KeySafe 5

KeySafe 5 v1.7.0 API documentation

8 April 2026

Table of Contents

The following API documentation is available for nShield KeySafe 5:

- Agent Management API Documentation
- CodeSafe Management API Documentation
- HSM Management API Documentation
- Licence Management API Documentation
- Monitoring Management API Documentation
- Security World Management API Documentation

You may use tools such as Swagger to generate client SDKs from these OpenAPI Specifications.

1. KeySafe 5 v1.7.0 API Examples

Examples for the nShield KeySafe 5 API are available in API Examples.

2. API Examples

The following examples are indicative only and are provided as a general guide. Before using them in production, you should update the code to more fully suit your environment and needs, for example, adding better error handling.

2.1. nShield HSM feature management

2.1.1. Scenario

As an nShield HSM manager, I want to upload and enable a new HSM feature certificate.

2.1.2. Steps

1. Upload feature file: `POST /mgmt/v1/feature-certificates`
2. Enable feature: `PATCH /mgmt/v1/feature-certificates/<featureid>`
3. Clear the HSM if necessary: `PUT /mgmt/v1/hsms/<hsmid>/clear`

2.1.3. Python code example

```
import sys
import base64
import requests

server = "https://192.0.2.1"
# STEP 1 Upload feature file as provided by order or support
with open(sys.argv[1], "rb") as file:
    # Read in feature file and convert to base64 url string
    file_content = base64.urlsafe_b64encode(file.read()).decode()
    # Remove the padding
    file_content = file_content.rstrip("=")

    # Post feature file to KeySafe 5
    response = requests.post(
        server + "/mgmt/v1/feature-certificates",
        json={"file": file_content},
        verify=False,
    )
    if not response:
        raise Exception(f"Non-success status code: {response.status_code}")

    # Location of created resource is in the Location header but can also be deduced from the response body
    featureURI = server + response.headers["Location"]
    clearRequired = response.json()["clearRequired"]

    # STEP 2 Enable feature
    response = requests.patch(featureURI, json={"operation": "enable"}, verify=False)
    if not response:
        raise Exception(f"Non-success status code: {response.status_code}")
```

```
    # STEP 3 Clear HSM if necessary
    if clearRequired:
        clearURI = server + response.json()["meta"]["clear"]
        response = requests.put(clearURI, verify=False)
        if not response:
            raise Exception(f"Non-success status code: {response.status_code}")

    print("Feature applied")
```

2.2. nShield HSM firmware management

2.2.1. Scenario

As an nShield HSM manager, I want to upgrade the firmware on an HSM.

2.2.2. Steps

1. Get the HSM information: `GET /mgmt/v1/hsms/<hsmid>`
2. Search for a suitable firmware image: `GET /mgmt/v1/upgrade-images`
3. Upgrade the firmware: `POST /mgmt/v1/hsms/<hsmid>/firmware`
4. (Optional) Monitor the operation to see when the upgrade is finished.

2.2.3. Python code example

```
import time
import requests

server = "https://192.0.2.1"
hsmURI = "/mgmt/v1/hsms/7cfd8df2-5e88-420e-98c6-e5408ce828d5"
imagesURI = "/mgmt/v1/upgrade-images"

# STEP 1 Get HSM Information
response = requests.get(server + hsmURI, verify=False)
if not response:
    raise Exception(f"Non-success status code: {response.status_code}")

hsmdata = response.json()["data"]["hsm"]

# STEP 2 Search for a suitable platform upgrade image
params = {}
params["type"] = "upgrade-platform"
params["model"] = hsmdata["productModel"].replace("/", ",")
if "hardwareRevision" in hsmdata:
    params["hardwareRevision"] = hsmdata["hardwareRevision"]
params["filter"] = 'version >= "{0}"'.format(hsmdata["versionInfo"]["versionString"])
params["filter"] += " AND moduleInfo.vsn >= {0}".format(hsmdata["vsn"])
if "remoteModuleinfo" in hsmdata:
    params["filter"] += " AND remoteModuleInfo.vsn >= {0}".format(
        hsmdata["remoteModuleInfo"]["vsn"]
    )
```

```
response = requests.get(server + imagesURI, params=params, verify=False)

if not response:
    raise Exception(f"Non-success status code: {response.status_code}")

# STEP 3 if we got a suitable image then upgrade to it
images = response.json()["data"]["images"]
# Upgrade to the first suitable image
if len(images) > 0:
    response = requests.post(
        server + hsmURI + "/firmware",
        json={"dryRun": True, "image": imagesURI + "/" + images[0]["id"]},
        verify=False,
    )

    if not response:
        raise Exception(f"Non-success status code: {response.status_code}")

    # STEP 4 (Optional) Monitor the operation to see when the upgrade is finished.
    # Location of created operation is in the Location header
    operationURI = response.headers["Location"]

    while True:
        response = requests.get(server + operationURI, verify=False)
        if not response:
            raise Exception(f"Non-success status code: {response.status_code}")

        operationStatus = response.json()["data"]["operation"]["state"]
        if operationStatus != "Requested":
            break
        # sleep for a while before retrying
        time.sleep(10)

    print("Status: " + operationStatus)

else:
    print("No upgrade possible")
```

2.3. nShield HSM management

2.3.1. Scenario

As an nShield HSM manager, I want to select a number of HSMs and output information about them in a csv file.

