Credential Store and Encryption Vault Service

Contents

Copyright GE Digital

Vault Service Overview

About Vault Service

Predix platform provides Vault as a service for securely storing and accessing credentials.

You can use the Vault service to securely store and manage access to tokens, passwords, API keys and other credentials. The service also provides detailed audit logs of Vault usage.

Note: The Vault service is available in the Predix US-East domain (`https://predix-io.run.asv-pr.ice.predix.io`) and the Predix US-West domain (`https://api.system.aws-usw02-pr.ice.predix.io`).

The vault service is based on open source Hashicorp Vault. For more information on Hashicorp Vault, see www.vaultproject.io.

Get Started With Vault Service

Vault Service Setup

Task Roadmap

#	Task	Information
1	Deploy your application to Cloud Foundry.	For an example of deploying a Predix Hello World Web application to cloud foundry, see Creating and Deploying a Simple Web App to Cloud Foundry.
2	Create an instance of the Vault service.	For more information, see Creating a Vault Service Instance on page 2.
3	Bind your application to the service instance.	To establish communication between your application and the platform service, you must bind the application to the service. See Binding an Application to the Vault Service Instance on page 3.
4	(optionally) Create a service key for your instance.	This is an alternative procedure to retrieve Vault credentials if you do not bind an application to your Vault service instance. See Create a Service Key for Vault Service on page 4.
5	Start using the Vault service.	See Using Vault Service.

Creating a Vault Service Instance

You can create an instance of Vault Service to securely store and manage access to tokens, passwords, API keys and other credentials.

Procedure

1. Sign into your Predix account at https://www.predix.io.
2. Navigate to **Catalog > Services** tab, and click the **Credential Store** tile.
3. Click **Subscribe** on the required plan.
4. On the **New Service Instance** page, enter:

Field	Description
Org	Select your org.
Space	Select the space for your application.
Service instance name	Specify a unique name for your instance.
Service plan	Select a plan.

5. Click **Create Service**.

Results

The service instance is created and displayed in your console.

Next Steps

You can retrieve the Vault Credentials from the VCAP_SERVICES environment variable after binding it to an application. Alternatively, you can retrieve the Vault Credentials by creating a service key.

Using Command Line to Create a Vault Instance

Procedure

1. Use the Cloud Foundry CLI to log into Cloud Foundry.

   ```
   cf login -a <API_Endpoint>
   ```

   Vault service is available for the users of Predix US-East domain and US-West domain. The value of `<API_Endpoint>` is `https://api.system.asv-pr.ice.predix.io` or `https://api.system.aws-usw02-pr.ice.predix.io`.
2. List the services in the Cloud Foundry marketplace.

   ```
   cf marketplace
   ```

   The Vault service, `predix-vault`, is listed as an available service.
3. Create a Vault service instance.

   ```
   cf create-service predix-vault <plan> <my_vault_instance>
   ```

   where:

   - `<plan>` is the plan associated with a service.
   - `<my_vault_instance>` is the service instance you are creating.

   The message on the screen indicates that the instance is created.

Next Steps

You can retrieve the Vault Credentials from the VCAP_SERVICES environment variable after binding it to an application. Alternatively, you can retrieve the Vault Credentials by creating a service key.

Binding an Application to the Vault Service Instance

You must bind your application to your Vault instance to provision its connection details in the VCAP_SERVICES environment variable. Vault service creates a token when you bind your application to the Vault service instance.

About This Task

Cloud Foundry runtime uses VCAP_SERVICES environment variable to communicate with a deployed application about its environment. When you bind your application to the Vault service instance, by default a token is created with time to live (TTL) value of 32 days. You can use custom parameters to specify a different TTL for the token and to create a read-only token.

Procedure

1. Use the Cloud Foundry CLI to log into Cloud Foundry.

   ```
   cf login -a <API_Endpoint>
   ```

Vault service is available for the users of Predix US-East domain and US-West domain. The value of `<API_Endpoint>` is `https://api.system.asv-pr.ice.predix.io` or `https://api.system.aws-usw02-pr.ice.predix.io`.

2. List the services in your Cloud Foundry space.

```
cf services
```

Your Vault service instance, `vault_instance_name`, is listed as an available service.

3. Bind your application to the Vault instance.

```
cf bind-service <your_app_name> <vault_instance_name>
```

The `<vault_instance_name>` instance is bound to your application, and the following message is returned:

```
Binding service <vault_instance_name> to app <your_app_name> in org
predix-platform / space predix as userx@ge.com...
OK
TIP: Use 'cf restage' to ensure your env variable changes take effect
```

4. (Optionally) You can specify custom parameters for the following:
   - To specify the duration of the token for your Vault service instance.
     - For a token with unlimited duration, use the following command:

```
cf bind-service <your_app_name> <vault_instance_name> -c
'{"token_type" : "perpetual"}'
```

     - For token with specific TTL value, use the following command:

```
cf bind-service <your_app_name> <vault_instance_name> -c
'{"token_type" : "periodic", "token_ttl" : "<duration>"}'
```

   - To add a read-only policy to the token. When you add a read-only policy, a user can read the token details but cannot create, update or delete the token. Use the following command:

```
cf bind-service <your_app_name> <vault_instance_name> -c '{"
token_read_only: true "}'
```

5. Use the following command to verify the binding and view the environment variables:

```
cf env <your_app_name>
```

Create a Service Key for Vault Service

You can optionally retrieve Vault credentials without binding an application to a Vault instance by creating a service key.

Procedure

1. Use the Cloud Foundry CLI to log into Cloud Foundry.

```
cf login -a <API_Endpoint>
```

Vault service is available for the users of Predix US-East domain and US-West domain. The value of `<API_Endpoint>` is `https://api.system.asv-pr.ice.predix.io` or `https://api.system.aws-usw02-pr.ice.predix.io`.

2. List the services in your Cloud Foundry space.

```
cf services
```

   Your Vault service instance `<vault_instance_name>` is listed as an available service.

