AppRole Auth Method (API)

This is the API documentation for the Vault AppRole auth method. For general information about the usage and operation of the AppRole method, please see the Vault AppRole method documentation (/docs/auth/approle.html).

This documentation assumes the AppRole method is mounted at the `/auth/approle` path in Vault. Since it is possible to enable auth methods at any location, please update your API calls accordingly.

List Roles

This endpoint returns a list the existing AppRoles in the method.

Method	Path	Produces
`LIST`	`/auth/approle/role`	`200 application/json`

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request LIST \
    http://127.0.0.1:8200/v1/auth/approle/role
```

Sample Response

```
{
  "auth": null,
  "warnings": null,
  "wrap_info": null,
  "data": {
    "keys": [
      "dev",
      "prod",
      "test"
    ]
  },
  "lease_duration": 0,
  "renewable": false,
  "lease_id": ""
}
```

Create New AppRole

Creates a new AppRole or updates an existing AppRole. This endpoint supports both `create` and `update` capabilities. There can be one or more constraints enabled on the role. It is required to have at least one of them enabled while creating or updating a role.

Method	Path	Produces
`POST`	`/auth/approle/role/:role_name`	`204 (empty body)`

Parameters

- `role_name` `(string: <required>)` - Name of the AppRole.
- `bind_secret_id` `(bool: true)` - Require `secret_id` to be presented when logging in using this AppRole.
- `bound_cidr_list` `(array: [])` - Comma-separated string or list of CIDR blocks; if set, specifies blocks of IP addresses which can perform the login operation.
- `policies` `(array: [])` - Comma-separated list of policies set on tokens issued via this AppRole.
- `secret_id_num_uses` `(integer: 0)` - Number of times any particular SecretID can be used to fetch a token from this AppRole, after which the SecretID will expire. A value of zero will allow unlimited uses.
- `secret_id_ttl` `(string: "")` - Duration in either an integer number of seconds (`3600`) or an integer time unit (`60m`) after which any SecretID expires.
- `token_num_uses` `(integer: 0)` - Number of times issued tokens can be used. A value of 0 means unlimited uses.
- `token_ttl` `(string: "")` - Duration in either an integer number of seconds (`3600`) or an integer time unit (`60m`) to set as the TTL for issued tokens and at renewal time.
- `token_max_ttl` `(string: "")` - Duration in either an integer number of seconds (`3600`) or an integer time unit (`60m`) after which the issued token can no longer be renewed.
- `period` `(string: "")` - Duration in either an integer number of seconds (`3600`) or an integer time unit (`60m`). If set, the token generated using this AppRole is a *periodic* token; so long as it is renewed it never expires, but the TTL set on the token at each renewal is fixed to the value specified here. If this value is modified, the token will pick up the new value at its next renewal.

Sample Payload

```
{
  "token_ttl": "10m",
  "token_max_ttl": "15m",
  "policies": [
    "default"
  ],
  "period": 0,
  "bind_secret_id": true
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/auth/approle/role/application1
```

Read AppRole

Reads the properties of an existing AppRole.

Method	Path	Produces
`GET`	`/auth/approle/role/:role_name`	`200 application/json`

Parameters

- `role_name (string: <required>)` - Name of the AppRole.

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    http://127.0.0.1:8200/v1/auth/approle/role/application1
```

Sample Response

```
{
  "auth": null,
  "warnings": null,
  "wrap_info": null,
  "data": {
    "token_ttl": 1200,
    "token_max_ttl": 1800,
    "secret_id_ttl": 600,
    "secret_id_num_uses": 40,
    "policies": [
      "default"
    ],
    "period": 0,
    "bind_secret_id": true,
    "bound_cidr_list": []
  },
  "lease_duration": 0,
  "renewable": false,
  "lease_id": ""
}
```

Delete AppRole

Deletes an existing AppRole from the method.

Method	Path	Produces
`DELETE`	`/auth/approle/role/:role_name`	`204 (empty body)`

Parameters

- `role_name` `(string: <required>)` - Name of the AppRole.

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request DELETE \
    http://127.0.0.1:8200/v1/auth/approle/role/application1
```

Read AppRole Role ID

Reads the RoleID of an existing AppRole.

Method	Path	Produces
`GET`	`/auth/approle/role/:role_name/role-id`	`200 application/json`

Parameters

- `role_name` `(string: <required>)` - Name of the AppRole.

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    http://127.0.0.1:8200/v1/auth/approle/role/application1/role-id
```

Sample Response

```
{
  "auth": null,
  "warnings": null,
  "wrap_info": null,
  "data": {
    "role_id": "e5a7b66e-5d08-da9c-7075-71984634b882"
  },
  "lease_duration": 0,
  "renewable": false,
  "lease_id": ""
}
```

Update AppRole Role ID

Updates the RoleID of an existing AppRole to a custom value.

