

Oracle Financial Services

File Upload and Download Utility User Guide



Release 25A
G33508-02
May 2025

ORACLE®

Copyright © 2022, 2025, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

1	Roles and Functions	
2	File Upload and Download Utility	
	Upload or Download File from Object Store Using Console	2-1
	Uploading/Downloading a File Using Utility	2-1
	Uploading/Downloading a File Using PAR URL	2-3
3	File Upload Automation	
	Step 1: Generate Access Token	3-1
	Step 2: Generate PAR URL	3-1
	Step 3: Upload file to Object Store	3-2
	Step 4: Scan the file to ensure Upload was Successful	3-2
	Automating the File Upload Process Using File Upload Utility	3-3
	Executing the File Upload Automation Script	3-3
4	Generating PAR URL for File Operations	
	Generating PAR URL for File Upload	4-1
	End Point Details	4-1
	Calling the API to Generate the URL	4-1
	Request JSON Parameters	4-2
	Response JSON Parameters	4-2
	Viewing List of Uploaded Files	4-3
	Generating PAR URL For File Download	4-4
	Calling the API to Generate PAR URL for File Download Using File Name	4-4
	Calling the API to Generate PAR URL for File Download Using File ID	4-5
5	Deleting A File	
	Endpoint Details	5-1
	Deleting a File Using the File ID	5-1
	Deleting a File Using Filename	5-2

Deleting Multiple Files Using Filenames

5-3

Delete files using a prefix value

5-4

1

Roles and Functions

The following table lists the role codes and function codes required to configure the File Upload/Download Utility.

Role Codes

Role Code	Function Code
FILE_READ	FILE_SUMMARY
FILE_UPLOAD	FILE_UPLOAD
FILE_DOWNLOAD	FILE_DOWNLOAD
FILE_ADV	FILE_UPLOAD
	FILE_DOWNLOAD
	FILE_DELETE
	FILE_SUMMARY

2

File Upload and Download Utility

The File Upload and Download Utility enables you to upload or download files to the Object Store. Complete the following steps to Upload or Download a file.

- [Upload or Download File from Object Store Using Console](#)
- [Uploading/Downloading a File Using Utility](#)
- [Uploading/Downloading a File Using PAR URL](#)

Upload or Download File from Object Store Using Console

1. From the left menu, click **Common Object Maintenance**.
2. Click **Data Management** in the left navigation pane.

The **File Upload and Download** Page is displayed. The Files that are uploaded to the Object Store are listed here. The following details are provided for each File.

- **File ID** - The unique file ID associated with the file. This is auto-generated during the upload.
- **Prefix** - The prefix is added to the file name.
- **File Name** - The name of the file that is uploaded. This is automatically updated after you select the file.
- **Stripe Name** - The Unique Identifier for storing a collection of files. Collection examples: Project, organization, tenant.
- **Uploaded Date** - The file upload date.
- **Download File** - Click to download a copy of the uploaded file.
- **Delete** - Click to delete the file.

Related Topics

- [Uploading/Downloading a File Using Utility](#)
Complete the following steps to Upload or Download a file using the Utility.
- [Uploading/Downloading a File Using PAR URL](#)
Complete the following steps to upload or download a file using the PAR URL.

Uploading/Downloading a File Using Utility

Complete the following steps to Upload or Download a file using the Utility.



1. Click **Upload** on the **File Upload/Download Summary** screen.

2. Click **Drag and Drop** to browse and select a file for upload from the local directory.

You can also browse to the local directory from the **File Explorer** and select the file and drop it here.

The file name is automatically updated in the **Selected File** field.

3. Enter the **Prefix** to be added to the file name.

The Prefix is added to the file name. In case, you have two files with the same file name, you can save them with different prefixes.

Example: **/abc/test.txt** and **/def/test.txt**. Both these files have the same file name but different Prefixes.

4. Click **Upload** to upload the selected file.

A confirmation message is displayed after successful upload and the file is listed in the Uploaded Files list.

Downloading file(s)

Complete the following steps to download a single file.