3. Use the following command to create the service key.

```
cf create-service-key <vault_instance_name> <service-key-name>
```

(Optionally) You can specify custom parameters for the following:

- To specify the duration of the token for your Vault service instance.
  - For a token with unlimited duration, use the following command:

```
cf create-service-key <vault_instance_name> <service-key-name> -c
'{"token_type" : "perpetual"}'
```

  - For token with specific TTL value, use the following command:

```
cf create-service-key <vault_instance_name> <service-key-name> -c
'{"token_type" : "periodic", "token_ttl" : "<duration>"}'
```

- To add a read-only policy to the token. When you add a read-only policy, a user can read the token details but cannot create, update or delete the token. Use the following command:

```
cf create-service-key <vault_instance_name> <service-key-name> -c
'{" token_read_only: true "}'
```

4. Retrieve the credential by reading the service key.

```
cf service-key <vault_instance_name> <service-key-name>
```

   This command retrieves the following information:

   - vault_accessor
     Alias of the token for Vault service instance.
   - vault_token
     Token for accessing the Vault service instance. You use this token to log in to Vault dashboard.
   - vault_url
     URL of the Vault service instance. You use this URL to log in to the Vault dashboard. The URL value includes the root path (as a set of GUIDs) to your Vault instance.

Using the Vault Service

Ways of Using the Vault Service

You can use Vault service in multiple ways depending on your use case. Vault service provides a complete set of REST API to use the service through you application. You can also use Vault command-line interface (CLI). Additionally Predix platform provides Vault dashboard to use graphical user interface to use the Vault service.

The Vault CLI is a command line application. To view a list of the available commands at any time, you can run Vault with no arguments. To get help for any specific subcommand, run the subcommand with the `-h` argument. For more information, see Vault Commands.

Vault REST APIs give you the flexibility of using Vault service through your applications. The following topics describe using Vault APIs to perform various tasks in Vault service:

- Managing Paths and Secrets
- Managing Tokens
- Data Encryption

Vault dashboard provide a user interface for working with Vault service. The following topic describes the tasks performed using Vault dashboard:

- Managing Paths and Secrets Using Vault Dashboard on page 28

Managing Paths and Secrets

About Managing Paths and Secrets

You can add, edit or delete paths and secrets in your Vault service instance. A path specifies the storage location of your secret. Vault service storage mechanism is similar to virtual file system. When you create an instance of Vault service, the default storage location is defined by a specific path that is equivalent to the home directory in a file system.

You can then add sub paths to the default path to define additional storage locations or nodes. The sub-paths are equivalent to the hierarchy of folders and sub-folders in a file system. The secrets are stored under a path as key value pairs.

You can either use REST APIs to manage the paths or use the Vault dashboard UI. For more information on Vault dashboard, see Managing Paths and Secrets Using Vault Dashboard on page 28.

To construct the URL for Vault service API, you can get the `vault_url` from your VCAP_SERVICES environment variable. The `vault_url` contains the root path (as a set of GUIDs) to your vault instance.

The following is an example of the `vault_url` containing the root path:

```
https://predix-vault.aws-usw02-pr.ice.predix.io/v1/secret/81f37567-
f14c-4289-817b-57b15ee24d2e/078221f7-da65-491c-9185-4d3f47442e9f/
6cb25da8-7206-4dd2-944e-9717c04a0a7e
```

You can use Vault service API to perform the following tasks related to managing paths and secrets:

- Storing a Secret on page 7
- Retrieving a Secret on page 7
- Retrieving a List of Key Names on page 8

- Deleting a Secret on page 8

Additionally, Vault service provides one time secret sharing capability by wrapping the value. To use this capability, you can use the following APIs:

- Wrapping a Value in a Response-Wrapped Token on page 9
- Returning Wrapping Token Properties on page 10
- Re-wrapping a Response-Wrapped Token on page 10
- Unwrapping a Wrapped Response on page 11

APIs: Creating, Retrieving, and Deleting Paths and Secrets

You can use APIs to manage paths and secrets in Vault.

Storing a Secret

API	`/secret`
Description	Stores a secret at the specified location.
Method	`POST/PUT`
URL	`/secret/<path>` Where, `<path>` is the root path to your vault instance plus any sub-path that you create.
Parameters	`key` Specify a key name paired with an associated value to be stored at the given path. You can specify multiple key/value pairs. You can retrieve all values using the read operation.
Header	`X-Vault-Token: <token>`
Returns	A `204` response code.

Retrieving a Secret

API	`/secret`
Description	Retrieve the secret at the specified location.
Method	`GET`
URL	`/secret/<path>` Where, `<path>` is the root path to your vault instance plus any sub-path that you create.
Parameters	None
Header	`X-Vault-Token: <token>`

Returns	<pre>{
	"auth":null,
	"data":{</pre>

	<pre>"foo":"bar"
	},
	"lease_duration":2764800,
	"lease_id":"",
	"renewable":false
	}</pre>

Retrieving a List of Key Names

API	`/secret`
Description	Returns a list of key names at the specified location. Folders are suffixed with `/`. The input must be a folder; list on a file will not return a value. The values themselves are not accessible via this command. **Note:** Policy-based filtering is not performed on keys. Therefore do not encode sensitive information in key names.
Method	`LIST/GET`
URL	`/secret/<path>` (LIST) or `/secret/<path>?list=true` (GET) Where, `<path>` is the root path to your vault instance plus any sub-path that you create.
Parameters	None
Header	`X-Vault-Token: <token>`

Returns	The example below shows output for a query path of `secret/` when there are secrets at `secret/foo` and `secret/foo/bar`.
	<pre>{
	"auth":null,
	"data":{
	"foo":"bar"
	},
	"lease_duration":2764800,
	"lease_id":"",
	"renewable":false
	}</pre>

Deleting a Secret

API	`/secret`
Description	Deletes the secret at the specified location.
Method	`DELETE`
URL	/secret/<path> Where, `<path>` is the root path to your vault instance plus any sub-path that you create.
Parameters	None

Header	`X-Vault-Token: <token>`
Returns	A `204` response code.

APIs: One Time Secret Sharing

Vault service provides one time secret sharing capability by wrapping the specified value.

