] IDispatch* vaultInfo, [in] BSTR
customerShortName, [out, retval] IDispatch** customerObject);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **customerShortName** – Short name of the customer

Remarks

The call returns:

- **customerObject** – CCustomer object

2.2.42 getAccountByName

Returns the account based on the name provided.

Syntax

```
HRESULT getAccountByName([in] IDispatch* vaultInfo, [in] BSTR
accountName, [out, retval] IDispatch** accountObject);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **accountName** – Name of the account

Remarks

The call returns:

- **accountObject** – CAccount object

2.2.43 getProcessList

Returns a list of vault processes.

Syntax

```
HRESULT getProcessList([in] IDispatch* vaultInfo, [out, retval]
ICollection** processList);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected

Remarks

The call returns:

- **processList** – List of CProcess objects

2.2.44 stopProcess

Stops a vault process. It does not wait until the vault process stops, but singles the process to stop (similar to JobMonitor).

Syntax

```
HRESULT stopProcess ([in] IDispatch* vaultInfo, ([in] IDispatch*
processToStop);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **processToStop** – CProcess object, indicates the process to be stopped

2.2.45 setOnlineStorageLocation

Adds or updates a primary storage location for a storage group.

Syntax

```
HRESULT setOnlineStorageLocation([in] IDispatch* vaultInfo, [in]
IDispatch* onlineLocation);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **onlineLocation** – CStorageLocation object

2.2.46 removeOnlineStorageLocation

Removes a primary storage location from a storage group.

Syntax

```
HRESULT removeOnlineStorageLocation([in] IDispatch* vaultInfo, [in]
BSTR groupName, [in] BSTR storageLocation);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **groupName** – storage group of the storage location
- **storageLocation** – path to the storage location

2.2.47 getOnlineStorageLocation

Returns all information about a primary storage location for a storage group.

Syntax

```
HRESULT getOnlineStorageLocation([in] IDispatch* vaultInfo, [in] BSTR
groupName, [in] BSTR storageLocation, [out, retval] IDispatch**
onlineLocation);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **groupName** – storage group of the storage location
- **storageLocation** – path to the storage location
- **onlineLocation** – CStorageLocation object

2.2.48 getOnlineStorageLocations

Returns a list of primary storage locations for a storage group.

Syntax

```
HRESULT getOnlineStorageLocations([in] IDispatch* vaultInfo, [in] BSTR
groupName, [out, retval] ICollection** onlineLocationList);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **groupName** – storage group of the storage locations
- **onlineLocationList** – list of CStorageLocation objects

2.2.49 testUNCCustomCredentials

Tests UNC custom credentials for a storage location.

Syntax

```
HRESULT testUNCCustomCredentials([in] IDispatch* vaultInfo, [in]
IDispatch* onlineLocation);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **onlineLocation** – CStorageLocation object

2.2.50 testUNCDefaultCredentials

Tests UNC default credentials for a storage location.

Syntax

```
HRESULT testUNCDefaultCredentials([in] IDispatch* vaultInfo, [in]
IDispatch* onlineLocation);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **onlineLocation** – CStorageLocation object

2.2.51 activateLicenses

Syntax

```
HRESULT activateLicenses([in] IDispatch* vaultInfo);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected

2.2.52 getLicenseList

Syntax

```
HRESULT getLicenseList([in] IDispatch* vaultInfo, [out, retval,
satype(BSTR)]SAFEARRAY** licensesList);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **licensesList** – list of licenses to get

2.2.53 setLicenseList

Syntax

```
HRESULT setLicenseList([in] IDispatch* vaultInfo, [in, satype(BSTR)]
SAFEARRAY* Licenses);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **Licenses** – list of licenses to set

2.2.54 getMaintenanceStatus

Syntax

```
HRESULT getMaintenanceStatus([in] IDispatch* vaultInfo, [out, retval]
VARIANT_BOOL* isEnabled);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **isEnabled** – true means enabled; false means disabled

2.2.55 setMaintenanceStatus

Syntax

