File*Catalyst* TransferAgent REST API: Usage Examples

Table of Contents

Introduction

This document is designed to serve as a language-agnostic guide on how to carry out common listing and transfer operations using the File*Catalyst* TransferAgent REST services API. The majority of these operations all have the same initial requirements, you must have:

- An instance of File*Catalyst* TransferAgent running
- A file-transfer server instance running

Note:

- If you are attempting to connect to a File*Catalyst* Server, then the File*Catalyst* Server's license must support the "2-way applet" feature. If the File*Catalyst* Server license doesn't include this feature, contact your File*Catalyst* Server Administrator or your File*Catalyst* sales representative to adding this feature to your license.
- If you are attempting to connect to a Third-Party transfer server, then you need to license the File*Catalyst* TransferAgent with a license that includes the "Connect to 3PP Server" feature. If you do not have a license that supports this functionality, contact your File*Catalyst* provider.

1 Connecting FileCatalyst TransferAgent to a FileCatalyst Server

To connect your instance of FileCatalyst TransferAgent to a FileCatalyst Server, you must invoke either the "/rs/agent/connect" or the "/rs/agent/register" REST services. By evoking either of these services, the FileCatalyst TransferAgent constructs a client, connects it to the FileCatalyst Server, and establishes a connection key that will be used in any future REST service that communicates with the FileCatalyst Server.

There are two main methods in which these REST services can be invoked. The first method is through the GET request on the connect/register resources. This method uses plain-text query parameters to pass along the necessary connection information to the FileCatalyst TransferAgent, so it properly establishes the connection. The second method is to evoke the POST request available the connect/register resources. This method uses an encrypted PGP message (you can generate these via encrypt.html) to safely hide FileCatalyst Server connection information with the XHR payload, and using it will provide an overall more secure method for connecting to FileCatalyst Servers.

Note: To gain access to a PGP message for your FileCatalyst Server, you must visit a web page entitled encrypt.html. This web page is bundled within all FileCatalyst TransferAgent Deployments and can be found at the base of the Deployment package. By using this small web application, you will be able to provide the connection information for any given FileCatalyst Server and use that to generate an encrypted PGP message for the connection.

1.1.1 Connecting via "/rs/agent/connect" GET Request example

- HTTP Method: **GET**
  - REST URL Example: https://localhost.filecatalyst.net:12680/rs/agent/connect?remoteServer=127.0.0.1&remotePort=21&username=user&password=pwd&usesSSL=false
- API Documentation for this resource: Documentation Link

1.1.2 Connecting via "/rs/agent/connect" POST Request example

- HTTP Method: **POST**
  - REST URL Example: https://localhost.filecatalyst.net:12680/rs/agent/connect
- API Documentation for this resource: Documentation Link

2 Retrieving a local file list

To retrieve a local file listing from the File*Catalyst* TransferAgent, you need to invoke the REST resource located at "/rs/files/local". Submitting a request at the root level of this resource will provide you with all of the file system roots that the File*Catalyst* TransferAgent has access to. Submitting requests with a file path parameter after the "rs/files/local" will give you a directory listing for the specified directory.

Notes:

- When navigating through local directory trees it is recommended that you use the "path" parameter contained within the FileObjectModel returned through the "/rs/files/local" REST response. This will ensure seamless navigation across the file system.
- Local file browsing with the File*Catalyst* TransferAgent also supports EFS paths if the File*Catalyst* TransferAgent has had EFS functionality enabled. For steps on how to accomplish this, please visit Enabling external file-system support for the FileCatalyst TransferAgent.

2.1.1 Local file root listing example

- HTTP Method: **GET**
  - REST URL Example: https://localhost.filecatalyst.net:12680/rs/files/local
- API Documentation for this resource: Documentation Link

2.1.2 Specific directory local file listing example

- HTTP Method: **GET**
  - REST URL Example: https://localhost.filecatalyst.net:12680/rs/files/local/C:/temp/SomeDirectory/
- API Documentation for this resource: Documentation Link

3 Retrieving a remote file list

To retrieve a remote file listing from the File*Catalyst* TransferAgent, you need to invoke two REST calls:

3.1 Generate a connection key for the remote File*Catalyst* Server

- HTTP Method: **GET** / **POST**
  - REST URL Example: https://localhost.filecatalyst.net:12680/rs/agent/connect
- API Documentation for this resource: Documentation Link

3.2 Construct an "rs/files/remote" REST call with the generated connection key

Once the connection key for the File*Catalyst* Server has been generated, you must construct a call to the RemoteResource REST entry point using the key as a query parameter. Submitting the request to the root level of this resource (i.e. no directory provided), will give you all of the files and directories present within the connected user's home directory. Submitting requests with a remote file path parameter after the "rs/files/remote" URL portion will give you a directory listing of the specified remote directory.