Wrapping a Value in a Response-Wrapped Token

API	`/sys/wrapping/wrap`
Description	Wraps the user-specified data inside a response-wrapped token.
Method	`POST`
URL	`/sys/wrapping/wrap`
Parameters	`:any (map<string|string>: nil)` (required) Specify keys:value pairs in a JSON object. The exact set of given parameters is contained in the wrapped response.
Header	`X-Vault-Token: <token>`

Sample Payload	<pre>{
	  "foo": "bar",
	  "zip": "zap"
	}</pre>

Sample Request	<pre>$ curl \
	    --header "X-Vault-Token: ..." \
	    --request POST \
	    --data @payload.json \
	    https://vault.rocks/v1/sys/wrapping/wrap</pre>

Returns	<pre>{
	  "request_id": "",
	  "lease_id": "",
	  "lease_duration": 0,
	  "renewable": false,
	  "data": null,
	  "warnings": null,
	  "wrap_info": {
	    "token": "fb79b9d3-d94e-9eb6-4919-c559311133d6",
	    "ttl": 300,
	    "creation_time":
	"2016-09-28T14:41:00.56961496-04:00",
	    "wrapped_accessor": ""
	  }
	}</pre>

Returning Wrapping Token Properties

API	`/sys/wrapping/lookup`
Description	Returns wrapping token properties.
Method	`POST`
URL	`/sys/wrapping/lookup`
Parameters	`token` (required) Specifies the wrapping token Id.
Header	`X-Vault-Token: <token>`
Sample Payload	`{ "token": "abcd1234" }`
Sample Request	`$ curl \ --header "X-Vault-Token: ..." \ --request POST \ --data @payload.json \ https://vault.rocks/v1/sys/wrapping/lookup`
Returns	`{ "request_id": "481320f5-fdf8-885d-8050-65fa767fd19b", "lease_id": "", "lease_duration": 0, "renewable": false, "data": { "creation_time": "2016-09-28T14:16:13.07103516-04:00", "creation_ttl": 300 }, "warnings": null }`

Re-wrapping a Response-Wrapped Token

API	`/sys/wrapping/rewrap`
Description	Re-wraps a response-wrapped token. The new token uses the same creation TTL as the original token and contains the same response. The old token is invalidated. This API is useful when you need to store a secret for a long time in a response-wrapped token that requires rotation.
Method	`POST`
URL	`/sys/wrapping/rewrap`
Parameters	`token` (required) Specifies the wrapping token Id.

Header	`X-Vault-Token: <token>`

Sample Payload	<pre>{
	  "token": "abcd1234"
	}</pre>

Sample Request	<pre>$ curl \
	    --header "X-Vault-Token: ..." \
	    --request POST \
	    --data @payload.json \
	    https://vault.rocks/v1/sys/wrapping/rewrap</pre>

Returns	<pre>{
	  "request_id": "",
	  "lease_id": "",
	  "lease_duration": 0,
	  "renewable": false,
	  "data": null,
	  "warnings": null,
	  "wrap_info": {
	    "token": "3b6f1193-0707-ac17-284d-e41032e74d1f",
	    "ttl": 300,
	    "creation_time":
	"2016-09-28T14:22:26.486186607-04:00",
	    "wrapped_accessor": ""
	  }
	}</pre>

Unwrapping a Wrapped Response

API	`/sys/wrapping/unwrap`
Description	Returns the original response inside the given wrapping token. The API validates the token, returns the original value, and ensures that the response is logged for audit purpose.
Method	`POST`
URL	`/sys/wrapping/unwrap`
Parameters	`token` (required) Specifies the wrapping token Id. This parameter is not required if you use a wrapping token as the client token in the API call. It is required if you use a different token with permissions to access this endpoint. **Note:** Do not use the token parameter along with wrapping token as client token. This is considered double use of the token. In such a case, the Vault service revokes the wrapping token and you cannot lookup the value.
Header	`X-Vault-Token: <token>`

Sample Payload	<pre>{
	  "token": "abcd1234"
	}</pre>

Sample Request	<pre>$ curl \
	    --header "X-Vault-Token: ..." \
	    --request POST \
	    --data @payload.json \
	    https://vault.rocks/v1/sys/wrapping/unwrap</pre>

Returns	<pre>{
	  "request_id": "8e33c808-f86c-cff8-f30a-
	fbb3ac22c4a8",
	  "lease_id": "",
	  "lease_duration": 2592000,
	  "renewable": false,
	  "data": {
	    "zip": "zap"
	  },
	  "warnings": null
	}</pre>

Managing Tokens for Your Vault Instance

About Managing Tokens

Vault service creates a token when you bind your application to the Vault service instance or create a service key. You use this token to access your Vault service instance. You can use Vault service REST APIs to manage these tokens. You can use the APIs to lookup the token details, renew the token, or revoke a token.

The default time to live (TTL) for a Vault service instance token is 32 days. You can generate tokens with specific TTL when you either bind your application to your Vault service instance or by creating a new service key.

You can use Vault service API to perform the following tasks:

- Lookup Token Details of your Vault Service Instance on page 13
- Renew the Token of your Vault Service Instance on page 13
- Revoke the Token of your Vault Service Instance on page 14

In addition, you can also:

- Rotate the token for your Vault service instance to get a new token.
- Specify duration of the token for your Vault instance.
- Add a read-only policy to the token.

APIs: Managing Tokens

Lookup Token Details of your Vault Service Instance

API	`/auth/token/lookup-self`
Description	Lookup details of the token for your Vault service instance.
Method	`GET`

URL	`/auth/token/lookup-self`
	For example, you can construct your URL to perform this operation as follows:
	`https://predix-vault-asv.gecis.io/v1/auth/token/lookup-self`
	Note: You can get the path to Vault service from your VCAP_SERVICES environment variable.

