# VMware Object Storage Interoperability Service - Development Guide

vmware®

## Table of contents

## Introduction

VMware Cloud Director Object Storage Extension (OSE) is a common Object Storage Service that allows service providers to enable Object Storage Service for Cloud Director tenants. By 2.0 release, OSE has built powerful integrations with three platforms: Cloudian HyperStore, DELL ECS, and Amazon native S3.

Nevertheless, OSE 2.0 also opens the extensibility for any 3rd party S3-compliant object storage platforms to integrate with Cloud Director, for instance, open-source Ceph. We call the extension point Object Storage Interoperability Service (OSIS). OSIS defines a set of management API specifications for OSE to communicate with 3rd party object storage platforms to exchange tenant and user information. ISV takes the ownership to implement the OSIS adapter for the vendor object storage platform.

By following the unified onboarding workflow, Cloud Director tenants can consume the vendor Object Storage in a unified manner, regardless of the type of the object storage platform. The user experience for Object Storage is coherent and adaptive in the Cloud Director portals. The below diagram shows the high-level architecture of OSE with all rely-on parties into consideration.

It is worth mentioning that switching Object Storage platforms after the deployment is super easy. Service providers only need to run a few commands to switch the underlying Object Storage platform from one to the other.

For the platforms integrated with OSE via OSIS, the data channel is between OSE and the platform. Still, the control channel is between OSE and the OSIS adapter.

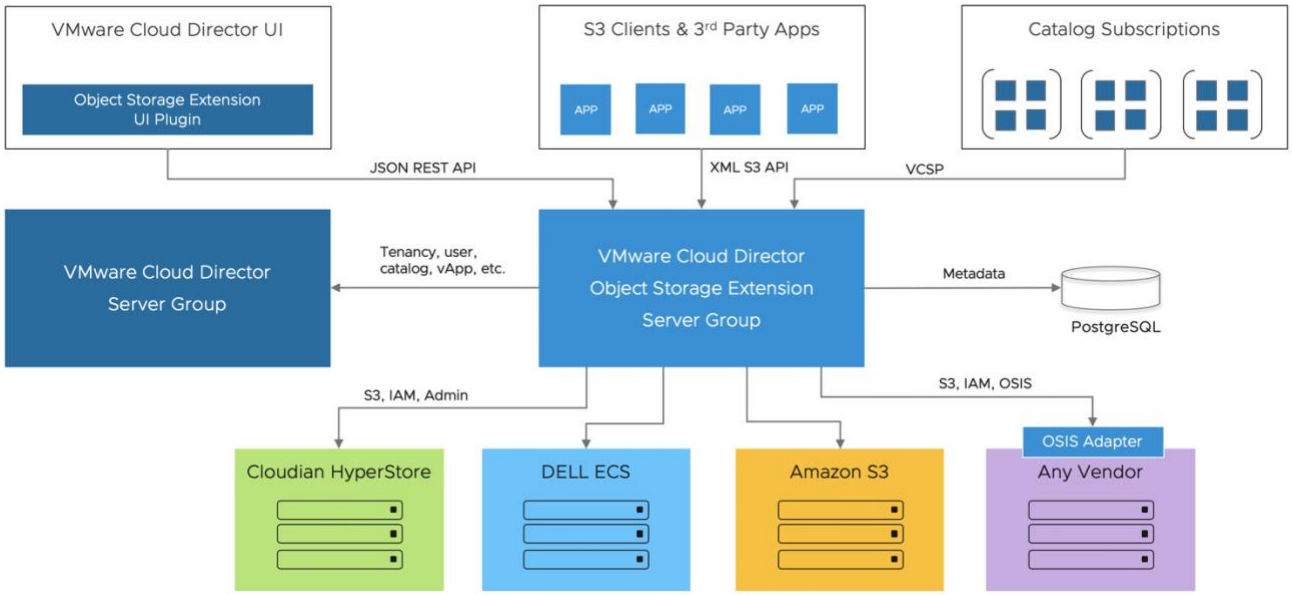


*Figure 1 OSE and OSIS High-level Architecture*

## Audience

This document is intended for VMware Cloud Provider architects and technical leads responsible for planning and executing the deployment and upgrades of a VMware-based cloud environment.