```
HRESULT setMaintenanceStatus([in] IDispatch* vaultInfo, [in]
VARIANT_BOOL isEnabled);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **isEnabled** – true means enable; false means disable

2.2.56 getReplServiceStatus

Syntax

```
HRESULT getReplServiceStatus([in] IDispatch* vaultInfo, [in]
VaultReplicationType replMode, [out, retval] VARIANT_BOOL* isEnabled);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **replMode** – specifies the vault replication role (enum VaultReplicationType) for which the operating mode is read. Valid values are Replication_Active, Replication_Passive, Replication_Base_Vault, and Replication_Satellite_Vault.
- **isEnabled** – true means enabled; false means disabled

2.2.57 setReplServiceStatus

Syntax

```
HRESULT setReplServiceStatus([in] IDispatch* vaultInfo, [in]
VaultReplicationType replMode, [in] VARIANT_BOOL isEnabled);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **replMode** – specifies the vault replication role (enum VaultReplicationType) for which the operating mode is read. Valid values are Replication_Active, Replication_Passive, Replication_Base_Vault, and Replication_Satellite_Vault.
- **isEnabled** – true means enable; false means disable

2.3 CCustomer

Implements the following interface:

CCustomer

↑

ICustomer

Properties

- **Id** – a unique identifier for a customer, read only
- **Name** – customer name
- **ShortName** – customer short name, unique across the vault
- **Address**
- **City**
- **ZipCode**
- **State**
- **Country**
- **Phone**
- **Email**
- **Url**
- **ContactPerson**
- **Notes**

Remarks

Only **Name** and **ShortName** are required. The other general customer information fields are optional.

2.3.1 Update

Updates the customer's properties.

Syntax

```
HRESULT update([in]IDispatch* vaultInfo);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected

2.4 CLocation

Clocation implements the following interface:

CLocation

↑

ILocation

Properties

- **Id** – a unique identifier for a location, read only
- **Name** – location name, must be unique under a customer
- **BillingCode** –alphanumeric code that uniquely identifies the location. The code can range from 5-20 characters in length.
- **Address**
- **City**
- **ZipCode**
- **State**
- **Country**
- **Phone**
- **Email**
- **Url**
- **ContactPerson**
- **Note**
- **CustomerId**

Remarks

Only **Name** and **BillingCode** are required. The other general customer information fields are optional .

2.4.1 Update

Updates the properties for a location.

Syntax

```
HRESULT update([in]IDispatch* vaultInfo);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected

2.5 CAccount

CAccount implements these interfaces:

CAccount
↑
IAccount4
↑
IAccount3
↑
IAccount2
↑
IAccount

Properties

- **Id** – read only ; a unique identifier for an account
- **Name** – account name; must be unique for customer
- **Description** – (optional)
- **SecondaryGroup**
- **ArchivePath**
- **RaidPath** – read only
- **WorkPath** – read only; will be an empty string for a vault with version later than 6.04
- **OperatingMode** – get/set account's operating mode (enum OperatingModeType). It is used by newly created tasks. An error is returned if the vault has more than one replication role. If a vault has more than one replication role, use the getOperatingMode or setOperatingMode method.

  Note: A vault that is version 7.0 or later can have more than one replication role (e.g., both Active and Base).

- **IsActive** – read only; indicates whether or not the account is active

2.5.1 getOperatingMode

Gets the operating mode on an account, for the specified vault replication role.

Syntax

```
HRESULT getOperatingMode([in] VaultReplicationType replMode, [out,
retval] OperatingModeType* opMode);
```

Parameters

- **replMode** – specifies the vault replication role (enum VaultReplicationType) for which the operating mode is read
- **opMode** – indicates the operating mode for the account (enum OperatingModeType)

2.5.2 setOperatingMode

Sets the operating mode on an account, for the specified vault replication role.

Syntax

```
HRESULT setOperatingMode([in] VaultReplicationType replMode, [in]
OperatingModeType opMode);
```

Parameters

- **replMode** – specifies the vault replication role (enum VaultReplicationType) for which the operating mode is set
- **opMode** – specifies the operating mode for the account (enum OperatingModeType)

2.5.3 Update

Updates the properties for an account.