Parameters	None
Header	`X-Vault-Token: <token>`

Returns	<pre>{
	"data":{
	"accessor":"REDACTED",
	"creation_time":1484093665,
	"creation_ttl":3600,
	"display_name":"github-armon",
	"explicit_max_ttl":0,
	"id":"ClientToken",
	"meta":
	{"user":"armon","organization":"hashicorp"},
	"num_uses":0,
	"orphan":true,
	"path":"auth/github/login",
	"policies":["web","stage"],
	"renewable":true,
	"ttl":3655
	}
	}</pre>

Renew the Token of your Vault Service Instance

API	`/auth/token/renew-self`
Description	Renews the lease associated with the calling token. This prevents the expiration and automatic revocation of the token. You can renew a token only if there is a lease associated with it.
Method	`POST`
URL	`/auth/token/renew-self`

	For example, you can construct your URL to perform this operation as follows: `https://predix-vault-asv.gecis.io/v1/auth/token/renew-self` **Note:** You can get the path to Vault service from your VCAP_SERVICES environment variable.
Parameters	• `increment` (optional) Specify the increment value for the token lease.
Header	`X-Vault-Token: <token>`
Returns	`{ "auth":{ "client_token":"ABCD", "policies":["web","stage"], "metadata":{"user":"armon"}, "lease_duration":3600, "renewable":true, } }`

Revoke the Token of your Vault Service Instance

API	`/auth/token/revoke-self`
Description	Revokes the token that you use for your Vault service instance. When the token is revoked, all secrets generated with it are also revoked.
Method	`POST`

URL	`/auth/token/revoke-self`
	For example, you can construct your URL to perform this operation as follows:
	`https://predix-vault-asv.gecis.io/v1/auth/token/revoke-self`
	Note: You can get the path to Vault service from your VCAP_SERVICES environment variable.

Parameters	None
Header	`X-Vault-Token: <token>`
Returns	A `204` response code.

Rotating the Token for your Vault Service Instance

You can rotate the token for your Vault Service Instance to get a new value for your token.

Procedure

1. To rotate the token, you can unbind your application from your Vault service instance and bind it again.
   For more information on binding your application, see Binding an Application to the Vault Service Instance on page 3.

2. Optionally, you can create a new service key for your Vault service. The new service key contains a new token value.

   For more information on service key, see Create a Service Key for Vault Service on page 4.

Specifying Duration of the Token for Your Vault Service Instance

The default time to live (TTL) for a Vault service instance token is 32 days. You can specify the duration of the tokens when you either bind your application to your Vault service instance or by creating a new service key.

About This Task

You can either generate a token that lives forever (perpetual) or a token with specific time to live (TTL).

Procedure

1. You can specify custom parameters when you bind your application to your Vault service instance.
   - For a token with unlimited duration, use the following command:

     ```
     cf bind-service <your_app_name> <vault_instance_name> -c
     '{"token_type" : "perpetual"}'
     ```

   - For token with specific TTL value, use the following command:

     ```
     cf bind-service <your_app_name> <vault_instance_name> -c
     '{"token_type" : "periodic", "token_ttl" : "<duration>"}'
     ```

   For more information on binding your application, see Binding an Application to the Vault Service Instance on page 3.
2. Optionally, you can create a new service key with custom parameters for your Vault service. The new service key contains a new token value.
   - For a token with unlimited duration, use the following command:

     ```
     cf create-service-key <vault_instance_name> <service-key-name> -c
     '{"token_type" : "perpetual"}'
     ```

   - For token with specific TTL value, use the following command:

     ```
     cf create-service-key <vault_instance_name> <service-key-name> -c
     '{"token_type" : "periodic", "token_ttl" : "<duration>"}'
     ```

   For more information on service key, see Create a Service Key for Vault Service on page 4.

Adding a Read-Only Policy to the Token

When you add a read-only policy to a token, a user can read the token details but cannot create, update or delete the token.

Procedure

1. You can specify custom parameters when you bind your application to your Vault service instance.

   ```
   cf bind-service <your_app_name> <vault_instance_name> -c '{"
   token_read_only: true "}'
   ```

   For more information on binding your application, see Binding an Application to the Vault Service Instance on page 3.

2. Optionally, you can create a new service key with custom parameters for your Vault service. The new service key contains a new token value.

```
cf create-service-key <vault_instance_name> <service-key-name> -c
'{" token_read_only: true "}'
```

For more information on service key, see Create a Service Key for Vault Service on page 4.

Data Encryption

About Data Encryption

You can use Vault service for encryption and decryption of data used or generated by applications.

Application developers can use Vault to encrypt and store data in a primary data store and do not need to develop additional functionality for encryption and decryption of their data. Vault service generates logs for every encryption and decryption event. Therefore all data encryption and decryption activity is recorded.

You can use Vault service API to perform the following tasks:

APIs: Data Encryption

You can use APIs to use the data encryption feature of Vault service.

Creating New Encryption Key

API	`/encryption/keys/`
Description	Creates a new named encryption key of the specified type. The values set here cannot be changed after key creation.
Method	`POST`
URL	`/encryption/keys/<name>`

Parameters	• `type` (required) Specifies the type of key to create. You can create the following types of keys: ◦ `aes256-gcm96` This is a AES-256 key wrapped with GCM using a 12-byte nonce size (symmetric). This is the default type. ◦ `ecdsa-p256` This is a ECDSA key using the P-256 elliptic curve (asymmetric). • `derived` (optional) Set it to `true` to indicate that the key derivation must be used for encryption or decryption. If you use this parameter, you must provide a context that is used for key derivation in all encrypt or decrypt requests to this named key. By default, it is set to `false`. • `convergent_encryption` (optional) Set it to `true` for the key to support convergent encryption. In convergent encryption, the plain text creates the same cipher text. This parameter requires the `derived` parameter to be set to true. When enabled, each encryption, decryption, rewrap, or datakey operation derives a `nonce` value instead of a random value. The key space for the nonce value is 96 bit. The default value is `false`. **Note:** For complete security, you must generate unique nonce values with a given context • `exportable` (optional) Set it to `true` to indicate that the key is exportable. This option is required only when you need to export out the value of the key. For more information, see Retrieving a Specific Type of Key on page 20. The default value is `false`.
Header	`X-Vault-Token: <token>`
Returns	A `204` response code.