## System Requirement

The OSIS Adapter needs to be developed as an API service available to VMware Cloud Director Object Storage Extension (OSE) server. You can implement the OSIS Adapter in any programming language and deploy to any server environment. The only requirement for the OSIS Adapter is network connectivity. It should be deployed in an intranet that the OSE server can connect to it, and OSIS can connect to the storage platform.

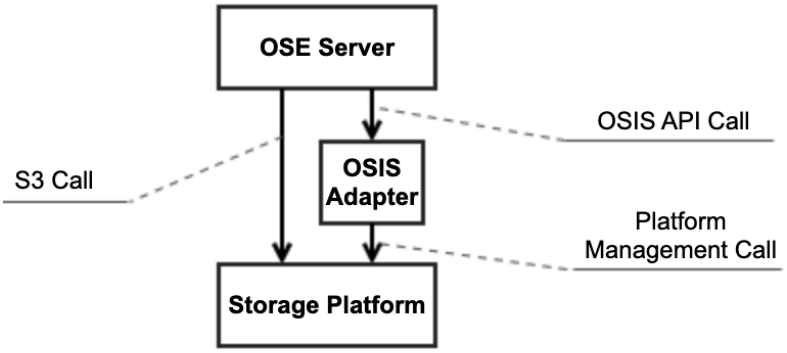


*Figure 2: OSIS Deployment View*

### Best Practice

If you install the OSIS Adapter in a standalone machine, the hardware configuration below is recommended.

- vCPU: 4 Core
- Memory: 8GB
- Disk: 100GB
- OS: Linux (CentOS 7+ is recommend)
- Database: PostgreSQL 10+

As a best practice, the OSIS Adapter can be installed on the same node as the OSE server and share same PostgreSQL database server that OSE uses. In this way, the additional maintenance effort can be reduced.

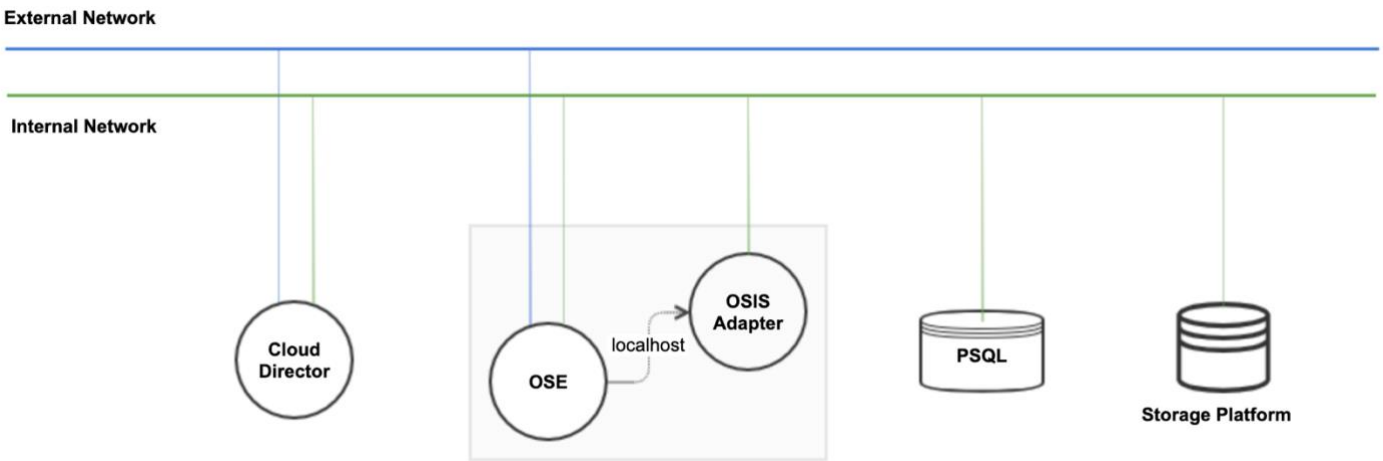


*Figure 3: OSIS Network*

## OSIS Workflow

The following diagrams depicts the S3 storage data flow between the OSIS adapter and OSE, and the OSIS and the storage platform.