- a. Click the **Actions** menu corresponding to the file you want to download and select **Download**.
- b. Specify a name (optional) and file type (optional).
- c. Click **Save**.

Complete the following steps to download multiple files at once.

- a. Select the checkbox corresponding to each file you want to download.
- b. Click the **Download Multiple Files** icon (top right corner).
- c. Specify a name for the compressed folder type.
- d. Click **Save**

Deleting file(s)

Complete the following steps to delete single file:

- a. Click the **Actions** menu corresponding to the file you want to delete and select **Delete**.
- b. Select **Yes** to confirm the deletion.

Complete the following steps to delete multiple files:

- a. Select the checkbox corresponding to each file you want to delete.
- b. Click the **Delete Multiple Files** icon (top left corner).
- c. Select **Yes** to confirm the deletion.

Uploading/Downloading a File Using PAR URL

Complete the following steps to upload or download a file using the PAR URL.

Figure 2-1 Get PAR URL

1. Click **Drag and Drop** to browse and select a file for upload from the local directory.
You can also browse to the local directory from the File Explorer and select the file and drop it. The file name is automatically updated in the **Selected File**.
2. Enter the **Prefix** to be added to the file name.
3. Click **Get PAR URL**, to generate the PAR URL and File ID which are required to upload the file.

You can also generate PAR URL using Rest API. For more information refer to, [Calling the API to Generate the URL](#).

4. Copy the PAR URL and note the related File ID.
5. Upload the file content referred with the specific PAR URL into the object store using the Console, CLI, or SDK.

To upload using the CLI, enter the following curl command directly in local Gitbash.

```
curl -X PUT --data-binary '@<local-filename>' <unique-PAR-URL>
```

You can also use the following command.

```
curl -T '<Filepath>' -X PUT <PAR_URL>
```

6. Scan the file referred with the specific File ID (obtained in Step 3) using Console, CLI, or SDK.

Use the following CURL command, to Scan using CLI:

```
curl -k --location --request PUT 'https://<Host:Port>/<Tenant-ID>/utils-service/v1/file/scan/<FileID>' \
--header 'ofs_tenant_id: <Tenant-ID>' \
--header 'ofs_service_id: <Service-ID>' \
--header 'ofs_workspace_id: <WorkspaceId>' \
--header 'Authorization: Bearer <Generated_Token>'
```

To generate a bearer token, refer to [Generate access token](#).

A confirmation message is displayed after successful upload and the file is listed in the Uploaded Files list.

3

File Upload Automation

To simplify the file upload process, configure and execute the File Upload automation utility.

File Upload automation can be implemented by completing the following steps in sequence.

- [Step 1: Generate Access Token](#)
- [Step 2: Generate PAR URL](#)
- [Step 3: Upload file to Object Store](#)
- [Step 4: Scan the file to ensure Upload was Successful](#)

Step 1: Generate Access Token

Generate the Access Token for your Profitability and Balance Sheet Management Cloud Service by:

1. Submitting a RESTful API Post Request to your Oracle IAM environment as defined in the Identity Cloud Service User Guide. For information, see [OAuth Runtime Tokens REST Endpoints](#).

Note that a sample code snippet has been provided below using cURL to generate the access token for Basic Authorization and assign it to a variable for use within a script:

```
access_token=`curl -s --insecure -H "Authorization: Basic $ENCODED" -H
"Content-Type:application/x-www-form-urlencoded;charset=UTF-8" --request
POST $IDCS_URL -
d"grant_type=password&username=$USERNAME&password=$PASSWORD&scope=urn:opc:ldm:
__myscopes__urn:opc:resource:expiry=9153600" | python -c "import sys,
json;print(json.load(sys.stdin)['access_token'])"`
```

Step 2: Generate PAR URL

Generate the PAR URL for your Profitability and Balance Sheet Management Cloud Service by:

1. Submitting a RESTful API Post Request to your Cloud Service as defined in the [Calling the API to Generate the URL](#) section.

