

**Qualcomm Car-to-Cloud Platform**

**USAGE OF STORAGE SERVICE INTERFACE**

**Version No.3.0**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Prepared By / Last Updated By** | **Reviewed By** | **Approved By** |
| **Name** | Soumya Dasgupta | Sudheer | Sudheer |
| **Role** | PAT | SM | SM |
| **Signature** |  |  |  |
| **Date** | 27/05/2021 | 11/06/2021 | 11/06/2021 |

Table of Contents

[1. STORAGE SERVICE INTERFACE 3](#_Toc74561569)

[2. INTERFACE APPLICATION 3](#_Toc74561570)

[2.1. c2c\_base\_storage\_service\_intf 4](#_Toc74561571)

[2.2. c2c\_base\_storage\_service\_impl 7](#_Toc74561572)

# STORAGE SERVICE INTERFACE

An interface contains variables and signatures of function. These signatures can be implemented by other classes that implements the interface. This functionality improves the flexibility and stability of programs and applications.

A significant way to utilize this functionality is by separating the interface and its implementation through converting the interface as jar and adding it to its implementation as a dependency. This separation of interface enables the possibility of having multiple implementations for the same interface.

**FEATURES**

* An interface project that outlines the signatures of basic functionalities
* An implementation project that implements the signatures in the interface project
* A base class whose main purpose is to call the implemented methods and determine the flow of the application
* An ability to change the implementation by changing the jar file in the application without changing or affecting the flow of the base program.

The idea is to use this concept to provide a common Cloud Agnostic Method.

# INTERFACE APPLICATION

The storage service application is meant to upload, delete, copy, move, download, list and do multiple upload of objects with respect to a storage via cloud services. Different cloud services provide this functionality, for example AWS S3 or Azure Blob. For this application AWS S3 has been chosen.

The project will be divided into two projects –

* **c2c\_base\_storage\_service\_intf** which has the interfaces containing signatures of functions related to upload, delete, copy, move, download, list, multiple upload and multiple delete. This is converted as a JAR file.
* **c2c\_base\_storage\_service\_impl** which contains the implementation of the interfaces c2c\_base\_storage\_service\_intf. The implementations will be responsible for upload, delete, copy, move, download, multiple delete, list and multiple upload of objects using AWS S3. It uses the interface’s JAR file as a dependency.

## c2c\_base\_storage\_service\_intf

*c2c\_base\_storage\_service\_intf* is the storage service interface project that contains the interfaces needed for upload, delete, copy, move, download, list, multiple upload objects. It can be used to implement any cloud service. It contains the following classes:

* ***IC2CStorageServiceClient*** an interface which contains:
* *public void upload(String bucketName, String filePath, InputStream inputStream) throws StorageApplicationException, StorageOperationException –*

Method for uploading an object into the bucket. It accepts the bucket name, file path of the object and an InputStream object that contains the data that needs to be uploaded as parameters.

It throws two custom exceptions – *StorageApplicationException* and *StorageOperationException*.

* *public void delete(String bucketName, String filePath) throws StorageApplicationException, StorageOperationException –*

Method for deleting an object from the bucket. It accepts the bucket name, file path of the object as parameters.

It throws two custom exceptions - *StorageApplicationException*, and *StorageOperationException*.

* *public void copy(String sourceBucketName, String filePathInSourceBucket, String destinationBucketName, String filePathInDestinationBucket) throws StorageApplicationException, StorageOperationException –*

Method for performing the Copy functionality of a particular object present in a storage service. It accepts the source bucket name, file path in source bucket, destination bucket name and file path in destination bucket as parameters.

It can throw two custom exceptions which are *StorageApplicationException*, and *StorageOperationException*.

* *public void move(String sourceBucketName, String filePathInSourceBucket, String destinationBucketName, String filePathInDestinationBucket) throws StorageApplicationException, StorageOperationException–*

Method for performing the move functionality of a particular object present in a storage service. It accepts the source bucket name, file path in source bucket, destination bucket name and file path in destination bucket as parameters.

It throws two custom exceptions - *StorageApplicationException*, and *StorageOperationException*.

* *public BufferedInputStream download(String bucketName, String filePath) throws StorageApplicationException, StorageOperationException –*

Method to download an object present in the bucket. It accepts the bucket name and file path as parameter.

It must return BufferedInputStream, an input stream data that is to be downloaded.

It throws *StorageApplicationException*, *StorageOperationException*.

* *public void delete(String bucketName*, *List<String> filePaths) throws StorageApplicationException, StorageOperationException –*

Method to delete multiple objects from the bucket. It accepts the bucket name and list of file paths as parameters.

It throws two custom exceptions - *StorageApplicationException*, and *StorageOperationException*.

* *public List<Object> list(String bucketName) throws StorageApplicationException, StorageOperationException–*

Method to list all the objects present in the bucket. It accepts the bucket name as parameter.

It should return a list of keys/path to all the objects in storage.

It throws two custom exceptions - *StorageApplicationException* and *StorageOperationException*.

* *public List<Object> list(String bucketName, String filePath, Boolean ListSubfolders) throws StorageApplicationException, StorageOperationException–*

Method to list all the objects present in a folder in the bucket. It accepts the bucket name, file path of the folder with delimiter and a Boolean variable ListSubfolders as parameters. If the Boolean is true, it will list all objects in the folder including the objects in the sub folders. If it is false, it will only list the objects in the folder excluding the sub folders.

It should return a list of keys/path to all the objects in the folder in the storage.

It throws two custom exceptions - *StorageApplicationException* and *StorageOperationException*.

* *public void upload(String bucketName, Map<String, InputStream> inputStreamMap)throws StorageApplicationException, StorageOperationException–*

Method to upload multiple objects present into bucket. It accepts the bucket name and *inputStreamMap*, which is the map that contains filePath as String Key and the stream as the InputStream.

It throws two custom exceptions - *StorageApplicationException* and S*torageOperationException*.

It contains a model class:

* ***C2CStorageConnectionConfig***– A model class that contains variables of possible parameters needed for configuration which are *region*, *accessKey*, *secretKey* and *projectId* with its getters and setters. *projectId* is a unique, user-assigned ID that can be used by Google APIs. *Region* is the region of the bucket in the cloud and *accessKey* and *secretKey* are the credentials needed for connecting to the cloud. It also contains a parameterized constructor. It also contains the Builder class with the same parameters. The builder class also provides a build method to create the connection config object.

It also contains custom exceptions:

* ***StorageApplicationException*** – A custom exception which extends *RuntimeException* that is thrown for application errors such as *IllegalArgumentException* and *SdkClientException*. An *IllegalArgumentException* is thrown when Access Key or Secret Key is NULL
* ***StorageOperationException***– A custom exception which extends *RuntimeException* that is thrown when we are not able to initialize or find the bucket using the details provided or if it is an invalid bucket.

## c2c\_base\_storage\_service\_impl

The implementation can be an AWS specific implementation for storage service, specifically S3, for c2c\_base\_storage\_service\_intf’s interfaces. It will contain the body for each interface function signatures.

Feature of the implementation would include -

* A constructor that is responsible for configuration with values provided by caller class of C2CStorageConnectionConfiguration type.
* A function to establish connection with AWS S3

The implementation will contain a jar file of c2c\_base\_storage\_service\_intf.