Syntax

```
HRESULT update([in]IDispatch* vaultInfo);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates a vault to be connected.

2.5.4 Init

Initializes an account object. This method should be called when a CAccount object is created outside of Vault API.

Syntax

HRESULT `init([in]IDispatch* vaultInfo);`

Parameters

- vaultInfo – CVaultConnection object, indicates a vault to be connected.

2.6 CUser

CUser implements the following interfaces:

CUser
↑
IUser2
↑
IUser

Properties

- **Id** – a unique identifier for a user, read only
- **Name** – (account and user name combination must be unique for the vault)
- **Password** – password for that user
- **SecondaryGroup**
- **ArchivePath**
- **RestrictionTypeId** – specify the access restriction: RESTRICTION_NONE, RESTRICTION_IP, RESTRICTION_NODE
- **RestrictionValue** – specify the value based on RestrictionTypeId
- **RaidPath** – read only
- **WorkPath** – read only, will be an empty string for vault with version later than 6.04
- **AccountID** – read only, indicates the account ID in which the user is created
- **IsActive** – read only, indicates whether or not the user is active

2.6.1 Update

Updates the properties of a user.

Syntax

```
HRESULT update([in]IDispatch* vaultInfo);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected

2.7 CComputer

CComputer implements the following interface:

CComputer

↑

IComputer

Properties

- **Id** – a unique identifier for a computer, read only
- **Name** – computer name, read only
- **IpAddress** – IP address of the computer, real only
- **Domain** – Network domain where computer is located, read only
- **sType** – OS Type, read only
- **OsVersion** – OS version, read only
- **AgentVersion** – Agent version installed on the computer, read only
- **AgentType** – Agent type installed on the computer
- **LocationId** – ID of the location, read only
- **GUID** – computer's GUID, read only

2.8 CTask

CTask implements these interfaces:

CTask

↑

ITask3

↑

ITask2

↑

ITask

Properties

- **Id** – a unique identifier for an task, read only
- **Name** – task name, read only
- **PoolFilesPath** – read only
- **IsActive** – read only, valid for vault with version earlier than 6.21
- **PoolVersion** – read only
- **PhysicalPoolSize** – read only
- **UsedPoolSize** – read only
- **SecondaryGroup** – secondary storage area
- **ArchivePath** – archive area
- **RaidPath** – read only
- **WorkPath** – read only; will be an empty string for a vault with version later than 6.04
- **OperatingMode** – get/set task's operating mode (enum OperatingModeType). Valid for vault versions later than 6.20. Beginning in version 7.0, an error is returned if the vault has more than one replication role. If the vault has more than one replication role, use the getOperatingMode or setOperatingMode method.

  Note: A version 7.0 or later vault can have more than one replication role (e.g., both Active and Base).
- **GUID** – task's GUID, read only
- **Status** – task's status (enum WorkingStatus). Valid for vault with version later than 6.20
- **ComputerId** – computer ID

2.8.1 getOperatingMode

Gets the operating mode on a task, for the specified vault replication role.

Syntax

```
HRESULT getOperatingMode([in] VaultReplicationType replMode, [out,
retval] OperatingModeType* opMode);
```

Parameters

- **replMode** – specifies the vault replication role (enum VaultReplicationType) for which the operating mode is read
- **opMode** – indicates the operating mode for the task (enum OperatingModeType)

2.8.2 setOperatingMode

Sets the operating mode on a task, for the specified vault replication role.

Syntax

```
HRESULT setOperatingMode([in] VaultReplicationType replMode, [in]
OperatingModeType opMode);
```

Parameters

- **replMode** – specifies the vault replication role (enum VaultReplicationType) for which the operating mode is set
- **opMode** – specifies the operating mode for the task (enum OperatingModeType)

2.8.3 Update

Updates the properties of a task.