Method	Path	Produces
`POST`	`/auth/approle/role/:role_name/role-id`	`204 (empty body)`

Parameters

- `role_name` `(string: <required>)` - Name of the AppRole.
- `role_id` `(string: <required>)` - Value to be set as RoleID.

Sample Payload

```
{
  "role_id": "custom-role-id"
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/auth/approle/role/application1/role-id
```

Sample Response

```
{
  "auth": null,
  "warnings": null,
  "wrap_info": null,
  "data": {
    "role_id": "e5a7b66e-5d08-da9c-7075-71984634b882"
  },
  "lease_duration": 0,
  "renewable": false,
  "lease_id": ""
}
```

Generate New Secret ID

Generates and issues a new SecretID on an existing AppRole. Similar to tokens, the response will also contain a `secret_id_accessor` value which can be used to read the properties of the SecretID without divulging the SecretID itself, and also to delete the SecretID from the AppRole.

Method	Path	Produces
POST	/auth/approle/role/:role_name/secret-id	200 application/json

Parameters

- `role_name` `(string: <required>)` - Name of the AppRole.
- `metadata` `(string: "")` - Metadata to be tied to the SecretID. This should be a JSON-formatted string containing the metadata in key-value pairs. This metadata will be set on tokens issued with this SecretID, and is logged in audit logs *in plaintext*.
- `cidr_list` `(array: [])` - Comma separated string or list of CIDR blocks enforcing secret IDs to be used from specific set of IP addresses. If `bound_cidr_list` is set on the role, then the list of CIDR blocks listed here should be a subset of the CIDR blocks listed on the role.

Sample Payload

```
{
  "metadata": "{ \"tag1\": \"production\" }"
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/auth/approle/role/application1/secret-id
```

Sample Response

```
{
  "auth": null,
  "warnings": null,
  "wrap_info": null,
  "data": {
    "secret_id_accessor": "84896a0c-1347-aa90-a4f6-aca8b7558780",
    "secret_id": "841771dc-11c9-bbc7-bcac-6a3945a69cd9"
  },
  "lease_duration": 0,
  "renewable": false,
  "lease_id": ""
}
```

List Secret ID Accessors

Lists the accessors of all the SecretIDs issued against the AppRole. This includes the accessors for "custom" SecretIDs as well.

Method	Path	Produces
`LIST`	`/auth/approle/role/:role_name/secret-id`	`200 application/json`

Parameters

- `role_name` `(string: <required>)` - Name of the AppRole.

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request LIST \
    http://127.0.0.1:8200/v1/auth/approle/role/application1/secret-id
```

Sample Response

```
{
  "auth": null,
  "warnings": null,
  "wrap_info": null,
  "data": {
    "keys": [
      "ce102d2a-8253-c437-bf9a-aceed4241491",
      "a1c8dee4-b869-e68d-3520-2040c1a0849a",
      "be83b7e2-044c-7244-07e1-47560ca1c787",
      "84896a0c-1347-aa90-a4f6-aca8b7558780",
      "239b1328-6523-15e7-403a-a48038cdc45a"
    ]
  },
  "lease_duration": 0,
  "renewable": false,
  "lease_id": ""
}
```

Read AppRole Secret ID

Reads out the properties of a SecretID.

Method	Path	Produces
POST	/auth/approle/role/:role_name/secret-id/lookup	200 application/json

Parameters

- `role_name` `(string: <required>)` - Name of the AppRole.
- `secret_id` `(string: <required>)` - Secret ID attached to the role.

Sample Payload

```
{
  "secret_id": "84896a0c-1347-aa90-a4f6-aca8b7558780"
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --payload @payload.json \
    http://127.0.0.1:8200/v1/auth/approle/role/application1/secret-id/lookup
```

Destroy AppRole Secret ID

Destroy an AppRole secret ID.

Method	Path	Produces
`POST`	`/auth/approle/role/:role_name/secret-id/destroy`	`204 (empty body)`

Parameters

- `role_name` `(string: <required>)` - Name of the AppRole.
- `secret_id` `(string: <required>)` - Secret ID attached to the role.

Sample Payload

```
{
  "secret_id": "84896a0c-1347-aa90-a4f6-aca8b7558780"
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --payload @payload.json \
    http://127.0.0.1:8200/v1/auth/approle/role/application1/secret-id/destroy
```

Read AppRole Secret ID Accessor

Reads out the properties of a SecretID.

Method	Path	Produces
`POST`	`/auth/approle/role/:role_name/secret-id-accessor/lookup`	`200 application/json`

Parameters

- `role_name` `(string: <required>)` - Name of the AppRole.
- `secret_id_accessor` `(string: <required>)` - Secret ID accessor attached to the role.

Sample Payload