Note a sample code snippet has been provided below using cURL to generate the PAR URL and assign it to a variable for use within a script:

PAR URL Generation Code Snippet

```
curl --location --insecure --request POST "$FILEUPLOADURL" --header
"Authorization: Bearer $access_token" --header 'Content-Type: application/
json' --data-raw "{
  \"fileName\": \"$1\",
  \"fileSize\": \"$2\",
```

```
\ "mimeType\": \"\$3\"
}" >> "$HOME"/FILEUPLOAD_UTIL/"$1"_PARURLresponse.out 2>&1
```

PAR URL Variable Assignment Code Snippet

```
grep -oE '(https)[^ ]*' "$HOME"/FILEUPLOAD_UTIL/"$1"_PARURLresponse.out >
"$HOME"/FILEUPLOAD_UTIL/"$1"_PARURLresponse1.csv
while IFS="," read -r H1 H2
do
    URLtrim="$H1"
    FinalPAR=${URLtrim::-1}
    echo -e "\n PAR_URL is $FinalPAR"
done < "$HOME"/FILEUPLOAD_UTIL/"$1"_PARURLresponse1.csv
```

PAR URL Variable Assignment Code Snippet – Used in File Scanning Step

```
grep -oE '(fileId)[^ ]*' "$HOME"/FILEUPLOAD_UTIL/"$1"_PARURLresponse.out >
"$HOME"/FILEUPLOAD_UTIL/"$1"_PARURLresponse2.csv
while IFS="," read -r F1 F2
do
    FILEIDtrim="$F1"
    FINALFILEID=`echo "$FILEIDtrim" | sed -r 's/^.{8}//'`

    echo -e "\n FILE ID is $FINALFILEID"
done < "$HOME"/FILEUPLOAD_UTIL/"$1"_PARURLresponse2.csv
```

Step 3: Upload file to Object Store

Upload the file to the Object Store of your Profitability and Balance Sheet Management Cloud Service by:

1. Submitting a RESTful API Post Request to your Cloud Service as defined in the [Uploading/Downloading a File Using PAR URL](#) topic.

Note a sample code snippet has been provided below using cURL to upload the file:

```
curl -T "$HOME"/FILEUPLOAD_UTIL/"$1" -X PUT "$FinalPAR"
```

Step 4: Scan the file to ensure Upload was Successful

Scan the file that was uploaded to the Object Store of your Profitability and Balance Sheet Management Cloud Service by:

1. Submitting a RESTful API Put Request to your Cloud Service as defined by the code snippet below:

File Scanning Code Snippet – Using File ID from Step 2 – Generate PAR URL

```
if [ $last_error -eq 0 ]; then
    echo -e "\n *****Scanning the File *****"
    curl -k --location --request PUT "$SCANURL/$FINALFILEID" --header
'ofs_tenant_id: $TENANT' --header 'ofs_service_id: OFS_FTP' --header
'ofs_workspace_id: WS001' --header "Authorization: Bearer $access_token"
```

```

last_error=$?
if [ $last_error -eq 0 ]; then
    echo -e "\n ***File Upload is Successful please check File
Upload / Download UI***"
else
    echo "Scan failed"
    exit -2;
fi
else
    echo "Upload failed"
    exit -3;
fi

```

Automating the File Upload Process Using File Upload Utility

This section provides the procedure including the script to automate the process of uploading input data files using the File Upload utility.

You can download this script from this MoS Doc ID [2927077.1](#).

Executing the File Upload Automation Script

File Upload Automation script assists you to upload the files seamlessly.

Complete the following procedure to execute the file upload automation script.

Python 3.10 is required to access data elements from the API JSON responses.

1. Extract the **FILEUPLOAD_UTIL.zip** file located in the **\$HOME** directory.
2. Copy the Data Loader input file (text file) to the **\$HOME/FILEUPLOAD_UTIL** directory.
3. Edit the script **Env_setup.sh** file to update the following environment variables.

- **IAM URL** - The Service Instance URL to access your IAM console.
You can get the IAM URL from the following menu:

Oracle Cloud Infrastructure Console > Identity Cloud.