Syntax

```
HRESULT update([in]IDispatch* vaultInfo);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected

2.9 CSafeset

Implements the following interface:

CSafeset

↑

ISafeset

Properties

- **synchNum** – Backup synchronization number, read only
- **BackupTime** – backup time, read only
- **RetentionOption** – Retention options
- **IsActive** – read only
- **Peripheral** – SIDF Media Type(b1)/Density(b2)
- **SynchCmp** – Synch number of parent backup
- **SerialNumber** – Unique serial number, read only
- **OriginalBytes** – Original size, read only
- **StorageBytes** – Storage bytes, read only
- **TotalComprBytes** – Compressed size (before delta) , read only
- **DeltaComprBytes** – Delta size (including compression) , read only
- **FileName** – read only
- **IsOnline** – read only
- **IsReplicated** – read only

2.9.1 Update

Updates the properties for a safeset.

Syntax

```
HRESULT update([in]IDispatch* vaultInfo ,[in]ULONG parentId);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected
- **parentID** – Task ID for the task to which the safeset belongs

2.10 CReplicationSatelliteConfig

Implements the following interface:

CReplicationSatelliteConfig

↑

IReplicationSatelliteConfig

Properties

- **Id** – satellite ID, read only
- **CustomerId** – customer associated with this satellite, read only
- **OTRK** – otrk key, read only
- **OTRKExpiry** – get/set expiration of OTRK
- **LastHeartbeat** – last heartbeat time, read only
- **HeartbeatInterval** – get/set heartbeat interval in minutes
- **BackupOption** – get/set satellite operating mode
- **SafesetPolicy** – safeset's policy (-1 – all safesets, otherwise days x 24)
- **RetentionDays** – retention days (-1 – all days)
- **RetentionCopies** – retention copies
- **DateInstalled** – read only
- **DateUpgraded** – read only
- **DisableReplication** – disable/enable replication
- **RetryCount** – read only
- **RetryTimeoutMin** – read only
- **TransferProtocol** – replication protocol name, read only
- **UseDataEncryption** – get/set encryption boolean flag
- **DisableThrottling** – is throttling disabled
- **ThrottleLimit** – throttling limit (Mbits/s)
- **UseThrottlingAllDay** – to use throttling 24/7
- **StartThrottleTime** – start throttling time (when UseThrottlingAllDay = false)
- **EndThrottleTime** – end throttling time (when UseThrottlingAllDay = false)

- **ThrottleDays** – throttling days of the week, bit mask (0 – Sunday)
- **WarmReplication** – warm replication (if true, replicate data as soon as possible)
- **StorageQuota** – storage quota
- **AllowSatelliteSettings** – to allow satellite to change its settings
- **SatelliteMode** – satellite mode (enum SatelliteMode)
- **ComputerName** – The name of computer which the satellite vault is running on (read only)
- **DomainName** – The domain of the computer which the satellite is running on (read only)
- **HardwareTag** – Hardware tag of the computer which the satellite is running on (read only)
- **Guid** – GUID of the satellite vault (read only)
- **BavGuid** – GUID of the BAV (read only)

2.10.1 Update

Updates the configuration of a satellite.