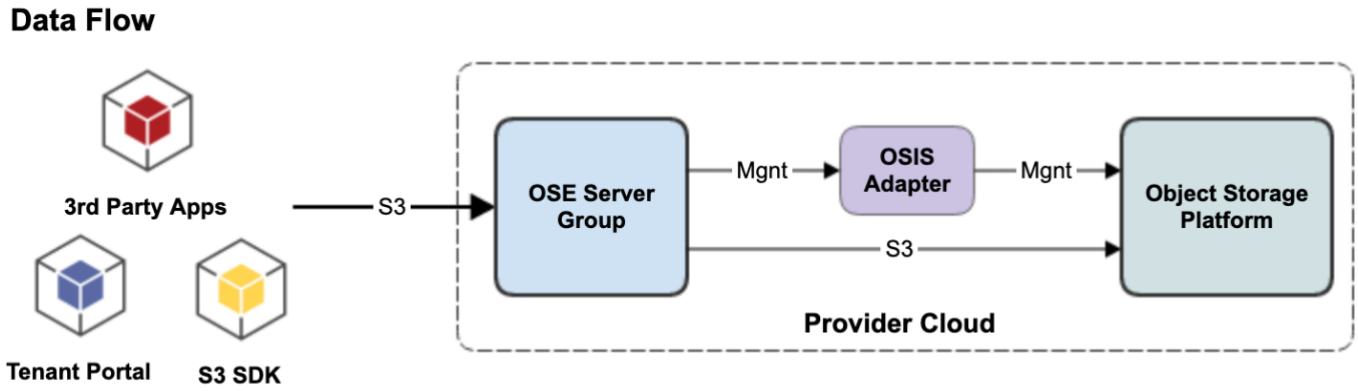


*Figure 4: OSIS Data Flow*

The general workflow to make a vendor Object Storage platform available for Cloud Director is as following:

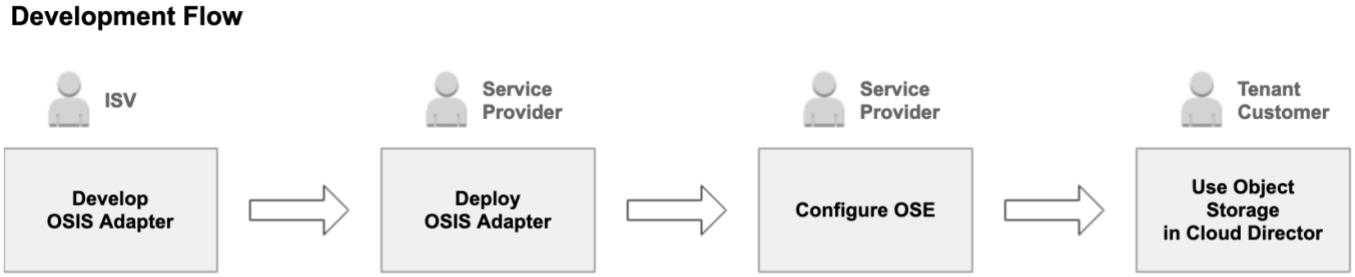


*Figure 5: OSIS Development Flow*

## Develop an OSIS Adapter

Implement the following mandatory and optional OSIS APIs when you develop your own OSIS adapter.

Refer to the following comprehensive list of OSIS APIs to start managing your S3 platform, tenants, users, buckets, and S3 authentication: *https://code.vmware.com/apis/1034/oject-storage-interoperability-service#/api*.

Here is a glimpse of the mandatory and optional OSIS APIs that need to be implemented in your OSIS adapter. There is a *reference CEPH implementation project* that you can refer to while developing and deploying your own OSIS adapter.

You can also use the following *OSIS stub* as a skeleton for your OSIS adapter.

