EVault Software

Vault Service API

User Guide

*version 1.0*

1 Introduction 2

1.1 Setting up the API 2

1.2 Installing the Vault Web Service on Windows Server 2008 2

1.3 Edit the web config file. 4

2 Vault Service API Objects 4

2.1 Customer 4

2.1 Computer 6

3 Vault Service API Web Methods 6

3.1 Get All Customers - GET /api/customers 6

3.2 Get Customer - GET /api/customers/[customerName] 7

3.3 Get Customer Computers - GET /api/customers/[customerName]/computers 7

3.3 Create Customer - POST /api/customers 7

3.4 Update Customer - PUT /api/customers/[customerName] 8

3.5 Delete Customer - Delete /api/customers/[customerName] 8

3.6 Enable Customer - POST /api/customers/[customerName]/enable 8

3.6 Disable Customer - POST /api/customers/[customerName]/disable 8

4 Return Codes and Error Codes 8

4.1 Error Codes 8

# 1 Introduction

This guide provides information and procedures for installing and configuring the EVault Vault API Web Service. The Vault API Web Service is a RESTful API built on Microsoft ASP.NET MVC 4 and .NET 4.5 framework. It calls the Vault API COM object to make remote calls to the EVault Director. It provides remote access to Vault functionality to allow the creation, deletion, retrieval and updating of Vault customers, accounts, and user objects.

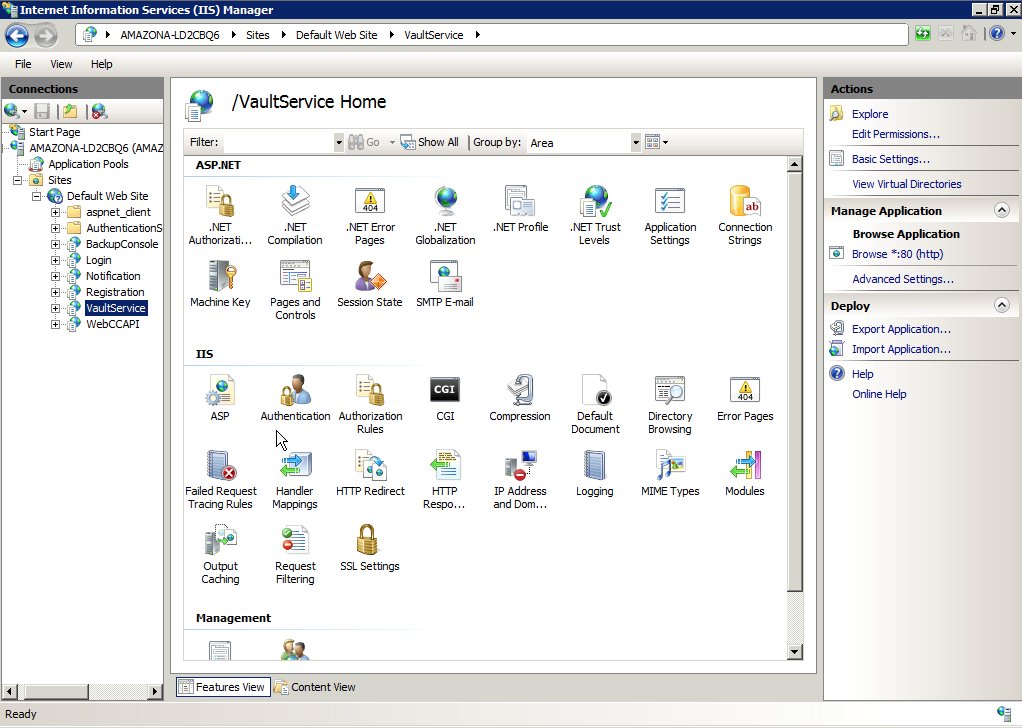
## Setting up the API

The Vault API Web Service is an unsecured API, and must not be exposed outside an internal network. Install the Vault API Web Service on an internal web site that can only be accessed from within the network, or a specific IP address.