Syntax

```
HRESULT update([in]IDispatch* vaultInfo);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected

2.11 CVault

Implements the following interfaces:

CVault

↑

IVault2

↑

IVault

Properties

- **Id** – Vault ID (read only)
- **HostName** – The name of computer which the vault is running on (read only)
- **Domain** – The domain of the computer which the vault is running on (read only)

- **NetworkAddress** – IP address of the vault (read only)
- **Description** – Hardware tag of the computer which the vault is running on (read only)
- **CreationDate** – when the vault was created (read only), valid for vault with version later than 6.20
- **GUID** – GUID of the vault (read only)
- **ReplicationType** – the replication role of this vault (enum VaultReplicationType). This property does not support multiple vault roles. If a vault has more than one role, "ReplicationUnknown" is returned by this property and ActiveRoleState, PassiveRoleState, SatelliteRoleState and BaseRoleState properties are used to find the vault roles.

  Note: A vault that is version 7.0 or later can have more than one replication role (e.g., both Active and Base).
- **ActiveRoleState** – check the Active role on this vault (enum RoleState) (read only)
- **PassiveRoleState** – check the Passive role on this vault (enum RoleState) (read only)
- **SatelliteRoleState** – check the Satellite role on this vault (enum RoleState) (read only)
- **BaseRoleState** – check the Base Vault role on this vault (enum RoleState) (read only)

2.12 CProcess

Implements the following interface:

CProcess

↑

IProcess

Properties

- **Operation** – type of operation (enum OperationCode), read only
- **Description** – description of the process, read only
- **ProcessId** – process ID, read only
- **ThreadId** – thread ID, read only
- **SessionGUID** – session GUID, read only
- **TaskName** – task name, read only
- **TaskId** – task ID, read only
- **StartedAt** – start time, read only
- **StatusMessage** – status message, read only
- **TotalBytes** – total bytes transferred, read only

- **TotalFiles** – total files transferred, read only
- **SafesetBytes** – safeset bytes transferred, read only
- **Status** – status of the process (enum ProcessStatus) , read only
- **MiscCode** – process code for VSMP replication or Unknown replication process (enum ProcessMiscCode), read only

2.13 CStorageLocation

Implements the following interface:

CStorageLocation

↑

IStorageLocation

Properties

- **OnlineGroup** – primary storage group of a storage location
- **Location** – storage location name
- **Policy** – storage policy for a storage location
- **User** – username for connecting to a storage location
- **Domain** – account domain for connecting to a storage location
- **Password** –password for connecting to a storage location
- **Id** – unique identifier for a storage location, read only

2.13.1 Update

Updates the storage location's properties.

Syntax

```
HRESULT update([in]IDispatch* vaultInfo);
```

Parameters

- **vaultInfo** – CVaultConnection object, indicates the vault to be connected

2.14 OperationCode enum

Used in the process object to specify the type of operation.

```
enum OperationCode
{
        Backup,
        Restore,
        Verify,
        Synch,
        Billing,
        Import,
        Export,
        Recall,
        Update,
        Migrate,
        ServerProcess,
        Purge,
        Analyze,
        Optimize,
        VMServer,
        Index,
        Rename,
        Extract,
        DBLoad,
        PoolopSummary,
        PoolopDump,
        PoolopVerify,
        PoolopRecreate,
        PoolopCheckCRC,
        PoolopSSIVerify,
        IndexOptimize,
```

```
IndexCreate,
IndexRepair,
IndexDump,
IndexVerify,
Archive,
QSMImport,
QSMExport,
QSMInit,
VVCopy,
VVCopyLocal,
DBBackckup,
UpgradeTo5,
MigrateSecondary,
SecondaryOpMount,
SecondaryOpUnmount,
SecondaryOpMountAll,
SecondaryOpUnmountAll,
SecondaryOpSSIVerOne,
SecondaryOpSSIVerAll,
SecondaryOpCheckCRC,
SecondaryOpClose,
QSMOpExport,
QSMOpImport,
QSMOpInit,
RegisterComputer,
RegisterTask,
SynchWeb,
ProgReplication,
ServerHost,
ReplHost,
```

```
        Dedup,
        ReplicationCMD,
        ScanLog,
        SecondaryOpDelete,
        SecondaryOpDeduplicate,
        SecondaryOpOptimize,
        SecondaryOpExpire,
        VVCopyDest,
        Updater,
        Maintenance,
        QSMImportDest,
        QSMExportDest,
        QSMInitDest,
        QMCMD,
        QMReportJobsOlderThanDays,
        MaintenanceHost,
        SecondaryOpCleanup,
        SecondaryOpVerifyIDXOne,
        SecondaryOpVerifyIDXAll,
        PoolopUnsuspect,
        ConfigurationReplication,
        MetadataReplication,
        DataReplication,
        ProgUnknown,
        Other
};
```

2.15 OperatingModeType enum

Used in account and task objects to specify or retrieve the operating mode.

```
enum OperatingModeType
{
    Default,
    PausedReplication,
    RedirectedBackups
};
```

2.16 OBJECT_TYPE_ENUM enum

Specifies the object type under control.