### Mandatory OSIS APIs

The following APIs are mandatory for using a vendor S3 storage platform with Cloud Director:

| HTTP Method | Request URI | Description |
|---|---|---|
| HEAD | /api/v1/tenants/{tenantId} | Checks whether the tenant exists. |
| PATCH | /api/v1/tenants/{tenantId} | Updates Cloud Director tenant ID of the S3 storage platform tenant. |
| GET | /api/v1/tenants | Gets the tenant. |
| POST | /api/v1/tenants | Lists the tenants of the S3 storage platform. |
| GET | /api/v1/tenants/query | Queries the tenants of the S3 storage platform. |
| GET | /api/v1/users/query | Queries users of the platform tenant. |
| POST | /api/v1/tenants/{tenantId}/users | Creates a user in the platform tenant. |
| GET | /api/v1/tenants/{tenantId}/users | Lists users of the platform tenant. |
| GET | /api/v1/tenants/{tenantId}/users/{userId} | Gets the user with user ID of the tenant. |
| PATCH | /api/v1/tenants/{tenantId}/users/{userId} | Sets enable or disable status in the tenant. |
| DELETE | /api/v1/tenants/{tenantId}/users/{userId} | Deletes the user in the platform tenant. |
| GET | /api/v1/users/{canonicalUserId} | Gets the user with user canonical ID. |
| GET | /api/v1/s3credentials/query | Queries S3 credentials of the platform user. |
| GET | /api/v1/tenants/{tenantId}/users/{userId}/s3credentials | Lists S3 credentials of the platform user. |

| | | |
|---|---|---|
| POST | /api/v1/tenants/{tenantId}/users/{userId}/s3credentials | Creates S3 credential for the platform user. |
| GET | /api/v1/s3credentials/{accessKey} | Gets S3 credential of the platform user. |
| GET | /api/v1/s3capabilities | Gets S3 capabilities of the platform. |
| GET | /api/info | Gets the REST services information. |

## Optional OSIS APIs

Those optional API endpoints are nice to have. The corresponding functions will be hidden in the VMware Cloud Director tenant portals if you do not implement them.

| HTTP Method | Request URI | Description |
|---|---|---|
| GET | /api/v1/tenants/{tenantId} | Get a tenant in the platform |
| DELETE | /api/v1/tenants/{tenantId} | Delete a tenant in the platform |
| HEAD | /api/v1/tenants/{tenantId}/users/{userId} | Check whether the user exists |
| PATCH | /api/v1/s3credentials/{accessKey} | Enable or disable S3 credential for the platform user |
| DELETE | /api/v1/s3credentials/{accessKey} | Delete the S3 credential of the platform user |
| GET | /api/v1/usage | Get the usage of the platform tenant or user |
| GET | /api/v1/bucket-list | Get the bucket list of the platform tenant |
| GET | /api/v1/bucket-logging-id | Get the bucket logging id of the platform |
| GET | /api/v1/anonymous-user | Get the anonymous user id and name of the platform. |
| GET | /api/v1/console | Get the console URI of the platform or platform tenant |

## OSIS APIs Explained

The OSIS APIS are divided into categories, including:

- Authentication
- Information
- Tenant
- User
- S3 credential
- S3 capability
- Miscellaneous - Bucket logging ID, Anonymous user, Usage, Console URL, etc.

**Authentication**

Two authentication schemas are defined for OSIS: basic and bearer API token. Basic authentication is simple and straightforward.

As the client of OSIS - OSE is configured, a refresh token with CLI sends a request with a refresh token to exchange JWT access token during starting up. Then use the access token for subsequent OSIS requests.

Once the access token expires, OSE uses a refresh token to renew the access token. The refresh token should be set with a long expiration time, like half a year. If the refresh token expires, it should be reconfigured in OSE by CLI.

Bear API token has one new API to achieve this: **`/api/v1/auth/token`**, which renews the access token with a refresh token.

OSIS accepts the bearer token in the HTTP header from OSE like, **`Authorization: Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJNUEcwUlZDRENERFRTUkpKT05KSyIsInNjb3...`**

**Information**

Vendors or communities could have a different implementation and configuration of the OSIS specification. OSE can get the customization metadata of OSIS via the information **`API/api/info`**, which doesn't require authentication.