Retrieving Information of an Encryption Key

API	`/encryption/keys/`
Description	Returns information about a specified encryption key. In the returned value, the `keys` object shows the creation time of each key version. The returned values are not the keys themselves. The returned information varies by the type of key. For example, an asymmetric key returns its public key in a standard format for the type.
Method	`GET`
URL	`/encryption/keys/<name>`
Parameters	None
Header	`X-Vault-Token: <token>`
Returns	`{` `"data":{`

	<pre>"type":"aes256-gcm96",
	"deletion_allowed":false,
	"derived":false,
	"exportable":false,
	"keys":{
	"1":1442851412
	},
	"min_decryption_version":0,
	"name":"foo",
	"supports_encryption":true,
	"supports_decryption":true,
	"supports_derivation":true,
	"supports_signing":false
	}
	}</pre>

Retrieving a List of Keys

API	`/encryption/keys/`
Description	Returns a list of names of the keys.
Method	`LIST/GET`
URL	`/encryption/keys` (LIST) or `/encryption/keys?list=true` (GET)
Parameters	None
Header	`X-Vault-Token: <token>`

Returns	<pre>{
	"data":{
	"keys":["foo","bar"]
	},
	"lease_duration":0,
	"lease_id":"",
	"renewable":false
	}</pre>

Deleting a Specified Encryption Key

API	`/encryption/keys/`
Description	Deletes a specified encryption key. After this operation, the key cannot be used to decrypt any data encrypted using this key. **Note:** You must set the `deletion_allowed` parameter in the `/config` endpoint of the key.
Method	`DELETE`
URL	`/encryption/keys/<name>`
Parameters	None
Header	`X-Vault-Token: <token>`

Returns	A `204` response code.

Configuring a Specified Key

API	`/encryption/keys/config`
Description	Specify configuration values for a specified key. The configuration values are used during a read operation on the specified key.
Method	`POST`
URL	`/encryption/keys/<name>/config`
Parameters	• `min_decryption_version` (optional) Specify the minimum version of ciphertext allowed to be decrypted. This value can be used for the following: ◦ As part of a key rotation policy to prevent decryption of the old copies of ciphertext. ◦ For key signatures to control the minimum version of signature used for key verification. ◦ For keyed-hash message authentication code (HMAC) to control the minimum version of a key allowed to be used as the key for the HMAC function. The default value is `0`. • `deletion_allowed` (optional) Set to `true` to allow the key to be deleted. The default value is `false`.
Header	`X-Vault-Token: <token>`
Returns	A `204` response code.

Rotating a Specified Version of the Key

API	`/encryption/keys/rotate/`
Description	Rotates the version of the specified key. When you rotate a key, any new plaintext request is encrypted with the new version of the key. This parameter is only supported with keys that supports encryption and decryption operations. **Note:** To upgrade ciphertext to be encrypted with the latest version of the key, use the `rewrap` endpoint. See Rewraping the Specified Ciphertext on page 22.
Method	`POST`
URL	`/encryption/keys/<name>/rotate`
Parameters	None
Header	`X-Vault-Token: <token>`
Returns	A `204` response code.

Retrieving a Specific Type of Key

API	`/encryption/export/encryption-key/<name>(/<version>)`
	`/encryption/export/signing-key/<name>(/<version>)`
	`/encryption/export/hmac-key/<name>(/<version>)`

Description	Returns the specified key. The `key` object shows the value of the key for each version. To retrieve a specific version, use the `version` option. To retrieve the current key, use the `latest` option. Depending on the type of key, different information may be returned. To support this operation, the key must be created with the exportable parameter and must have a valid version.
Method	`GET`
URL	`/encryption/export/<key-type>/<name>/<version>`
Parameters	None
Header	`X-Vault-Token: <token>`

Returns	<pre>{
	    "data":{
	        "name":"foo",
	        "keys":{
	        "1":"eyXYGHbTmugUJn6EtYD/yVEoF6pCxm4R/
	cMEutUm3MY=",
	        "2":"Euzymqx6iXjS3/NuGKDCiM2Ev6wdhnU
	+rBiKnJ7YpHE="
	        }
	    }
	}</pre>

Encrypting Plain Text

API	`/encryption/encrypt/`
Description	Encrypts the specified plaintext using the specified key. This operation only supports symmetric keys (`aes256-gcm96`). This path supports the `create` and `update` policy capabilities as follows: if the user has the `create` capability for this endpoint in their policies, and the key does not exist, it will be upserted with default values (whether the key requires derivation depends on whether the context parameter is empty or not). If the user only has `update` capability and the key does not exist, an error will be returned.
Method	`POST`
URL	`/encryption/encrypt/<name>`
Parameters	• `plaintext` (required) Specify the base64-encoded plain-text value that you need to encrypt. • `context` (optional)

	Specify the base64-encoded context for deriving the key. This parameter is required if you set the `convergent_encryption` parameter to `true` while generating a new key. • `nonce` (optional) Specify the base64-encoded nonce value. The value must be exactly 96 bits (12 bytes) long. **Note:** You must never reuse a nonce value or the generated encryption key for any given context. • `batch_input` (optional) Specify a list of items to be encrypted in a single batch. If this parameter is set, the parameters `plaintext`, `context` and `nonce` are ignored. Use the following format for the input: `[ { "context":"c2FtcGxlY29udGV4dA==", "plaintext":"dGhlIHF1aWNrIGJyb3duIGZveA==" }, { "context":"YW5vdGhlcnNhbXBsZWNvbnRleHQ=", "plaintext":"dGhlIHF1aWNrIGJyb3duIGZveA==" }, ... ]` • `type` (optional) This parameter is required when encryption key is expected to be created. When performing an upsert operation, the type of key to create. Currently, "aes256-gcm96" (symmetric) is the only type supported. Defaults to "aes256-gcm96". • `convergent_encryption` (optional) Set it to `true` for the generated key to support convergent encryption. In convergent encryption, the plain text creates the same cipher text. This parameter is used only when the specified key does not exist and a new key is created. If a new key is generated, it is created with key derivation enabled and will require all requests to carry both a context and 96-bit (12-byte) nonce. When enabled, each encryption operation derives a `nonce` value instead of a random value. Therefore when the same context and nonce are supplied, the same ciphertext is generated. **Note:** For complete security, you must generate unique nonce values with a given context
Header	`X-Vault-Token: <token>`
Returns	`{ "data":{ "ciphertext":"vault:v1:abcdefgh" } }`