```
enum OBJECT_TYPE_ENUM
{
    OTE_VAULT = 1,
    OTE_CUSTOMER,
    OTE_LOCATION,
    OTE_COMPUTER,
    OTE_TASK,
    OTE_ACCOUNT,
    OTE_USER,
    OTE_BILLINGCODE,
    OTE_SAFESET,
    OTE_SATELLITE_CFG,
OTE_TASKPATH,
    OTE_UNKNOWN = 256
};
```

2.17 ProcessStatus enum

Specifies the status of a process.

```
enum ProcessStatus
{
    Inactive,
    Running,
    RequestPending,
    Queued
};
```

2.18 RoleState enum

Specifies whether a vault is acting as an active, passive, satellite or Base vault.

```
enum RoleState
{
    Off,
    On,
    Unconfigured
};
```

2.19 SatelliteMode enum

Defines satellite modes.

```
enum SatelliteMode
{
    Regular,
    Disconnected // EVault for DPM satellite
};
```

2.20 VaultReplicationType enum

Specifies the replication type of a vault.

```
enum VaultReplicationType
{
    Replication_Active,
    Replication_Passive,
    Replication_Base_Vault,
    Replication_Satellite_Vault,
    Replication_1To1_Unconfig,
    Replication_1To1_Not_Coupled_Yet,
    Replication_None,
    Replication_Unknown
};
```

2.21 WorkingStatus enum

Specifies the status of a task.

```
enum WorkingStatus
{
    Enabled,
    Disabled,
    Suspect
};
```

3 Sample Code

This section provides C# syntax examples for implementing COM objects.

3.1 Creating a Customer, Location, Account, and User

…

```
                try
                {
                        SbeAccountManager.CVaultConnectionClass v = new
SbeAccountManager.CVaultConnectionClass();
                        v.Address = "192.168.1.100"; // add your vault here
                        v.userName = "user"; // add your credentials here
                        v.Password = "pwd";
                        v.Domain = "";
                        v. DefaultRaidArea = "SG01";
                        SbeAccountManager.CCustomer c = new
SbeAccountManager.CCustomerClass();
                        c.Name = "EVault";
                        c.ShortName = "EVLT";
                        c.Address = "address";
                        c.City = "city";
                        c.ZipCode = "zipCode";
                        c.State = "state";
                        c.Country = "country";
                        c.Phone = "1-800-123-4567";
                        c.Email = "help@evault.com";
                        c.Url = "http://www.evault.com";
                        c.Notes = "notes";
                        SbeAccountManager.CLocationClass l = new
SbeAccountManager.CLocationClass();
                        l.Name = "Oakville";
                        l.billingCode = "o1234567891011112";
```

```
                    SbeAccountManager.CAccountClass a = new
SbeAccountManager.CAccountClass();

                    a.Name = "EVaccount";

                    a.Description = "";

                    a.OperatingMode =
SbeAccountManager.SatelliteOperatingMode.Default;

                    SbeAccountManager.CUserClass u = new
SbeAccountManager.CUserClass();

                    u.Name = "EVusr";

                    u.Password = "EVpwd";

                    SbeAccountManager.CManagerClass com = new
SbeAccountManager.CManagerClass();

                    com.ReqestId = "myId";

                    com.LogFileName = @"c:\temp\ComExecution.log";

                    com.AuditFileName = @"c:\temp\ComAudit.csv";

                    com.Create( v, c, l, a, u);

                }

                catch(COMException ex)

                {

…

                }

                catch(Exception ex)

                {

…

                }

…
```

3.2 Retrieving a Customer list

```
SbeAccountManager.ICustomerCollection coll = com.getCustomerList(v) as
SbeAccountManager.ICustomerCollection;

foreach (SbeAccountManager.ICustomer cust in coll)
```

Update Customer

```
cust.update(v);
```

3.3 Retrieving a Location list

```
int custId = cust.Id;

SbeAccountManager.ICollection coll = com.getLocationList(v, custId);

foreach (SbeAccountManager.ILocation l in coll)