Syntax: <IAM-url>/oauth2/v1/token

- **Encoded** - The <OAuth Client ID>:<OAuth Client Secret> encoded using **base64encode**
To extract and encode the Client ID and Client Secret, refer to the following steps:

- a. Login to **Admin Console**.
- b. Go to **System Configuration** tab, and click **Component Details** tile.
- c. Click **OAUTH Creds** tab to view and copy the **OAUTH Client ID** and **OAUTH Client Secret** details.
- d. Using any **base64encode** utility, encode <OAUTH Client ID>:<OAUTH Client Secret>.

Example (Input OAUTH Client ID and Client Secret):

ftptenant-prd_APPID:99140e14-4d30-4e86-85fb-09501fe45fe0

Example (Encoded OAUTH Client ID and Client Secret):

ZnRwcWExMDEyMzEtcHJkX0FQUElEOjBkMmU5MDBiLTlhYjItNGFmOS05OWM0LTEwNTYyMDV
kYWYwNQ==

- **Username** - The Username to access the application.

 Note:

The user should have appropriate roles and privileges. For more information about roles and privileges, refer to [Roles and Functions](#).

- **Password** - The password to login to the application.
 - **Tenant** - The tenant associated with the application.
 - **PBSM Host** - Details of the PBSM host on which the application is hosted.
- Sample Env_setup.sh**

```
IDCS_URL=https://idcs-xyz123.identity.c9xyz.oc9xyz.com/oauth2/v1/token
ENCODED=ZnRwbWFydXAxNDIyMzEtchJkX0FQUElEOjk5MTQwZTE0LTRkMzAtNGU4Ni04NWZi
LTA5NTAxZmU0NWZlMA==
USERNAME=<user_name>
PASSWORD=<password>
TENANT=<tenant-prd>
PBSMHOST=dc.pbsmcloud.us-xxxxx -1.ocs.oc-test.com
```

4. Execute **PBSMCS_PAR_fileupload.sh** with the following parameters –

- **filename** - The file to be uploaded
- **filesize** - The file size in Bytes
- **Mimetype** - The mimetype of the file.

Format: Filename<space>Filesize<space>mimetype

Example: ./PBSMCS_PAR_fileupload.sh input_20150101_filename_example_8007.txt
334 text/plain

This script generates the access token and the PAR URL. It also uploads the file into the object store and scans the file too.

 Note:

To upload multiple files, you must execute the script for each file, separately.

Once the script is executed successfully, the file is uploaded and added to the list of files in the File Upload/Download page. To access the File Upload/Download page refer to [Upload or Download File from Object Store Using Console](#).

4

Generating PAR URL for File Operations

The PAR URL for File Operations API creates a PAR File that you can use to perform file operations in the Object Store for end-to-end integrations.

Generating PAR URL for File Upload

Generate PAR URL for File Upload

You can use this REST API to generate the PAR URL for File Upload. See the following sections for information on how to perform the POST operation.

- [End Point Details](#)
- [Calling the API to Generate the URL](#)

End Point Details

- **Method** – POST
- **URL** - `https://<HOST_NAME:PORT>/<TENANT>/utils-service/v1/file/uploadfile/parURL?prefix=<prefix>`
- **Content-Type** - Application/Json

Calling the API to Generate the URL

To call the API:

1. Open a relevant tool, such as via cURL command.
2. Prepare a cURL command with the authentication token and other details. For more information refer to the following code.