Decrypting the Specified Ciphertext

API	`/encryption/decrypt/`
Description	Decrypts the specified ciphertext using the specified key. This operation only supports symmetric keys (`aes256-gcm96`).
Method	`POST`
URL	`/encryption/decrypt/<name>`
Parameters	• `ciphertext` (required) Specify the encrypted ciphertext that you need to decrypt. • `context` (optional) Specify the base64-encoded context for deriving the key. This parameter is required if you set the `convergent_encryption` parameter to `true` while generating a new key. • `nonce` (optional) Specify the base64-encoded nonce value that you specified during encryption. • `batch_input` (optional) Specify a list of items to be decrypted in a single batch. If this parameter is set, the parameters `ciphertext`, `context` and `nonce` are ignored. Use the following format for the input: `[ { "context":"c2FtcGxlY29udGV4dA==", "ciphertext":"vault:v1:/DupSiSbX/ ATkGmKAmhqD0tvukByrx6gmps7d VI=" }, { "context":"YW5vdGhlcnNhbXBsZWNvbnRleHQ=", "ciphertext":"vault:v1:XjsPWPjqPrBi1N2Ms2s1QM798Y yFWnO4TR4ls FA=" }, ... ]`
Header	`X-Vault-Token: <token>`
Returns	`{ "data":{ "plaintext":"dGhlIHF1aWNrIGJyb3duIGZveAo=" } }`

Rewraping the Specified Ciphertext

API	`/encryption/rewrap/`

Description	Rewrap the specified ciphertext using the latest version of the specified key. This parameter does not return plain-text. Therefore you can use this API with untrusted users or scripts.
Method	`POST`
URL	`/encryption/rewrap/<name>`
Parameters	• `ciphertext` (required) Specify the encrypted ciphertext that you need to rewrap. • `context` (optional) Specify the base64-encoded context for deriving the key. This parameter is required if you set the `convergent_encryption` parameter to `true` while generating a new key. • `nonce` (optional) Specify the base64-encoded nonce value that you specified during encryption. • `batch_input` (optional) Specify a list of items to be rewrapped in a single batch. If this parameter is set, the parameters `ciphertext`, `context` and `nonce` are ignored. Use the following format for the input: `[ { "context":"c2FtcGxlY29udGV4dA==", "ciphertext":"vault:v1:/DupSiSbX/ ATkGmKAmhqD0tvukByrx6gmps7d VI=" }, { "context":"YW5vdGhlcnNhbXBsZWNvbnRleHQ=", "ciphertext":"vault:v1:XjsPWPjqPrBi1N2Ms2s1QM798Y yFWnO4TR4ls FA=" }, ... ]`
Header	`X-Vault-Token: <token>`
Returns	`{ "data":{ "ciphertext":"vault:v2:abcdefgh" } }`

Generating a New High-Entropy Key

API	`/encryption/datakey/`
Description	Generate a new high-entropy key and the value encrypted with the specified key. Optionally, you can return the plain-text of the key as well.
Method	`POST`

URL	`/encryption/datakey/<plaintext|wrapped>/<name>`
Parameters	• `plaintext|wrapped` (path parameter) (required) If you specify `plaintext`, the plain-text key is returned along with the ciphertext. If you specify `wrapped`, only the ciphertext value is returned. • `context` (optional) Specify the base64-encoded context for deriving the key. This parameter is required if you set the `convergent_encryption` parameter to `true` while generating a new key. • `nonce` (optional) Specify the base64-encoded nonce value. The value must be exactly 96 bits (12 bytes) long. **Note:** You must never reuse a nonce value or the generated encryption key for any given context. • `bits` (optional) Specify the number of bits in the requested key. You can specify 128, 256, or 512. The default value is 256.
Header	`X-Vault-Token: <token>`
Returns	`{ "data":{ "plaintext":"dGhlIHF1aWNrIGJyb3duIGZveAo=", "ciphertext":"vault:v1:abcdefgh" } }`

Generating Random Bytes

API	`/encryption/random`
Description	Generates an string of high-quality random bytes of specified length that can be used for cryptographic encryption.
Method	`POST`
URL	`/encryption/random(/<bytes>)`
Parameters	• `bytes` (optional) Specify the number of bytes to return. The default value is 32 (256 bits). You can specify this value either in the request body, or as a part of the URL. The URL format is as follows: `/encryption/random/48` • `format` (optional) Specify the encoding for the output value. You can specify either `hex` or `base64`. The default value is `base64`.
Header	`X-Vault-Token: <token>`

Returns	<pre>{
	"data":{
	"random_bytes":"dGhlIHF1aWNrIGJyb3duIGZveAo="
	}
	}</pre>

Generating Hash of a Specified Data

API	`/encryption/hash`
Description	Generates the hash of given data using the specified algorithm.
Method	`POST`
URL	`/encryption/hash(/<algorithm>)`
Parameters	• `input` (required) Specify the base64-encoded input data. • `algorithm` (optional) Specify the hash algorithm to use. You can specify this value either in the request body, or as a part of the URL. The URL value takes precedence if both are set. You can specify the following algorithms: ◦ `sha2-224` ◦ `sha2-256` ◦ `sha2-384` ◦ `sha2-512` The default value is `sha2-256`. • `format` (optional) Specify the encoding of the output value. You can specify either `hex` or `base64`. The default value is `hex`.
Header	`X-Vault-Token: <token>`

Returns	<pre>{
	"data":{
	"sum":"dGhlIHF1aWNrIGJyb3duIGZveAo="
	}
	}</pre>

Generating the Digest of the Data

API	`/encryption/hmac/`
Description	Generates the digest of the given data using the specified hash algorithm and the specified key. The key can be of the type that is supported by the `encryption` API. A raw key is marshalled into bytes to be used for the HMAC function. If the key is of a type that supports rotation, the latest (current) version of the key is used.