{

}
```

3.4 Retrieving a Computer List

```
SbeAccountManager.ICollection comps = com.getComputerList(v,
locationId);

foreach (SbeAccountManager.IComputer c in comps)
```

3.5 Retrieving a Task List

```
SbeAccountManager.ICollection coll = com.getTaskList(v, computerId);

foreach (SbeAccountManager.ITask t in coll)
```

3.6 Retrieving a Safeset List

```
SbeAccountManager.ICollection coll = com.getSafesetList(v, taskId);

foreach (SbeAccountManager.ISafeset s in coll)
```

3.7 Deleting a Computer

```
com.deleteComputer(_v, locationId, computerId);
```

3.8 Deleting a Task

```
com.deleteTask(_v, computerId, taskId);
```

3.9 Enabling a User

```
com.enableObject(v,SbeAccountManager.OBJECT_TYPE_ENUM.OTE_USER,
userId,true);
```

3.10 Disabling an Account

```
com.enableObject(v,SbeAccountManager.OBJECT_TYPE_ENUM.OTE_ACCOUNT,
accountId,false);
```

3.11 Retrieving a Primary Storage Group List

```
Array onlineGroupList = com.getOnlineGroupList(v);

foreach (String s in onlineGroupList)
```

3.12 Using a GUID to Find a Task

```
SbeAccountManager.CTask task =
(SbeAccountManager.CTask)com.getTaskByGuid (_v, taskGuid);
```

3.13 Reading and Modifying a Satellite Replication Configuration

```
            IReplicationSatelliteConfig replicationSatelliteConfig =
(IReplicationSatelliteConfig)com.getSatelliteConfiguration(_v,
customerId);

            replicationSatelliteConfig.UseDataEncryption = false;

            replicationSatelliteConfig.ThrottleLimit = (float)23.3;

            replicationSatelliteConfig.ThrottleDays = 3;

            replicationSatelliteConfig.StartThrottleTime = new
DateTime(2001,1,1,14,0,0);

            replicationSatelliteConfig.EndThrottleTime = new
DateTime(2001, 1, 1, 15, 0, 0);

            replicationSatelliteConfig.DisableThrottling = false;

            replicationSatelliteConfig.UseThrottlingAllDay = false;

            replicationSatelliteConfig.WarmReplication = false;

            replicationSatelliteConfig.SafesetPolicy = 1;

            replicationSatelliteConfig.RetentionDays = 2;

            replicationSatelliteConfig.RetentionCopies = 7;

            replicationSatelliteConfig.HeartbeatInterval = 3;
```

```
            replicationSatelliteConfig.BackupOption =
SatelliteOperatingMode.ReplicateCustomerOnly;

            replicationSatelliteConfig.update(vault);
```

3.14 Retrieving and Removing Computer Licenses

```
Array claimedList = _com.getComputerLicenseList(_v,computerId);

_com.removeComputerLicenseList(_v, computerId, claimedList);
```

3.15 Getting the First Vault