Syntax

```
curl -k --location --request POST 'https://<hostname>/<TENANT-ID>/utils-service/v1/file/uploadfile/parURL?prefix=' \
--data-raw '{"fileName": "<remote filename>", "fileSize": <file size>, "mimeType": "<file type>"}' \
--header 'ofs_remote_user: <USERID>' \
--header 'locale: en-US' \
--header 'ofs_tenant_id: <TENANT-ID>' \
--header 'ofs_workspace_id: WS001' \
--header 'content-type: application/json' \
--header 'Authorization: Bearer <TOKEN>'
```

Example (truncated)

```
curl -k --location --request POST 'https://<hostname>/<TENANT-ID>/utils-service/v1/file/uploadfile/parURL?prefix=' \
--data-raw '{"fileName": "idcs_log1.txt", "fileSize": 100, "mimeType":
```

```

"text/plain"}' \
--header 'ofs_remote_user: cneadmin' \
--header 'locale: en-US' \
--header 'ofs_tenant_id: aaitestdev1001-prd' \
--header 'ofs_workspace_id: WS001' \
--header 'content-type: application/json' \
--header 'Authorization: Bearer
eyJ4NXQjUzI1NiI6Ildia25rQUR5TUZIMlhlQ1pKcTY1c3o4VzdEVWhKa0s4MldYY0hadk4wWkk
iLCJ4
...
...
sQXj0iohsSIEmQXVwjjhhqnc4eJNnmCjx8Tb7TXjx1MIQLeOIcfrIj5gkzoMKX94_7USxHv-6Lh
Bzw'

```

Request JSON Parameters

This section provides the list of parameters in the JSON Request.

Table 4-1 Request JSON Parameters

Name	Type	Required	Description
fileName	STRING	Yes	The name of the file to be uploaded. The following are the conditions for to enter in this field: • Must start with an Alphanumeric Character • Allowed characters are alphabets, numbers, and special characters - hyphen(-), dot(.), and underscore(_) • Length of characters must not be greater than 255 characters
fileSize	INTEGER	Yes	The size of the file (in Bytes) to be uploaded. The size of the file should be greater than 1 Byte and should be less than 10 TB. It is recommended to use multipart upload for uploading files of size more than 100 MB. For more information about uploading large objects and multipart uploads, refer to Working with Pre-Authenticated Requests.
contentType	STRING	Yes	The mime type to be uploaded. The following mime types are allowed: • Text/CSV • Text/plain • DAT

Request JSON Sample

```

[
  {
    "fileName": "File.csv",
    "fileSize": 7654,
    "contentType": "text/csv"
  }
]

```

Response JSON Parameters

This section provides the list of parameters in the JSON Response.

Table 4-2 POST JSON Response

Name	Type	Description
fileName	STRING	The name of the file to be uploaded.
uploadURL	STRING	The generated pre-authenticated URL to upload a file.
fileId	INTEGER	The unique File Identifier.

Response JSON Sample

```
{
  "payload": {
    "uploadURL": "https://objectstorage.us-phoenix-1.oraclecloud.com/p/
bdSI-hzigiAoUU0lyEKnuK0YGs05L172gt_woZAgqNFYmUFQeexV3BDfT097mhBI/n/
oraclegbudevcorp/b/fsgbu_pbsm_cndevcorp_ftpqa101231-prd_default/o/default/
2023-01-31/jfr/f9ce031f-4a42-471d-b4da-d0577f3eca15",
    "createUser": "user1",
    "stripeName": "default",
    "fileId": 5025,
    "createDate": "2023-01-31T09:14:16",
    "token": "",
    "status": "success"
  }
}
```

Viewing List of Uploaded Files

Run the following cURL command to generate and view all the files that are uploaded using PAR URL.

Syntax

```
curl -k --location --request GET 'https://<hostname>/<TENANT-ID>/utils-
service/v1/listfiles stripeName=default' \
--header 'locale: en-US' \
--header 'ofs_remote_user: <user id>' \
--header 'ofs_tenant_id: < TENANT-ID >' \
--header 'ofs_workspace_id: WS001' \
--header "Authorization: Bearer <TOKEN>"
```

Example

```
curl -k --location --request GET 'https://dc.pbsmcloud.us-phoenix-1.ocs.oc-
test.com/aaitestdev1001-prd/utils-service/v1/listfiles?stripeName=default' \
--header 'locale: en-US' \
--header 'ofs_remote_user: cneadmin' \
--header 'ofs_tenant_id: aaitestdev1001-prd' \
--header 'ofs_workspace_id: WS001' \
--header "Authorization: Bearer ${TOKEN}"
```