Note: It is recommended that you use the "path" parameter contained within the FileObjectModel returned through the "/rs/files/remote" REST response. This will ensure seamless navigation across the remote file system.

3.2.1 Remote file root listing example

- HTTP Method: **GET**
  - REST URL Example:
    https://localhost.filecatalyst.net:12680/rs/files/remote?connectionKey=57a608828b7b9901c9499fffde001099df4615b57bd619f510c37a8c47c5cb70aa0af48353b283ea2841e8f6a853a403e92a0cbf1ab38f73263c670b1e2ec87343ca69e044d6bd39
- API Documentation for this resource: Documentation Link

3.2.2 Remote file listing of a specific directory (EG: someDir/innerDir) example

- HTTP Method: **GET**
  - REST URL Example:
    https://localhost.filecatalyst.net:12680/rs/files/remote/someDir/innerDir?connectionKey=57a608828b7b9901c9499fffde001099df4615b57bd619f510c37a8c47c5cb70aa0af48353b283ea2841e8f6a853a403e92a0cbf1ab38f73263c670b1e2ec87343ca69e044d6bd39
- API Documentation for this resource: Documentation Link

4 Performing an upload transfer

To perform an upload transfer you need to perform the following steps:

4.1 Generate a connection key for the File*Catalyst* Server

- HTTP Method: **GET** / **POST**
  - REST URL Example: https://localhost.filecatalyst.net:12680/rs/agent/connect
- API Documentation for this resource: Documentation Link

4.2 Generate a TransferFilesRequestModel containing data for the Transfer

Once a connection has been established you must construct a TransferFilesRequestModel JSON object that contains the data for the transfer. In this object, you must define three key items:

- SourceConnectionKey: This is set to the string "local". By setting this, you inform the File*Catalyst* TransferAgent that the source of files in the transfer will be the File*Catalyst* TransferAgent itself.
- DestinationConnectionKey: This is set to the connection key generated in the previous step. This informs the File*Catalyst* TransferAgent where the files are going to be transferred to.
- fileList: This contains the list of local files that you are transferring to the remote File*Catalyst* Server.
  Note: This field supports EFS paths if the File*Catalyst* TransferAgent has had EFS functionality enabled. For steps on how to enable EFS functionality, please see Enabling external file-system support for the FileCatalyst TransferAgent.

4.2.1 Upload TransferFilesRequest JSON example

```
{
        "SourceConnectionKey": "local",
        "DestinationConnectionKey":
        "57a608828b7b9901c9499fffde00109908cb791734ba59823155420e0d7418df
        84e53336d1b73ee97009c186518481fc0752ec3f6d53f8fd33e35c7267c16a8c9
        af9fa11d8d0ba0d5bc884d71dcf0bdc5805a192ae4ca8e3eb7f19daaec059eb",
        "fileList": [
                    "C:/Users/examp/Downloads/api-json-client.jar"
                    ]
}
```

4.3 Submit the TransferFilesRequestModel to the "rs/files/transferRev1" resource

After the TransferFilesRequestModel has been generated, you will submit a REST call to the "/rs/files/transferRev1" resource. This will submit the transfer request to the File*Catalyst* TransferAgent, where it will then be validated, and started if it succeeds validation succeeds. If the request is properly acknowledged by the File*Catalyst* TransferAgent it will return with a TransferFilesRequestModel JSON response that will contain a job ID. This job ID can be used to poll on the status of the transfer while it executes.

4.3.1 REST Example:

- HTTP Method: **POST**
  - REST URL Example: https://localhost.filecatalyst.net:12680/rs/files/transferRev1
- API Documentation for this resource: Documentation Link

4.4 Polling on the status of the transfer through "/rs/agent/status" resource

While the transfer is running you can use the job ID to make status requests to the File*Catalyst* TransferAgent to get the ongoing status of the transfer. These status objects provide you with statistics and overall information about the transfer's current progress.