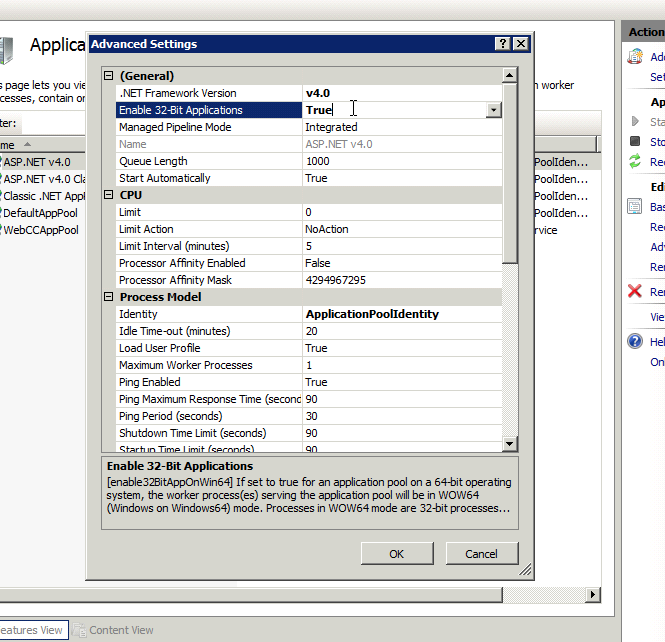
## Installing the Vault Web Service on Windows Server 2008

To set up the web service:

1. Extract the VaultServiceDeployment.zip files to C:\inetpub\wwwroot\VaultService directory.
2. Install the Vault API com object is installed the server. Contact EVault support for the Vault API installation file.
3. Install .NET 4.5 framework is installed on web server (<http://msdn.microsoft.com/en-us/library/5a4x27ek.aspx>).
4. Ensure Web Server has network connectivity to the Vault Servers. Port 809 is required to communicate with the Vault Service.
5. Ensure IIS 6 or 7 is installed on the web server.
6. In IIS, set up a virtual application named ‘VaultService’ under the default web site.



1. Set up a Application Pool running .Net 4.0 framework. Go to Edit / Advanced Settings, and Enable 32 bit Applications. This is necessary on 64 bit machines to run the unsigned 32 bit Vault API com object.



1. Configure the VaultService web application to use the above .NET 4.0 application pool.
2. Go to Vault Service web application, and then Edit Application.
3. Set the Application Pool to the above .NET 4.0 application pool.
4. The Vault Service web application needs file access to the Vault API dll. Click ‘Connect As’ and select a user with read/write access to the Vault Service directory physical path.
5. Stop/Start the web site (either type ‘iisreset’ in run command or right click Stop/Start on the default web site.

## Setting Log file location and WebCC url

## WebCC Url configuration:

The WebCC API is not a RESTful API – it’s a SOAP service. The test page uses a proxy server to pass through the request to the WebCC API and in effect, allow the WebCC API to be called RESTfully.

1. Navigate to C:\Inetpub\wwwroot directory.
2. Double-click the web.config file.
3. Navigate to:

<VaultService.Properties.Settings>

<setting name="VaultService\_WebCCApi\_Service" serializeAs="String">

<value>[**http://ec2-107-22-101-18.compute-1.amazonaws.com/WebCCAPI/Service.asmx**</value](http://ec2-107-22-101-18.compute-1.amazonaws.com/WebCCAPI/Service.asmx%3c/value)>

1. Enter the appropriate url for the WebCC API (see WebCC API documentation for install procedures).

## Log/Audit file configuration:

The Vault API (com object) writes a log of it’s activities to a text file. This can be configured in the Vault Service web config. Note, the VaultService web application will need read/write privaleges to the this folder.