Response

```
{
  "payload": [
    {
      "v_file_name": "Idcs_log3.txt",
      "n_file_id": 9916,
      "d_upload_date": "31-JAN-23 06:33:43 AM",
      "v_stripe_name": "default"
    },
    {
      "v_file_name": "Idcs_log4.txt",
      "n_file_id": 9917,
      "d_upload_date": "31-JAN-23 06:40:25 AM",
      "v_stripe_name": "default"
    }
  ],
  "count": 2
}
```

Generating PAR URL For File Download

You can use this REST API to generate the PAR URL for File Download. See the following sections for information on how to perform the post operation.

- [Calling the API to Generate PAR URL for File Download Using File Name](#)
- [Calling the API to Generate PAR URL for File Download Using File ID](#)

Calling the API to Generate PAR URL for File Download Using File Name

To call the API:

1. Open a relevant tool, such as via the cURL command.
2. Prepare a cURL command with the authentication token and other details. For more information refer to the following code.

Syntax

```
curl -k --location --request GET <'https://<hostname>/<TENANT-ID>/utils-service/v1/file/download?fileName=<file name>&stripeName=default&prefix=' \
--header 'ofs_remote_user: <userid>' \
--header 'locale: en-US' \
--header 'ofs_tenant_id: <TENANT-ID>' \
--header 'ofs_workspace_id: WS001' \
--header "Authorization: Bearer <TOKEN>"
```

Example

```
curl -k --location --request GET 'https://<hostname>/<TENANT-ID>/utils-service/v1/file/download?fileName=test3GB.xml&stripeName=default&prefix=' \
--header 'ofs_remote_user: cneadmin' \
--header 'locale: en-US' \
--header 'ofs_tenant_id: aaitestdev1001-prd' \
--header 'ofs_workspace_id: WS001' \
--header "Authorization: Bearer ${TOKEN}"
```

Response

```
{
  "payload": {
    "downloadURL": "https://objectstorage.us-phoenix-1.oraclecloud.com/p/8R68eVcQAxQjNjK__S04MZjS-v4BqEbWSILvu0w40kJNrzfKeCB8vWBwugW5XvsK/n/oraclegbudevcorp/b/fsgbu_pbsm_cndevcorp_aaitestdev1001-prd_default/o/default/2023-01-20/rnz/6c023e75-09e2-4265-815e-32cedcd2415e?httpResponseContentDisposition=ATTACHMENT%3B%20filename%3Dtest3GB.xml"
  }
}
```

Calling the API to Generate PAR URL for File Download Using File ID

To call the API, follow these steps:

1. Open a relevant tool, such as via the cURL command.
2. Prepare a cURL command with the authentication token and other details. For more information, refer to the following code.

Syntax

```
curl -k --location --request GET 'https://<hostname>/<TENANT-ID>/utils-  
service/v1/file/downloadfile/<file id>' \  
--header 'ofs_remote_user: <userid>' \  
--header 'locale: en-US' \  
--header 'ofs_tenant_id: < TENANT-ID> ' \  
--header 'ofs_workspace_id: WS001' \  
--header "Authorization: Bearer <TOKEN>"
```

Example

```
curl -k --location --request GET 'https://<hostname>/<TENANT-ID>/utils-  
service/v1/file/downloadfile/9916' \  
--header 'ofs_remote_user: cneadmin' \  
--header 'locale: en-US' \  
--header 'ofs_tenant_id: aaitestdev1001-prd' \  
--header 'ofs_workspace_id: WS001' \  
--header "Authorization: Bearer ${TOKEN}"
```

Response

```
{"payload":{"downloadURL":"https://objectstorage.us-  
phoenix-1.oraclecloud.com/p/  
gTxxzhqLEea4Or2TRkBqTqHxt_JogVF9G_0wtN8NYy_op0Zk4lvKGDxxeXGhLq7/n/  
oraclegbudevcorp/b/fsgbu_pbsm_cndevcorp_aaitestdev1001-prd_default/o/  
default/2023-01-31/fae/2d63d2fe-2090-4fb7-a4c8-9940d22987db?  
httpResponseContentDisposition=ATTACHMENT%3B%20filename%3DIdcs_log3.txt"}}
```

5

Deleting A File

Delete (DELETE) API helps to delete an uploaded file.