Method	`POST`
URL	`/encryption/hmac/<name>(/<algorithm>)`
Parameters	• `input` (required) Specify the base64-encoded input data. • `algorithm` (optional) Specify the hash algorithm to use. You can specify this value either in the request body, or as a part of the URL. The URL value takes precedence if both are set. You can specify the following algorithms: ◦ `sha2-224` ◦ `sha2-256` ◦ `sha2-384` ◦ `sha2-512` The default value is `sha2-256`. • `format` (optional) Specify the encoding of the output value. You can specify either `hex` or `base64`. The default value is `hex`.
Header	`X-Vault-Token: <token>`
Returns	`{ "data":{ "hmac":"dGhlIHF1aWNrIGJyb3duIGZveAo=" } }`

Generating the Cryptographic Signature of the Specified Data

API	`/encryption/sign/`
Description	Generates the cryptographic signature of the given data using the specified key and the specified hash algorithm. The key must be of a type that supports key signing.
Method	`POST`
URL	`/encryption/sign/<name>(/<algorithm>)`
Parameters	• `input` (required) Specify the base64-encoded input data. • `algorithm` (optional) Specify the hash algorithm to use. You can specify this value either in the request body, or as a part of the URL. The URL value takes precedence if both are set. You can specify the following algorithms: ◦ `sha2-224` ◦ `sha2-256` ◦ `sha2-384` ◦ `sha2-512` The default value is `sha2-256`.

Header	`X-Vault-Token: <token>`

Returns	<pre>{
	    "data":{
	
	"signature":"vault:v1:MEUCIQCyb869d7KWuA0hBM9b5NJrmWzMW3/
	pT+0XYCM9VmGR
	+QIgWWF6ufi4OS2xo1eS2V5IeJQfsi59qeMWtgX0LipxEHI="
	    }
	}</pre>

Validating a Signature

API	`/encryption/verify/`
Description	Validates the provided signature for the given data.
Method	`POST`
URL	`/encryption/verify/<name>(/<algorithm>)`
Parameters	• `input` (required) Specify the base64-encoded input data. • `signature` (required) Specify the signature output from the `/transit/sign`. You must specify either this parameter or the `hmac` parameter. • `hmac` (required) Specify the signature output from the `/encryption/hmac` function. Either this must be supplied or `signature` must be supplied. • `algorithm` (optional) Specify the hash algorithm to use. You can specify this value either in the request body, or as a part of the URL. The URL value takes precedence if both are set. You can specify the following algorithms: ◦ `sha2-224` ◦ `sha2-256` ◦ `sha2-384` ◦ `sha2-512` The default value is `sha2-256`.
Header	`X-Vault-Token: <token>`

Returns	<pre>{
	    "data":{
	        "valid":true
	    }
	}</pre>

Using Vault Dashboard

Accessing Vault Dashboard

Vault Dashboard provides user interface for using the Vault service.

Before You Begin

- Create an instance of Vault service.
- Retrieve the token for your Vault instance. To retrieve the token, you can one of the following procedure:
  - Bind your application to your instance of Vault service and view the details in the VCAP_SERVICES environment variable.
  - Create a service key for your instance and retrieve the credentials by reading the service key.

Procedure

1. Retrieve the Dashboard URL. The Dashboard URL is part of your service instance details. Use the following command to view your instance details:

   ```
   cf service <vault_service_instance>
   ```

   The `Dashboard` parameter shows the Vault Dashboard details.
2. In a browser window, specify the Vault Dashboard URL.

   The Vault Dashboard login page is displayed.
3. In the login page, specify the Token for your Vault service instance.

   The Vault dashboard is displayed.

Managing Paths and Secrets Using Vault Dashboard

Vault dashboard displays the path information for an instance of Vault service. You can add, edit or delete paths and secrets in your Vault service instance. The secrets are stored as key value pairs.

About This Task

You can either use REST APIs to manage the paths or use the Vault dashboard UI. For more information on using the APIs, see Managing Paths and Secrets.

The following procedure shows managing paths and secrets using the Vault dashboard.

Procedure

1. Log in to Vault Dashboard.

   For more details see Accessing Vault Dashboard on page 28.
2. To add a new path, click on the plus sign (+) next to the home folder in Paths pane.
3. Specify the name of the sub-path and click the check mark next to it.

   The default path location is equivalent to the vault instance URL. You can retrieve the Vault service instance URL from the VCAP_SERVICES environment variable after binding it to an application or by creating a service key. The sub-path that you specify is then appended to the default location.

   For example, if your default path was `https://predix-vault-asv-sb.gecis.io/v1/secret/0e90b8cf-e023-4f56-9ad0-69b9a09f9914/f3eaa2a7/39ba8924-a57e-47ec`

and you specify the sub-path as time-series, you new path is `https://predix-vault-asv-sb.gecis.io/v1/secret/0e90b8cf-e023-4f56-9ad0-69b9a09f9914/f3eaa2a7/39ba8924-a57e-47ec/time-series`.

4. To remove a sub-path, click on the minus (-) sign next to the sub-path name.

   When you remove a sub-path, all the key value information stored in that path is deleted as well.
5. Specify the name of the key and the corresponding value in the **Keys** and **Values** fields.

   Note: If you do not specify a key value pair for a path, the path information is not saved.
6. Click **Add**.

Vault Playground

About Vault Playground

Predix Vault service runs in a controlled environment where connectivity can only be established from Predix applications. The Vault Playground provides you an environment to test the Predix Vault features from applications running outside of Predix, for example, your local environment. Using the Vault Playground, you can locally host applications and test them against the Predix Vault APIs.

Note: Secrets stored cannot be retrieved once the token is expired.

You can use the Predix Vault APIs to perform the following tasks:

- Create a Vault token
- Store a secret
- Retrieve a secret
- Retrieve a list of key names
- Delete a secret

Create a Vault Token

API	`/provision`
URL	US-WEST environments: • `https://predix-vault-playground.run.aws-usw02-dev.ice.predix.io/provision` • `https://predix-vault-playground.run.aws-usw02-pr.ice.predix.io/provision`
Description	Creates a Vault token that is valid for 32 days and returns the Vault URL.
Method	`POST`
Header	None
Parameters	None
Response	A `200` response is returned.