4.4.1 REST Example:

- Example JobID: n685vp263gailg39dgtpgtqe9g_M274724060473474-3-NULL
- HTTP Method: **GET**
  - REST URL Example: https://localhost.filecatalyst.net:12680/rs/agent/status/n685vp263gailg39dgtpgtqe9g_M274724060473474-3-NULL
- API Documentation for this resource: Documentation Link

5 Performing a download transfer

To perform a download transfer you need to perform the following steps:

5.1 Generate a connection key for the File*Catalyst* Server

- HTTP Method: **GET** / **POST**
  - REST URL Example: https://localhost.filecatalyst.net:12680/rs/agent/connect
- API Documentation for this resource: Documentation Link

5.2 Generate a TransferFilesRequestModel containing data for the Transfer

Once a connection has been established you must construct a TransferFilesRequestModel JSON object that contains the data for the transfer. In this object, you must define three key items:

- SourceConnectionKey: This is set to the connection key generated by the request in the previous step.
- DestinationConnectionKey: This is set to the string "local". By setting this, you are informing the File*Catalyst* TransferAgent that the destination for the files will be the File*Catalyst* TransferAgent's local file system.
- fileList: This contains the list of remote files that will be transferred to the remote File*Catalyst* Server.
- LocalWorkingDirectory: This is set to the directory that you want to store the downloaded files in.

  Note: This field supports EFS paths if the File*Catalyst* TransferAgent has had EFS functionality enabled. For steps on how to enable EFS functionality, please see Enabling external file-system support for the FileCatalyst TransferAgent.

5.2.1 Example Download TransferFilesRequest JSON

```
{
        "SourceConnectionKey":
        "57a608828b7b9901c9499fffde00109908cb791734ba59823155420e0d7418df
        84e53336d1b73ee97009c186518481fc0752ec3f6d53f8fd33e35c7267c16a8c9
        af9fa11d8d0ba0d5bc884d71dcf0bdc5805a192ae4ca8e3eb7f19daaec059eb",
        "DestinationConnectionKey": "local",
        "fileList": [
                    "someDirectory/api-json-client.jar",
                    "testFile1.txt"
                    "testFile2.txt"
                    "testFile3.txt"
                    ],
        "LocalWorkingDirectory" : "C:/Users/example/Downloads/"
}
```

5.3 Submitting the download TransferFilesRequest to the File*Catalyst* TransferAgent

Once the transfer request has been generated, it can then be submitted to the "rs/files/transferRev1" resource. Upon submission, the File*Catalyst* TransferAgent will start the

asynchronous download, and then return a response containing a job id, that can be used to poll for the ongoing status of the job.

5.3.1 REST Example:

- HTTP Method: **POST**
  - REST URL Example: https://localhost.filecatalyst.net:12680/rs/files/transferRev1
- API Documentation for this resource: Documentation Link

5.4 Polling for the status of the download transfer

After the transfer has started, you can poll on the transfer status through the use of the job ID returned by the previous step.

5.4.1 REST Example

- Example JobID: n685vp263gailg39dgtpgtqe9g_M274724060473474-3-NULL
- HTTP Method: **GET**
  - REST URL Example: https://localhost.filecatalyst.net:12680/rs/agent/status/n685vp263gailg39dgtpgtqe9g_M274724060473474-3-NULL
- API Documentation for this resource: Documentation Link

6 Performing REST calls against a remote File*Catalyst* TransferAgent

Normally the File*Catalyst* TransferAgent only supports receiving REST calls from a connection from localhost, or an IP address that matches the machine that it is currently running on. While this is the standard functionality there is a way to setup the File*Catalyst* TransferAgent application to receive REST calls from external sources.

6.1 Configuring network / firewall so REST calls can be executed

To allow the File*Catalyst* TransferAgent to receive REST calls from external resources you must ensure that the File*Catalyst* TransferAgent's HTTP port can be accessed by external systems. To accomplish this, you must enable the current File*Catalyst* TransferAgent HTTP port for incoming TCP connection on your firewall. Failure to do so will make it impossible for external systems to communicate with the File*Catalyst* TransferAgent.

6.2 Licensing the File*Catalyst* TransferAgent with the "Remote Control" functionality

Once the proper network configurations have been set up, you must license the File*Catalyst* TransferAgent with a license that supports the "Remote Control" feature. If you currently don't have a license, or the license you have doesn't support the feature, contact your File*Catalyst* provider and request a license that supports this functionality.

6.3 Applying a credential set that will be used for Remote Control access

With the File*Catalyst* TransferAgent licensed for Remote Control access, you must set up a credential set that will be used to validate REST requests coming from external resources. To do this, you can do either of the following:

6.3.1 Configure Remote Control through fcta.conf

To configure the Remote Control credentials through the file, you must modify the corresponding configuration properties:

- FC.TransferAgent.config.remote.control.username
- FC.TransferAgent.config.remote.control.password

Once these properties have been set to the credentials that you want, restart the File*Catalyst* TransferAgent, and they will be applied.