```
SbeAccountManager.IVault vault = _com.getFirstVault(_v) as
SbeAccountManager.CVault;
```

4 Error Codes

These are the error codes that can appear when you run COM syntax:

- Invalid number of parameters (0x80040001);
- Failed to create vault COM object (0x80040002);
- Failed to create customer COM object (0x80040003);
- Failed to create location COM object (0x80040004);
- Failed to create account COM object (0x80040005);
- Failed to create user COM object (0x80040006);
- Memory failure (0x80040007);
- Failed to connect to the vault (0x80040008);
- Vault authorization failed (0x80040009);
- Location with the same name already exists (0x8004000a);
- Customer with the same name already exists (0x8004000b);
- Billing code already exists (0x8004000c);
- Account with the same name already exists (0x8004000d);
- User with the same name already exists (0x8004000e);
- Failed to create location (0x8004000f);
- Failed to create billing code (0x8004000f);
- Failed to get customer list (0x80040010);
- Failed to update location information (0x80040011);
- Failed to get customer list (0x80040012);
- Failed to create account and user (0x80040013);
- Failed to properly disconnect (0x80040014);
- Vault error (0x80040015);
- COM error (0x80040016);
- C++ error (0x80040017);
- General error (0x80040018);
- Failed – customer does not exist (0x80040019);

- Failed to get customer quota information (0x8004001a);
- Invalid customer quota type (0x8004001b);
- Exceeded customer quota (0x8004001c);
- Failed to assign customer quota (0x8004001d);
- Invalid billing number (0x8004001e);
- Invalid location name; length exceeded (0x8004001f);
- Invalid location name; zero length (0x80040020);
- Invalid customer name; length exceeded (0x80040021);
- Invalid zero length customer name; zero length (0x80040022);
- Invalid customer short name; length exceeded (0x80040023);
- Invalid zero length customer short name; zero length (0x80040024);
- Invalid account name; length exceeded (0x80040025);
- Invalid zero length account name; zero length (0x80040026);
- Invalid user name; length exceeded (0x80040027);
- Invalid zero length user name; zero length (0x80040028);
- Invalid password; length exceeded (0x80040029);
- Invalid parameter NULL passed (0x8004002A);
- Failed to delete task (0x8004002B);
- Failed to delete Computer (0x8004002C);
- Failed to delete User (0x8004002D);
- Failed to delete Customer (0x8004002E);
- Failed to delete Location (0x8004002F);
- Failed to delete Safeset (0x80040030);
- Failed to delete Account (0x80040031);
- Failed to enable Object (0x80040032);
- Failed to disable Object (0x80040033);
- Failed to update Object (0x80040034);
- Failed to delete Billing Code (0x80040035);
- Invalid Secondary group property (0x80040036);

- Invalid Archive group property (0x80040037);
- Satellite cannot be created from a non-Base vault (0x80040038);
- Storage quota type is not available on BAV (0x80040039);
- Failed to create Satellite entry (0x8004003A);
- Failed to get license usage (0x8004003B);
- Failed to get license setting (0x8004003C);
- Failed to get satellite quota list (0x8004003D);
- Customer is already associated with the satellite (0x8004003E);
- Invalid expiry time (0x8004003F);
- Unsupported property (0x80040040);
- Provided operating mode is not supported (0x80040042);
- Provided operating mode is not supported on Passive (0x80040043);
- Changing of operating mode is not allowed (0x80040044);
- Overwriting global operating mode is not allowed (0x80040045);
- Updating satellite configuration is not allowed (0x80040046);
- Updating AllowSatelliteSetting flag is not allowed (0x80040047);
- Updating satellite configuration failed (0x80040047);
- Provided vault is not a BAV or a satellite (0x80040048);
- Operating mode cannot be changed on the satellite if it is already set to 'Customer Only' (0x80040049);
- Can’t set 'Customer Only' operating mode on satellite (0x8004004A);
- Vault's version is higher than VaultApi version (0x80040050);
- Failed to delete satellite (0x80040051);
- Failed to find the task (0x80040052);
- Failed to find the computer (0x80040053);
- Failed to get licenses for a given computer (0x80040054);
- Failed to remove licenses for a given computer (0x80040055);
- Failed to handle safe array for computer licenses (0x80040056);
- Failed to get first vault (0x80040057);

- Failed to get creation date from vault (0x80040058);
- Failed to get safeset list (0x80040059);
- Failed to get replication type (0x8004005A);
- Failed to get account (0x8004005B);
- Failed to get user (0x8004005C);
- Failed to get customer (0x8004005D);
- Failed to get operating mode (0x8004005E);
- Failed to set operating mode (0x8004005F);
- Cannot modify satellite configuration on BAV_P (0x80040060);
- Failed to stop vault process (0x80040061);
- Invalid online storage group (0x80040062);
- Invalid online storage location (0x80040063);
- Invalid replication mode specified or unconfiged replication (0x80040064);
- Failed to get Replication Service Status (0x80040065);
- Failed to set Replication Service Status (0x80040066);
- Failed to get maintenance Service Status (0x80040067);
- Failed to set maintenance Service Status (0x80040068);
- Failed to cancel the pending maintenance jobs (0x80040069);