1. In the web.config, navigate to and ensure the appropriate config settings:

<add key="LogFileName" value="C:\VaultService\ComExecution.log" />

<add key="AuditFileName" value="C:\VaultService\ComAudit.csv" />

## Multiple Vault Configuration

The Vault Service can comminicate to multiple vaults. Each web request takes a “Vault” parameter which specifies the Vault the web request will act against. The Vault Service will then look up the appopriate connectivity settings for that Vault in the web config. If no “Vault” is specified, it will use the default Vault. The server where the Vault Service is installed must have network connectivity to the appropriate vaults and additionally, each vault must have the configuration set in the web config.

To edit the web.config file:

1. Navigate to C:\Inetpub\wwwroot directory.
2. Double-click the web.config file.
3. Ensure the network connectivity for the vault information is correct.

## Default Vault configuration:

<!-- default vault connection settings -->

<add key="Address" value="ec2-107-22-101-18.compute-1.amazonaws.com" />

<add key="DefaultRaidArea" value="" />

<add key="DefaultWorkArea" value="SG01" />

<add key="Description" value="" />

<add key="Domain" value="" />

<add key="ConnectionPassword" value="iufz$4dZ9Nb" />

<add key="ConnectionUser" value="Administrator"

## Multiple Vault configuration:

Additional Vaults can be configured by appending the Vault name before the configuration key. In this case, the additional vault is named “DogtownServer”. When VaultService requests pass in the parameter “Vault=DogtownServer”, the request will be routed to matching Vault connection information.

<add key="DogtownServer-Address" value="ec2-107-22-101-18.compute-1.amazonaws.com" />

<add key="DogtownServer-DefaultRaidArea" value="" />

<add key="DogtownServer-DefaultWorkArea" value="SG01" />

<add key="DogtownServer-Description" value="" />

<add key="DogtownServer-Domain" value="" />

<add key="DogtownServer-ConnectionPassword" value="iufz$4dZ9Nb" />

<add key="DogtownServer-ConnectionUser" value="Administrator" />

# 2 Vault Service API Objects

## 2.1 Customer

|  |  |  |  |
| --- | --- | --- | --- |
| Customer field | Required | Field Type | Description |
| CustomerName | true | String |  |
| CustomerAddress | False | String |  |
| CustomerCity | False | String |  |
| CustomerZipCode | False | String |  |
| CustomerState | False | String |  |
| CustomerCountry | False | String |  |
| CustomerURL | False | String |  |
| CustomerNotes | False | String |  |
| CustomerEmail | False | String |  |
| CustomerContactName | False | String |  |
| VaultName | False | String |  |
| ShortName | False | String |  |
| Address | False | String |  |
| UserName | False | String |  |
| Password | False | String |  |
| Domain | False | String |  |
| Description | False | String |  |
| DefaultRaidArea | False | String |  |
| DefaultRaidArea | False | String |  |
| Computers | False | List<Computer> |  |

Example customer object in JSON format

{

"CustomerName": "Bouygue Telecom",

"CustomerAddress": "",

"CustomerCity": "",

"CustomerZipCode": "",

"CustomerState": "",

"CustomerCountry": "",

"CustomerURL": "",

"CustomerNotes": "",

"CustomerEmail": "john.doe@mail.com",

"CustomerContactName": "",

"VaultName": "",

"ShortName": "bytel",

"Address": "",

"UserName": "",

"Password": "",

"Domain": "",

"Description": "",

"DefaultRaidArea": "",

"DefaultWorkArea": "",

"BillingCode": "SV1OK",

"Computers": [

{

"AgentType": "ATE\_SERVER\_AGENT",

"ProductCode": "",

"AgentVersion": "6.82.3388",

"IpAddress": "188.168.1.102",

"Name": "FS01",

"OsType": "Windows (32 bit)",

"OsVersion": "Windows Server 2008 SP2.0 ",

"Domain": "",

"Tasks": [

{

"Name": "Job2",

"IsActive": true,

"Id": 2,

"PhysicalPoolSize": 284076,

"UsedPoolSize": 284076

}

],

"TotalCompressedSize": 282144,

"TotalOriginalSize": 333485

}

]