6.3.2 Configure Remote Control through the command-line

To configure the Remote Control credentials through the command line, you must invoke the following command in the File*Catalyst* TransferAgent's install directory:

- Command: jre/bin/java -jar FileCatalystTransferAgent.jar -configureRemoteControl

6.4 Constructing RESTAuthorization header for future REST calls

Now that the Remote Control credentials are set, you must construct a value for the "RESTAuthorization" header that will be used in remote REST requests. This header is a Base64

encoding of the username and password (delimited with a ":") that you defined within the File*Catalyst* TransferAgent's configuration. An example of this encoding is as follows:

- Sample Remote Control username: admin
- Sample Remote Control password: system
- Sample non-encoded RESTAuthorization value: admin:system
- Sample encoded RESTAuthorization value: YWRtaW46c3lzdGVt

6.5 Evoking a remote REST call against the File*Catalyst* TransferAgent

With all of the preconditions established you can perform remote REST calls against the File*Catalyst* TransferAgent application. To do this, you construct the REST call that you wish to perform and then you add the "RESTAuthorization" header to the request.

- RESTAuthorization: YWRtaW46c3lzdGVt
- HTTP Method: **GET**
    - REST URL: https://localhost.filecatalyst.net:12680/rs/agent/heartbeat

7 Performing REST calls through Central REST Proxy

To perform File*Catalyst* TransferAgent REST calls through Central's REST proxy, there are a couple of steps that you must perform. They are as follows:

7.1 Connecting the File*Catalyst* TransferAgent application to Central

To perform REST calls against a File*Catalyst* TransferAgent through Central you must first connect the File*Catalyst* TransferAgent to a File*Catalyst* Central instance. This can be accomplished by specifying the connection information of your File*Catalyst* Central instance through the File*Catalyst* TransferAgent configuration UI, or by invoking the following command in the File*Catalyst* TransferAgent's install directory:

- CLI command: jre/bin/java -jar FileCatalystTransferAgent.jar -configureCentral

7.2 Determining the RESTAuthorization Header for Central

To determine the RESTAuthorization header that will be needed to authenticate your REST calls against File*Catalyst* Central, you must first gather the credentials for a File*Catalyst* Central user account that has access to your File*Catalyst* TransferAgent node. Once the user credentials are collected you can construct the RESTAuthorization header in the same fashion as the one generated in the Remote Control section.

7.3 Finding the File*Catalyst* TransferAgent's node ID

With the RESTAuthorization now known, perform a GET request to Central's /rs/nodes REST resource. This will return the currently connected nodes within File*Catalyst* Central.

7.3.1 REST Example:

- HTTP Method: GET
  - Example Central REST URL: http://127.0.0.1:8080/rs/nodes
- File*Catalyst* Central REST API Documentation Link: Link

7.4 Invoking the File*Catalyst* TransferAgent REST calls through Central

Once your File*Catalyst* TransferAgent node has been found in the current node list, you will be able to see the list of available REST services available to that node. All of these REST services use a different URL prefix then the normal File*Catalyst* TransferAgent REST services, but they function identically to the methods outlined in File*Catalyst* TransferAgent's system. If you need help with any of the REST services listed, open the File*Catalyst* TransferAgent's REST documentation and view the documentation for the service you wish to access.

8 Enabling external file-system support for the File*Catalyst* TransferAgent

To enable external filesystem (EFS) access within File*Catalyst* TransferAgent, you need to extract the "fc_EFS_drivers.zip" distributable into the File*Catalyst* TransferAgent install directory. Extracting this file will enable support of the currently supported external filesystems, and will allow you to supply EFS paths for all resources that would have normally supported file paths from the File*Catalyst* TransferAgent's local file system.

To provide an EFS path to the File*Catalyst* TransferAgent's REST services, you must construct a fully quantified URL that can be used to access the path on the EFS.

8.1.1 Example: External Filesystem URL construction (Using Samba filesystem)

- EFS Scheme: SMB
- EFS Username: ExampleUser
- EFS Password: ExamplePass
- EFS Host: 192.168.1.68
- Example Samba Sharename: TestSambaShare
- EFS File URL Syntax: ${efs.scheme}://${efs.username}:${efs.password}@${efs.host}/

Resulting Example Samba File URL:
smb://ExampleUser:ExamplePass@192.168.1.68/TestSambaShare

9 Support

For support contact information, support hours and live chat: visit our support website at https://support.filecatalyst.com