For more information about the Delete API, refer to [Endpoint Details](#).

You can delete a file using one of the following methods:

- [Using File ID](#)
- [Using File Name](#)
- [Delete multiple Files using File Names](#)
- [Delete files using a prefix value](#)

Endpoint Details

Delete (DELETE) API helps to delete an uploaded file.

- **HTTP Method** - Delete
- **Header Parameters**
 - **ofs_remote_user** - User ID of the user mapped to 'BATCH_EXEC' function.
 - **locale** - locale in languageCode-countryCode format. For example, en-US.
 - **ofs_tenant_id** - Tenant ID of the Application
 - **ofs_workspace_id** - Workspace ID of the Application. It is defaulted to ws001 and same should be passed each time.
 - **Content-type** - The media type of the body of the request. The content-type details are required for POST and PUT requests, and the supported types vary with each endpoint. The value is application/json.
 - **Authorization** - Access token required to authenticate the API. If this token is not provided, 401 Unauthorized error is generated. For more information about Bearer token, refer to [Generate the Access Token](#).

Deleting a File Using the File ID

Delete a file from the object store, using the file ID as the reference.

To delete a file:

1. Open a relevant tool, such as via cURL command.
2. Prepare a cURL command with the authentication token and other details. For more information refer to the following code.

Syntax

```
curl -k --location --request POST 'https://<hostname>/<TENANT-ID>/utils-  
service/v1/file/deletefile/{fileId}  
--header 'ofs_remote_user: <USERID>' \
```

```
--header 'locale: en-US' \
--header 'ofs_tenant_id: <TENANT-ID>' \
--header 'ofs_workspace_id: WS001' \
--header 'content-type: application/json' \
--header 'Authorization: Bearer <TOKEN>'
```

Example (truncated)

```
curl -k --location --request POST 'https://<hostname>/<TENANT-ID>/utils-
service/v1/file/deletefile/5
--header 'ofs_remote_user: cneadmin' \
--header 'locale: en-US' \
--header 'ofs_tenant_id: aaitestdev1001-prd' \
--header 'ofs_workspace_id: WS001' \
--header 'content-type: application/json' \
--header 'Authorization: Bearer
eyJ4NXQjUzI1NiI6Ildia25rQUR5TUZIM1hlQ1pKcTY1c3o4VzdEVWhKa0s4MldYY0hadk4wWkk
iLCJ4
...
...
sQXj0iohsSIEmQXVwvjhhqnc4eJNnmCjx8Tb7TXjx1MIQLeOIcfrIj5gkzoMKX94_7USxHv-6Lh
Bzw'
```

Response

```
{"payload":"File Deleted Successfully"}
```

Deleting a File Using Filename

Delete a file from the object store, using the file name as the reference.

To delete a file:

1. Open a relevant tool, such as via cURL command.
2. Prepare a cURL command with the authentication token and other details. For more information refer to the following code.

Syntax

```
curl -k --location --request POST 'https://<hostname>/<TENANT-ID>/utils-
service/v1/file/deletefilename/{filename}?prefix=<foldername>
--header 'ofs_remote_user: <USERID>' \
--header 'locale: en-US' \
--header 'ofs_tenant_id: <TENANT-ID>' \
--header 'ofs_workspace_id: WS001' \
--header 'content-type: application/json' \
--header 'Authorization: Bearer <TOKEN>'
```

Note:

Prefix is an optional parameter.