}

## 2.1 Computer

Example Computer object in JSON format:

{

"AgentType": "ATE\_SERVER\_AGENT",

"ProductCode": "",

"AgentVersion": "6.82.3388",

"IpAddress": "188.168.1.102",

"Name": "FS01",

"OsType": "Windows (32 bit)",

"OsVersion": "Windows Server 2008 SP2.0 ",

"Domain": "",

"Tasks": [

{

"Name": "Job2",

"IsActive": true,

"Id": 2,

"PhysicalPoolSize": 284076,

"UsedPoolSize": 284076

}

],

"TotalCompressedSize": 282144,

"TotalOriginalSize": 333485

}

# 3 Vault Service API Web Methods

## 3.1 Get All Customers - GET /api/customers

This is a GET http request that returns all the customers on the vault. Note, when populating computers and usage, extra processing to retrieve computers and calculate usage is required. As the number of customers on a vault increases, the performance will decrease.

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Required | Field Type | Description |
| populateComputers | False | Boolean [true, false] | Populates the computer lists for each customer. |
| populateUsage | False | Boolean [true, false] | Populates the usage for each computer when populateComputers is also set to true. |
| Vault | False | String | Specifies which vault to run the request against. Note, it will default when there is only one vault. It is required only in instances where the Vault API Web Service acts upon multiple vaults |

## 3.2 Get Customer - GET /api/customers/[customerName]

This is a GET http request that returns a single customerName.

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Required | Field Type | Description |
| customerName | true | String | The customer name of the request. |
| populateComputers | False | Boolean [true, false] | Populates the computer lists for each customer. |
| populateUsage | False | Boolean [true, false] | Populates the usage for each computer when populateComputers is also set to true. |
| Vault | False | String | Specifies which vault to run the request against. Note, it will default when there is only one vault. It is required only in instances where the Vault API Web Service acts upon multiple vaults |

## 3.3 Get Customer Computers - GET /api/customers/[customerName]/computers

This is a GET http request that returns a list of computers by customerName.

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Required | Field Type | Description |
| customerName | true | String | The customer name of the request. |
| populateComputers | False | Boolean [true, false] | Populates the computer lists for each customer. |
| populateUsage | False | Boolean [true, false] | Populates the usage for each computer when populateComputers is also set to true. |
| Vault | False | String | Specifies which vault to run the request against. Note, it will default when there is only one vault. It is required only in instances where the Vault API Web Service acts upon multiple vaults |

## 3.3 Create Customer - POST /api/customers

This is a POST http request that takes a customer object and creates a new customer. See the Customer object in section 2. Note, the Vault where customer located is passed in via the VaultName parameter. Null or empty VaultName will mean the default Vault will be used.

## 3.4 Update Customer - PUT /api/customers/[customerName]

This is a PUT http request that takes a customer object and creates a new customer. See the Customer object in section 2.

## 3.5 Delete Customer - Delete /api/customers/[customerName]

This is a DELETE http request that takes a customer name and deletes the customer. Note, a delete is not reversible and will also delete the corresponding backup data from the vault.

## 3.6 Enable Customer - POST /api/customers/[customerName]/enable

This is a POST http request that enables a customer account and allows for successful continuation of Backup/Restore processes.

## 3.6 Disable Customer - POST /api/customers/[customerName]/disable

This is a POST http request that disables a customer account and prevents Backup/Restore processes. Typical use case is when the customer’s account is not current (lack of payment).

# 4 Return Codes

## 4.1 Return Codes

Return Codes:

“0” = Success

“1” = General Error

“2” = Object Not Found Error

“3” = Object Already Exists Error. Uniqueness validation error.