```
{
  "secret_id_accessor": "84896a0c-1347-aa90-a4f6-aca8b7558780"
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --payload @payload.json \
    http://127.0.0.1:8200/v1/auth/approle/role/application1/secret-id-accessor/lookup
```

Destroy AppRole Secret ID Accessor

Destroy an AppRole secret ID by its accessor.

Method	Path	Produces
`POST`	`/auth/approle/role/:role_name/secret-id-accessor/destroy`	`204 (empty body)`

Parameters

- `role_name` `(string: <required>)` - Name of the AppRole.
- `secret_id_accessor` `(string: <required>)` - Secret ID accessor attached to the role.

Sample Payload

```
{
  "secret_id_accessor": "84896a0c-1347-aa90-a4f6-aca8b7558780"
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --payload @payload.json \
    http://127.0.0.1:8200/v1/auth/approle/role/application1/secret-id-accessor/destroy
```

Create Custom AppRole Secret ID

Assigns a "custom" SecretID against an existing AppRole. This is used in the "Push" model of operation.

Method	Path	Produces
`POST`	`/auth/approle/role/:role_name/custom-secret-id`	`200 application/json`

Parameters

- `role_name` `(string: <required>)` - Name of the AppRole.
- `secret_id` `(string: <required>)` - SecretID to be attached to the Role.
- `metadata` `(string: "")` - Metadata to be tied to the SecretID. This should be a JSON-formatted string containing the metadata in key-value pairs. This metadata will be set on tokens issued with this SecretID, and is logged in audit logs *in plaintext*.
- `cidr_list` `(array: [])` - Comma separated string or list of CIDR blocks enforcing secret IDs to be used from specific set of IP addresses. If `bound_cidr_list` is set on the role, then the list of CIDR blocks listed here should be a subset of the CIDR blocks listed on the role.

Sample Payload

```
{
  "secret_id": "testsecretid"
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/auth/approle/role/application1/custom-secret-id
```

Sample Response

```
{
  "auth": null,
  "warnings": null,
  "wrap_info": null,
  "data": {
    "secret_id_accessor": "84896a0c-1347-aa90-a4f6-aca8b7558780",
    "secret_id": "testsecretid"
  },
  "lease_duration": 0,
  "renewable": false,
  "lease_id": ""
}
```

Login With AppRole

Issues a Vault token based on the presented credentials. `role_id` is always required; if `bind_secret_id` is enabled (the default) on the AppRole, `secret_id` is required too. Any other bound authentication values on the AppRole (such as client IP CIDR) are also evaluated.

Method	Path	Produces
`POST`	`/auth/approle/login`	`200 application/json`

Parameters

- `role_id` `(string: <required>)` - RoleID of the AppRole.
- `secret_id` `(string: <required>)` - SecretID belonging to AppRole.

Sample Payload

```
{
  "role_id": "59d6d1ca-47bb-4e7e-a40b-8be3bc5a0ba8",
  "secret_id": "84896a0c-1347-aa90-a4f6-aca8b7558780"
}
```

Sample Request

```
$ curl \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/auth/approle/login
```

Sample Response

```
{
  "auth": {
    "renewable": true,
    "lease_duration": 1200,
    "metadata": null,
    "policies": [
      "default"
    ],
    "accessor": "fd6c9a00-d2dc-3b11-0be5-af7ae0e1d374",
    "client_token": "5b1a0318-679c-9c45-e5c6-d1b9a9035d49"
  },
  "warnings": null,
  "wrap_info": null,
  "data": null,
  "lease_duration": 0,
  "renewable": false,
  "lease_id": ""
}
```

Read, Update, or Delete AppRole Properties

Updates the respective property in the existing AppRole. All of these parameters of the AppRole can be updated using the `/auth/approle/role/:role_name` endpoint directly. The endpoints for each field is provided separately to be able to delegate specific endpoints using Vault's ACL system.

Method	Path	Produces
GET/POST/DELETE	/auth/approle/role/:role_name/policies	200/204
GET/POST/DELETE	/auth/approle/role/:role_name/secret-id-num-uses	200/204
GET/POST/DELETE	/auth/approle/role/:role_name/secret-id-ttl	200/204
GET/POST/DELETE	/auth/approle/role/:role_name/token-ttl	200/204
GET/POST/DELETE	/auth/approle/role/:role_name/token-max-ttl	200/204
GET/POST/DELETE	/auth/approle/role/:role_name/bind-secret-id	200/204
GET/POST/DELETE	/auth/approle/role/:role_name/bound-cidr-list	200/204
GET/POST/DELETE	/auth/approle/role/:role_name/period	200/204

Refer to `/auth/approle/role/:role_name` endpoint.