Here is a sample of the information representation. The explanation is provided inline for some fields.

```
{
  "platform_name": "ceph",
  "platform_version": "15.2.3",
  "api_version": "1.0.0", // The e API version which the implementation complies with.
  "status": "NORMAL", // The status of the OSIS server instance.
  "not_implemented": [ // Any API defined in the specification, which is not implemented, will be put in this array. The API code can be found in the OSIS API specification.
    "getAnonymousUser",
    "getBucketLoggingId",
    "updateCredentialStatus",
    "getBucketList"
  ],
  "logo_uri": "https://10.139.113.213:28443/ceph.png", // This is the logo URI for the storage platform.
```

```
  "auth_modes": [ // This is the current working authentication mode.
    "BASIC"
  ],
  "services": {
    "s3": "http://ceph.osis.ose.vmware.com:31383", // Endpoints of the storage platform other than the OSIS server.
    "iam": "https://10.139.113.213:28443/admin"
  },
  "regions": [
    "default"
  ],
  "storage_classes": [
    "default"
  ]
}
```

OSE accesses this information resource and takes different actions based on the response. e.g., different API code in **`not_implemented`** will make the OSE portal show the UI components adaptively.

API code in **`information.not_implemented`** is the Operation ID in the OSIS specification.

**Tenant**

OSIS APIs for storage tenants cover CRUD and query operations. OSE calls these APIs for any tenant functions. Tenant APIs handle two kinds of ID: **`cd_tenant_id`** and **`tenant_id.`**

**`cd_tenant_id`** is the ID of a Cloud Director tenant, and **`tenant_id`** is the storage platform tenant ID, which is chosen by the OSIS implementation.

The tenant query API accepts query criteria which is flexible and susceptible for extension in the future.

**User**

OSIS user APIs are like tenant APIs, which allow OSE to do many operations against platform users. They also have similar ID properties, including **`cd_tenant_id`**, **`tenant_id`**, **`cd_user_id`**, and **`user_id`** for Cloud Director and the storage platform, respectively.

Generally speaking, **`canonical_user_id`** is a globally unique ID for a user of the storage platform; however, **`user_id`** is specific for a certain tenant.

**S3 Credentials**

OSIS APIs for S3 credentials support CRUD and query operations. One S3 credential is always attached to a user; therefore, the S3 credential model has the IDs of a tenant and user.

For most storage platforms, the access key is the unique ID for the S3 credential. But this is not true for some platforms. Thus, **`GET/PATCH/DELETE`** operations for **`/api/v1/s3credentials/{accessKey}`** have optional parameters **`tenant_id`** and **`user_id`** to identify one S3 credential.

**S3 Capability**

Object storage platforms have different support and compatibility for S3 APIs, which impacts the OSE and its consumers. When a platform is integrated with OSE via the OSIS service, it is required for OSE to know the compatibilities of S3 APIs so that OSE can customize the provider portal and tenant portal. This is what S3 capability API is for.

**Miscellaneous**

The APIs like usage, console URI, bucket logging ID, etc., provide the essential data for several OSE administrative functions.

For example, if the API of a bucket logging ID is implemented, the Bucket Properties tab in the OSE tenant portal will have a card for the bucket logging configuration. If not, the card will be hidden. In fact, these APIs are not mandatory.

## Tenant and User Mapping

OSIS server accepts Cloud Director tenant/user model from OSE and returns platform tenant/user model to OSE. The mapping from Cloud Director to the storage platform is designed and maintained by the OSIS implementation.

Different platforms have different schema and interface for tenant/user. Here is an example of how to design and maintain the tenant mapping.

1. List a request payload to create a platform tenant and an OSIS tenant model

| Payload of a tenant creation request | OSIS tenant model of any S3 platform |
|---|---|
| **{**<br>**"name": "<CD_TENANT_NAME>",**<br>**"cd_tenant_ids": [**<br>**"<CD_TENANT_ID>"**<br>**]**<br>**}** | **tenant_id**<br>**name**<br>**active**<br>**cd_tenant_ids** |

2. Decide the rule to generate the platform **`tenant_id`** according to the platform capability and fill in the values for the OSIS tenant model.