2.3.2. Steps

1. Select HSMs (in a pool, by label) and get data about them: `GET /mgmt/v1/hsms`

2.3.3. Python code example

```
import csv
import sys
```

```
import requests

server = "https://192.0.2.1"
# STEP 1 Get a list of HSMs which ...
params = {}  # would get all HSMs
params["pool"] = "44820fb6-bace-4dcd-ad97-7b6e0a5e4138"  # HSMs in this pool
params["labels"] = ["test1", "test2"]  # and HSMs with labels test1 AND test2
params["limit"] = (
    "5"  # Optionally choose the amount to fetch (otherwise get the default)
)

# Write output to a csv file
with open(sys.argv[1], "w") as file:
    wr = csv.writer(file)
    wr.writerow(["id", "type", "esn", "vcm", "version"])

    cursor = None
    # while there are more HSM records
    while cursor is None or cursor != "":
        # Read a page of records
        response = requests.get(server + "/mgmt/v1/hsms", params=params, verify=False)

        if not response:
            raise Exception(f"Non-success status code: {response.status_code}")

        # use the cursor to get the next page of records
        cursor = response.json()["meta"]["page"]["next"]
        params["cursor"] = cursor

        hsms = response.json()["data"]["hsms"]
        for hsm in hsms:
            # write out some information to the file
            wr.writerow(
                [
                    hsm["id"],
                    hsm["hsmType"],
                    hsm["esn"],
                    hsm["vcm"] if "vcm" in hsm else "",
                    hsm["versionInfo"]["versionNumber"],
                ]
            )
```

2.4. nShield Security World management

2.4.1. Scenario

As an nShield Security World manager, I want to create a new softcard.

2.4.2. Steps

1. Create a new softcard: `POST /mgmt/v1/worlds/<worldid>/softcards`
2. Get the authorisations needed: `GET /mgmt/v1/worlds/<worldid>/operations/<operationid>`
3. Authorise the creation with a new passphrase: `POST /mgmt/v1/worlds/<worldid>/opera`

tions/<operationid>/actions

4. (Optional) Check the operation is now authorised: GET /mgmt/v1/worlds/<worldid>/operations/<operationid>

2.4.3. Python code example

```python
import requests

server = "https://192.0.2.1"
# STEP 1 Create a new softcard
response = requests.post(
    server + "/mgmt/v1/worlds/be94c849-3a73-4ace-b493-a30b5b2243da/softcards",
    json={"name": "NewSoftcardName", "pprecovery": False},
    verify=False,
)
if not response:
    raise Exception(f"Non-success status code: {response.status_code}")

# Location of created operation is in the Location header
operationURI = response.headers["Location"]

# STEP 2 Get the authorisations needed
response = requests.get(server + operationURI, verify=False)
if not response:
    raise Exception(f"Non-success status code: {response.status_code}")

authorisations = response.json()["data"]["operation"]["authorizations"]

# STEP 3 Authorise the creation with a new passphrase
for auth in authorisations:
    # If it is a passphrase authorisation in requested state (there will actually only be one of these)
    if auth["authorizationType"] == "Passphrase" and auth["state"] == "Requested":
        response = requests.post(
            server + operationURI + "/actions",
            json={
                "actionType": "authorization",
                "resourceURI": operationURI + "/authorizations/" + auth["id"],
                "passphrase": "NewPassphrase",
            },
            verify=False,
        )
        if not response:
            raise Exception(f"Non-success status code: {response.status_code}")

# STEP 4 (Optional) Check the operation is now authorised
response = requests.get(server + operationURI, verify=False)
if not response:
    raise Exception(f"Non-success status code: {response.status_code}")

print(response.json()["data"]["operation"]["state"] == "Authorized")
```

3. Agent Management API Documentation

```
<redoc id='redoc-container'></redoc>
<script src="https://cdn.jsdelivr.net/npm/redoc@2.0.0-rc.64/bundles/redoc.standalone.js"></script>
<script>
    Redoc.init('./_attachments/agent-mgmt.yaml',
    {scrollYOffset: '.toolbar'},
    document.getElementById('redoc-container'))
</script>
```

4. CodeSafe Management API Documentation

```
<redoc id='redoc-container'></redoc>
<script src="https://cdn.jsdelivr.net/npm/redoc@2.0.0-rc.64/bundles/redoc.standalone.js"></script>
<script>
    Redoc.init('./_attachments/codesafe-mgmt.yaml',
    {scrollYOffset: '.toolbar'},
    document.getElementById('redoc-container'))
</script>
```

5. HSM Management API Documentation

```
<redoc id='redoc-container'></redoc>
<script src="https://cdn.jsdelivr.net/npm/redoc@2.0.0-rc.64/bundles/redoc.standalone.js"></script>
<script>
    Redoc.init('./_attachments/hsm-mgmt.yaml',
    {scrollYOffset: '.toolbar'},
    document.getElementById('redoc-container'))
</script>
```

6. Licence Management API Documentation

```
<redoc id='redoc-container'></redoc>
<script src="https://cdn.jsdelivr.net/npm/redoc@2.0.0-rc.64/bundles/redoc.standalone.js"></script>
<script>
    Redoc.init('./_attachments/licence-mgmt.yaml',
    {scrollYOffset: '.toolbar'},
    document.getElementById('redoc-container'))
</script>
```

7. Monitoring Management API Documentation

```
<redoc id='redoc-container'></redoc>
<script src="https://cdn.jsdelivr.net/npm/redoc@2.0.0-rc.64/bundles/redoc.standalone.js"></script>
<script>
    Redoc.init('./_attachments/monitoring-mgmt.yaml',
    {scrollYOffset: '.toolbar'},
    document.getElementById('redoc-container'))
</script>
```

8. Security World Management API Documentation

```
<redoc id='redoc-container'></redoc>
<script src="https://cdn.jsdelivr.net/npm/redoc@2.0.0-rc.64/bundles/redoc.standalone.js"></script>
<script>
    Redoc.init('./_attachments/sw-mgmt.yaml',
    {scrollYOffset: '.toolbar'},
    document.getElementById('redoc-container'))
</script>
```