Example (truncated)

```
curl -k --location --request POST 'https://<hostname>/<TENANT-ID>/utils-
service/v1/file/deletefilename/test.txt?prefix=folder1
--header 'ofs_remote_user: cneadmin' \
--header 'locale: en-US' \
--header 'ofs_tenant_id: aaitestdev1001-prd' \
--header 'ofs_workspace_id: WS001' \
--header 'content-type: application/json' \
--header 'Authorization: Bearer
eyJ4NXQjUzI1NiI6Ildia25rQUR5TUZIMlhlQ1pKcTY1c3o4VzdEVWhKa0s4MldYY0hadk4wWkk
iLCJ4
...
sQXj0iohsSIEmQXVwvjhhqnc4eJNnmCjx8Tb7TXjx1MIQLeOicfrIj5gkzoMKX94_7USxHv-6Lh
Bzw'
```

Response

```
{"payload": "File Deleted Successfully"}
```

Deleting Multiple Files Using Filenames

Delete multiple files from the object store, using the file names as the reference.

To delete multiple files:

1. Open a relevant tool, such as via cURL command.
2. Prepare a cURL command with the authentication token and other details. For more information refer to the following code.

Syntax

```
curl -k --location --request POST 'https://<hostname>/<TENANT-ID>/utils-
service/v1/file/deletefilenames/{filenames}
--data-raw '[filenames]' \
--header 'ofs_remote_user: <USERID>' \
--header 'locale: en-US' \
--header 'ofs_tenant_id: <TENANT-ID>' \
--header 'ofs_workspace_id: WS001' \
--header 'content-type: application/json' \
--header 'Authorization: Bearer <TOKEN>'
```

Example (truncated)

```
curl -k --location --request POST 'https://<hostname>/<TENANT-ID>/utils-
service/v1/file/deletefilenames
--data-raw '["filename1.txt","filename2.txt","filename3.txt"]' \
--header 'ofs_remote_user: cneadmin' \
--header 'locale: en-US' \
--header 'ofs_tenant_id: aaitestdev1001-prd' \
--header 'ofs_workspace_id: WS001' \
--header 'content-type: application/json' \
--header 'Authorization: Bearer
```

```

eyJ4NXQjUzI1NiI6Ildia25rQUR5TUZIMlhlQ1pKcTY1c3o4VzdEVWhKa0s4MldYY0hadk4wWkk
iLCJ4
...
...
sQXj0iohsSIEmQXVwvjhhqnc4eJNnmCjx8Tb7TXjx1MIQLeOIcfrIj5gkzoMKX94_7USxHv-6Lh
Bzw'

```

Response

```
{"payload": "File Deleted Successfully"}
```

Delete files using a prefix value

Delete files from the object store, using a prefix value as the reference.

To delete files using a prefix value:

1. Open a relevant tool, such as via cURL command.
2. Prepare a cURL command with the authentication token and other details. For more information refer to the following code.

Syntax

```

curl -k --location --request POST 'https://<hostname>/<TENANT-ID>/utils-
service/v1/file/deleteprefix/<prefix_value>
--header 'ofs_remote_user: <USERID>' \
--header 'locale: en-US' \
--header 'ofs_tenant_id: <TENANT-ID>' \
--header 'ofs_workspace_id: WS001' \
--header 'content-type: application/json' \
--header 'Authorization: Bearer <TOKEN>'

```

Example (truncated)

```

curl -k --location --request POST 'https://<hostname>/<TENANT-ID>/utils-
service/v1/file/deleteprefix/prefixvalue1
--header 'ofs_remote_user: cneadmin' \
--header 'locale: en-US' \
--header 'ofs_tenant_id: aaitestdev1001-prd' \
--header 'ofs_workspace_id: WS001' \
--header 'content-type: application/json' \
--header 'Authorization: Bearer
eyJ4NXQjUzI1NiI6Ildia25rQUR5TUZIMlhlQ1pKcTY1c3o4VzdEVWhKa0s4MldYY0hadk4wWkk
iLCJ4
...
...
sQXj0iohsSIEmQXVwvjhhqnc4eJNnmCjx8Tb7TXjx1MIQLeOIcfrIj5gkzoMKX94_7USxHv-6Lh
Bzw'

```

Response

```
{"payload": "Files Deleted Successfully"}
```