   - The platform accepts the UUID as a tenant ID, e.g. Cloudian accepts UUID as a tenant ID

| Payload of a tenant creation request | OSIS tenant model for Cloudian |
|---|---|
| **{**<br>**"name": "<CD_TENANT_NAME>",**<br>**"cd_tenant_ids": [**<br>**"<CD_TENANT_ID>"**<br>**]**<br>**}** | **tenant_id=<CD_TENANT_ID>**<br>**name=<CD_TENANT_NAME>**<br>**active=true**<br>**cd_tenant_ids=[<CD_TENANT_ID>]** |

   - Platform does NOT accept UUID as a tenant ID e.g., CEPH doesn't support UUID as a tenant ID.
     Use <CD_TENANT_NAME>__trim(<CD_TENANT_ID>) as a tenant ID.

| Payload of a tenant creation request | OSIS tenant model for CEPH |
|---|---|
| **{**<br>**"name": "<CD_TENANT_NAME>",**<br>**"cd_tenant_ids": [**<br>**"<CD_TENANT_ID>"**<br>**]**<br>**}** | **tenant_id=<CD_TENANT_NAME>__trim(<CD_TENANT_ID>)**<br>**name=<CD_TENANT_NAME>**<br>**active=true**<br>**cd_tenant_ids=[<CD_TENANT_ID>]** |

   - Platform has no concept of a tenant e.g., for Amazon, an AWS account can be used as a tenant.

| Payload of a tenant creation request | OSIS tenant model for Amazon |
|---|---|

```
{
 "name": "<CD_TENANT_NAME>",
 "cd_tenant_ids": [
  "<CD_TENANT_ID>"
 ]
}
```

```
tenant_id=<AWS_ACCOUNT_ID>
name=<CD_TENANT_NAME>
active=true
cd_tenant_ids=[<CD_TENANT_ID>]
```

The cases are not limited to those listed above. Any mapping rule that complies with the guideline should work well with the OSIS implementation: OSIS **tenant_id** can identify a unique "tenant" in the platform.

On the other hand, the OSIS server can query platform tenants when receiving **`<CD_TENANT_ID>.`** This means the OSIS server should maintain the mapping between CD tenants and platform tenants to support query by **`<CD_TENANT_ID>.`**
The OSIS implementation can persist the mapping in the storage platform or a dedicated database.

- For example, in the OSIS CEPH reference project, the mapping is stored in property **`display_name`** of the CEPH model (*you can find details in the documentation of the reference project).*
- If a dedicated database is the choice, the OSIS implementation should keep the data synchronized between the database and platform.

In the end-to-end flow OSE → OSIS → Platform, the previous steps focus on OSE→ OSIS. The OSIS implementation need to handle and the conversion between the OSIS model and the platform model.

The mapping of a user/S3Credential is like a tenant.

## Choose the Platform Administrative and S3 Capabilities

In the OSIS specification introduction, we already know that there are two kinds of capabilities for OSIS to provide: administrative capability and S3 capability. Administrative capability is straightforward - specifying which optional APIs are not implemented.

Let's talk a bit more about the S3 capability. Below is a sample representation of the S3 capability resource.

The **`s3_`** prefixed keys are converted from the action names in *Amazon S3 API Reference*, e.g., **`CreateBucket`** → **`s3_create_bucket`**. Inline is explained what the representation of the S3 capability stands for.

```
{
 "CEPH": {
  "exclusions": { // Defines which S3 APIs are excluded.
   "s3_set_bucket_encryption": {
     "by_payload": [ // SetBucketEncryption is supported except the option kms in the payload.
      "kms"
    ]
   },
   "s3_set_object_metadata": {
    "by_headers": [ // SetObjectMetadata is supported except two HTTP headers x-amz-request-payer and x-amz-expire.
      "x-amz-request-payer",
      "x-amz-expire"
    ]
   },
   "s3_get_bucket_cors": {} // GetBucketCors is totally not supported.
  }
 }
}
```

The S3 capability resource specifies which S3 APIs are not supported. There are four kinds of exclusions:

1. totally excluded
2. excluded by_params
3. excluded by_headers
4. excluded by_payload

From the implementation view, developers should elaborate and figure out which S3 APIs are supported by the storage platform:

- Study the storage platform S3 API documentation and compare it with Amazon S3
- Hand on storage platform S3 API

Then describe and persist the capabilities in a specific schema, like in JSON or YAML (*see the sample in the CEPH reference project*). In runtime, convert the data in the schema to a capability resource model and marshal out.