Example	**Request** : curl -v --request POST https://predix-vault-playground.run.aws-usw02-dev.ice.predix.io/provision
	
	Response :
	{
	"vaultUrl": "https://predix-vault-playground.run.aws-usw02-dev.ice.predix.io/v1/cubbyhole/",
	"vaultToken": "f72a7ee7-63fe-fbe1-e6fe-e21e63e98011",
	"vaultAccessor": "ab19fc89-4aea-6e5e-630d-cde0c7ed8403",
	"renewTokenUrl": "https://predix-vault-playground.run.aws-usw02-dev.ice.predix.io/v1/auth/token/renew-self"
	}

Store a Secret

API	`v1/cubbyhole/<path>`
URL	US-WEST environments: • `https://predix-vault-playground.run.aws-usw02-dev.ice.predix.io/v1/cubbyhole/<path>` • `https://predix-vault-playground.run.aws-usw02-pr.ice.predix.io/v1/cubbyhole/<path>` Where, `<path>` is the root path to your Vault service instance.
Description	Stores a secret at the specified location
Method	`POST/ PUT`
Request	Specify a key (in JSON format) name paired with an associated value to be stored at the given path.. You can specify multiple key/value pairs.
Header	`X-Vault-Token: <token>` Where, `<token>` is the token that you received when you create a vault token.
Parameters	None
Response	A `204` (No Content) response is returned.

Example	**Request:** curl -v --header "X-Vault-Token:<token>" --request POST --data '{"zip":"zap","foo":"bar"}' https://predix-vault-playground.run.aws-usw02-dev.ice.predix.io/v1/cubbyhole/<path> **Response:** No content

Retrieve a Secret

API	v1/cubbyhole/<path>
URL	US-WEST environments: • https://predix-vault-playground.run.aws-usw02-dev.ice.predix.io/v1/cubbyhole/<path> • https://predix-vault-playground.run.aws-usw02-pr.ice.predix.io/v1/cubbyhole/<path> Where, <path> is the root path to your Vault instance.
Description	Retrieve the secret at the specified location
Method	GET
Request	None
Header	X-Vault-Token: <token> Where, <token> is the toke that you received when you create a vault token.
Response	A 200 response is returned.
Example	**Request:** curl -v --header "X-Vault-Token:<token>" --request GET https://predix-vault-playground.run.aws-usw02-dev.ice.predix.io/v1/cubbyhole/<Path> **Response:** {

	"request_id": "954416f9-af97-d266-722e-f68fc773a718",
	"lease_id": "",
	"renewable": false,
	"lease_duration": 0,
	"data": {
	"foo": "bar",
	"zip": "zap"
	},
	"wrap_info": null,
	"warnings": null,
	"auth": null
	}

Retrieve a List of Key Names

API	`v1/cubbyhole/<path>`
URL	US-WEST environments: • `https://predix-vault-playground.run.aws-usw02-dev.ice.predix.io/v1/cubbyhole` • `https://predix-vault-playground.run.aws-usw02-pr.ice.predix.io/v1/cubbyhole/<path>` Where, `<path>` is the root path to your vault instance.
Description	Returns a list of key names at the specified location.
Method	`LIST`
Request	None
Header	`X-Vault-Token: <token>` Where, `<token>` is the toke that you received when you create a vault token.
Response	A `200` response is returned.
Example	**Request:** `curl -v --header "X-Vault-Token:<token>" --request LIST https://predix-vault-playground.run.aws-usw02-dev.ice.predix.io/v1/cubbyhole` The example below shows output for a query path of cubbyhole/ when there are secrets at cubbyhole/foo.

	Response:
	{
	"request_id":
	"a6affb42-30eb-64cc-4c62-2798aaf7bef6",
	"lease_id": "",
	"renewable": false,
	"lease_duration": 0,
	"data": {
	"keys": ["myfirstkey"]
	},
	"wrap_info": null,
	"warnings": null,
	"auth": null
	}

Delete a Secret

API	`v1/cubbyhole/<path>`
URL	US-WEST environments: • `https://predix-vault-playground.run.aws-usw02-dev.ice.predix.io/v1/cubbyhole/<path>` • `https://predix-vault-playground.run.aws-usw02-pr.ice.predix.io/v1/cubbyhole/<path>` Where, *<path>* is the root path to your Vault instance.
Description	Deletes a secret at the specified location.
Method	`Delete`
Request	None
Header	`X-Vault-Token: <token>` Where, `<token>` is the toke that you received when you create a vault token.
Parameter	None
Response	A `204 (No Content)` response is returned.
Example	**Request:** `curl -v --header "X-Vault-Token:<token>" --request DELETE`

	https://predix-vault-playground.run.aws-usw02-dev.ice.predix.io/v1/cubbyhole/<path> **Response:** No content

Credential Store and Encryption Vault Service Release Notes

Credential Store and Encryption Vault Service Q1 2018

New Feature

The following new feature was added:

Vault Playground

The Vault Playground provides you an environment to test the Predix Vault features from applications running outside of Predix, for example, your local environment. Using the Vault Playground, you can locally host applications and test them against the Predix Vault APIs.

For more information, see Vault Playground.

Credential Store and Encryption Vault Service Q3 2017

New Features

The following new features were added.

Specifying Time to Live (TTL) for a Token

The default TTL for a Vault service instance token is 32 days. You can now specify the duration of the tokens when you either bind your application to your Vault service instance or by creating a new service key.

For more information, see Specifying Duration of the Token for Your Vault Service Instance on page 15.

Adding a Read-Only Policy to the Token

You can now add a read-only policy to the tokens when you either bind your application to your Vault service instance or when you create a new service key. When you add a read-only policy to a token, a user can read the token details but cannot create, update or delete the token.

For more information, see Adding a Read-Only Policy to the Token on page 15.

Credential Store and Encryption Vault Service Q1 2017

New Features

The following new features were added.

Data Encryption

You can now use the Vault service for encryption and decryption of data used or generated by applications.

For more information, see About Data Encryption on page 16.

One Time Secret Sharing Capability

Vault service now provides APIs to support one time secret sharing capability by wrapping the secret value.

For more information, see About Managing Paths and Secrets on page 6.

Enhancements

The following new enhancement was added.

Availability of Service in Multiple Data Centers

The Vault service is now available in the Predix US-East domain (`https://predix-io.run.asv-pr.ice.predix.io`) and the Predix US-West domain (`https://api.system.aws-usw02-pr.ice.predix.io`).