## Construct the OSIS Adapter Implementation

Once the stubs and design are ready, it is time to construct the implementation.

This topic depends on the programming language and the framework. We will give some general points for preparing the components:

- Client to exchange data with the storage platform. *See an example here.*
- A database utility if a database is adopted.
- Converter to handle the OSIS and platform models. *See* an example *here*.
- Service to cooperate the components above, and the stubs should call the service to serve the OSE requests. *See an example here.*

## Best Practices in the OSIS Adapter Development

The following practices are recommended for developing your OSIS adapter:

1. Ensure all required APIs are implemented.
2. Unimplemented optional APIs should return `501 Not Implemented` HTTP code.
3. Do not partially implement the optional OSIS APIs (e.g., if an API can accept two URL parameters in the specification, support both parameters, or none).
4. Consider the string length and the special-character acceptance of the storage platform for model properties like ID and name.
5. Do not introduce a database If it is not mandatory.

## Secure the OSIS Authentication

As we know, OSIS specification includes two authentication modes: basic and API token. Several recommended rules are listed below for the implementation of the OSIS authentication module.

- API token is preferred over basic authentication.
- When basic or API token authentication is adopted, the identity used to authenticate entities must be unique. This means the unique ID remains the same even if the username changes.
- It must implement countermeasures to defend against brute force attacks.
- User passwords must be stored in the form of a salted hash.
- Salts used for one-way hashing must be generated using cryptographically strong random data and must be at least 128-bits long.
- The one-way hash must be with one of the approved algorithms: PBKDF2 and Scrypt.

## Secure the Channel between OSE and OSIS

No matter how OSE and OSIS communicate - over Intranet or Internet, HTTPS is preferred in the OSIS implementation.

The platform administrator can use the OSE CLI *ose osis admin set* to configure the HTTPS endpoint of OSIS, like *https://osis.ceph.ose.vmware.com:8443.*

If HTTP is adopted in the OSIS implementation, the OSIS server should be deployed on the same machine with an OSE server, or in the same secure private network with NAT (only NAT rule for OSE so that OSIS is isolated in the private network).

When they are on the same machine, HTTP port should be made loopback only, other than bind on the external network interface.

Besides user authentication, the deployment of OSIS can limit the access from allowed sources: OSE and the storage platform by configuring firewall rules.

Only necessary communication ports need to be open for the OSIS host, like the HTTPS port for OSIS RESTful services.

In the deployment, it must be taken appropriate measures to protect OSE against Denial of Service (DoS) attacks.

### Security Recommendations

The implementation must always validate that the input data meets the expectations defined by the specifications to avoid various attacks.

OSE should use storage platform account to access platform admin endpoint. In the platform, the account should not be configured with additional unnecessary privileges.

Log the following types of security-relevant events for the purpose of a future audit:

- All authentication attempts; successful or not. However, do not log the passwords.
- Significant operational actions such as application startup and shutdown, application failures, and major application configuration changes.
- Client requests and server responses. Care must be taken to not log sensitive information such as passwords.
- All events related to the serviceability.
- Any other operations performed by authenticated entities that could be relevant in a security audit.
- Any other operations performed by automated process (with or without authenticated users) that could be relevant in a security audit. Examples are automated encryption key rolls.

## Configure OSIS

Now assumed you have developed an OSIS adapter for your vendor Object Storage platform named X-S3-IO, and service providers want to switch OSE from Amazon native S3 to vendor Object Storage platform. OSE provides convenient CLIs to complete the switch. Here is a CLI snippet.

1. Configure the OSIS API connection

   ```
   ose osis admin set --name X-S3-IO --url <https://osis-adapter-api-server> --user <username> --secret <password>
   ```

2. To set a refresh token, run the following command:

   ```
   ose args set --k oss.platform.X-S3-IO.admin.refresh-token -v <refresh-token>
   ```

3. Configure the X-S3-IO S3 endpoint

   ```
   ose osis s3 set --name X-S3-IO --url <https://x-s3-io-s3-endpoint>
   ```

4. Switch to platform X-S3-IO

   ```
   ose platforms enable osis --name X-S3-IO
   ```

5. Restart the OSE service

   ```
   ose service restart
   ```

After restarting OSE, you would be able to enable tenants to use the new Object Storage platform.

## Verifying the OSIS Implementation

### OSIS Verifier

Before integrating with Object Storage Extension (OSE) and the object storage platform, it is recommended to run the *OSIS Verifier tool* to verify the OSIS implementation, which would find issues with less effort.

**Requirements**

Python 3.4+

**Installation & Usage**

`pip install`

If the python package is hosted on a repository, you can install it directly using:

`pip install -r requirements.txt`

(You may need to run pip with root permission: `sudo pip install -r requirements.txt`)

**Getting Started**

Execute the command `python osis_verify.py` to verify OSIS required APIs:

`python osis_verifier.py`

```
 ___ ____ ___ ____  __   __    _ __ _
/ _ \/ ___|_ _/ ___| \ \ / /__ _ __(_)/ _(_) ___ _ __
| | | \___ \| |\___ \  \ V / _ \'__| | |_| |/ _ \'__|
| |_| |___) | | ___) |  \ V /  __/ | | |  _| |  __/ |
 \___/|____/___|____/    \_/ \___|_| |_|_| |_|\___|_|
```

Input OSIS Endpoint: [example - https://localhost:8443]https://localhost:9443

Input OSIS Username: MPG0RVCDCDDTSRJJONJK

Input OSIS Password :c5XMURa3Eu3KGPvHDmwTw8GQSOzDujfmNA94tlPh

Input OSIS S3 Endpoint: [example - http://ceph.osis.ose.vmware.com:31383]http://ceph.osis.ose.vmware.com:31383

**********

* The verification items *

**********

0. Quit the verifier

1. Verify S3 Capabilities API

2. Verify S3 Credential APIs

3. Verify Info API

4. Verify Tenant APIs

5. Verify User APIs

6. Verify tenant/user onboard and S3 request

Choose what of the OSIS implementation you want to verify. Once there is any error during the verification, check the log file `osis-verifier.log` for details.

## Test OSIS with OSE and the Storage Platform

Unit tests and integration tests should be done before testing with OSE because it will take a significant effort to troubleshoot any issue that may appear in Cloud Director, OSE, OSIS and the storage platform.

We can test the function on the OSE portal. Mainly follow this order

1. Log in to the VMware Cloud Director with the service provider credentials.
2. Launch the OSE provider administrator portal.
3. In the tenant list, check the general information of the tenants.
4. Pick and enable one tenant.
5. Check the information of the onboarded tenant, especially the storage tenant ID.
6. Log out.
7. Log in to the VMware Cloud Director with a tenant administrator/user credentials.
8. Launch the OSE tenant portal.
9. Try an S3 operation like creating a bucket.
10. Check the S3 credentials of the user.
11. Go around the portal.

If steps 3 and 7 work well, Cloud Director users can consume S3 service of the storage platform successfully. The remaining work is to adjust and polish implementation to fulfill your requirements.

## OSIS CEPH Reference Implementation

To help you design and deploy your OSIS adapter, we have prepared a reference OSIS implementation, which integrates CEPH (an open-source distributed storage) with OSE.

The reference project can be accessed *here*. It includes information on how to design, build, and configure your OSIS adapter for the CEPH storage. Have a look at the documentation of the project to deploy your OSIS adapter and start using CEPH with OSE.

## Glossary

| | |
|---|---|
| OSIS | VMware Object Storage Interoperability Service |
| OSE | VMware Vloud Director Object Storage Extension |
| API | Application Programming Interface |
| ISV | Independent Software Vendor |
| Tenant | A tenant is an organizational unit in VMware Cloud Director. A tenant can represent a business unit in an enterprise or a company that subscribes to cloud services from a service provider. |
| Cloud Director | VMware Cloud Director |
| CEPH | Open-source software storage platform that implements object storage. |

**VMware, Inc.** 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 vmware.com  Item No: vmw-wp-tech-temp-a4-word-101-proof 6/20