TIBCO® Managed File Transfer Command Center

API Guide

Version 8.7.0 | October 2025

Contents

JSON API Introduction

TIBCO® Managed File Transfer Internet Serverprovides a mechanism for navigating through the directory tree associated with a transfer user. TIBCO MFT Internet Server supports FT (File Transfer) Rest calls, but TIBCO MFT Command Center does not.

After you log in to TIBCO Managed File Transfer (MFT) Internet Server, a program can navigate through the TIBCO MFT Internet Serverdirectory structure using `jsondirtree.jsp`. Parameters are passed to the server in the URL and the data returned is formatted into JSON. Only one level of files and directories is returned for each call.

The jsondirtree API is an HTTP servlet call that is used to perform the following tasks:

- List files and folders in a particular path.
- Create a directory.
- Delete a directory.
- Delete a file.
- Rename a file or directory.

Note: To use the jsondirtree API to process file structure, you must first log in to the MFT Server and be assigned **TransferRight**. The MFT Server uses basic authentication. Only authorized transfer requests for the authenticated user are allowed by the jsondirtree call.

URL

The JSON API interface is exposed through a standard URL that contains the call methods and parameters provided by TIBCO MFT Internet Server.

To call a JSON API, you have to navigate to the following formatted URL with your browser:

```
https://your.server.com:port/context
/control?view=view/filetransfer/jsondirtree.jsp&action=action
&path=path&dest=dest&org.apache.catalina.filters.CSRF_NONCE=
19BEA97FB025306EED7023F14F620558
```

The following table lists the parameters in the URL:

Name	Description
`your.server.com`	IP name or address of the MFT server.
`port`	HTTPS port. If using 443, this parameter is not required.
`context`	Context of the application. Generally, cfcc is used.
`action`	Action requested. The supported actions are as follows: • `gettree`: returns an array of files or folders in the defined path. • `createdir`: creates a folder. • `deldir`: deletes an empty folder. • `defile`: deletes a file. • `rename`: renames a file or folder.
`path`	The alias path defined in the transfer definitions for the user. For example, `/VirtualAlias1` indicates viewing all files in Virtual Alias1.
`dest`	The target file or folder name. This parameter is only used on the rename action.

Name	Description
`org.apache.catalina.filters.CSRF_NONCE`	The CSRF nonce is used to protect against CSRF attacks. The first jsondirtree call must include no parameters, and it returns a CSRF_NONCE. The CSRF_NONCE must be included in the URL on the next jspondortree call. Each jsondirtree call returns a CSRF_NONCE that must be included on the URL of the next jsondirtree call.

Examples

The following table lists the parameters and their sample URLs:

Parameter	Sample URLs
`gettree`	https://mft.acme.com/cfcc/control? view=view/filetransfer/jsondirtree.jsp& action=gettree&path=/&org.apache.catalina.filters.CSRF_ NONCE=0BC0207D47896DBF5EEA627EEA716193 or https://mft.acme.com/cfcc/control? view=view/filetransfer/jsondirtree.jsp& action=gettree&path=/Dir1/FY2018&org.apache.catalina.filters.CSRF_ NONCE=0BC0207D47896DBF5EEA627EEA716193
`createdir`	https://mft.acme.com/cfcc/control? view=view/filetransfer/jsondirtree.jsp& action=createdir&path=/Dir1/FY2018&org.apache.catalina.filters.CSRF_ NONCE=0BC0207D47896DBF5EEA627EEA716193
`deldir`	https://mft.acme.com/cfcc/control? view=view/filetransfer/jsondirtree.jsp&

Parameter	Sample URLs
	action=deldir&path=/Dir1/FY2018&org.apache.catalina.filters.CSRF_NONCE=0BC0207D47896DBF5EEA627EEA716193
`delfile`	https://mft.acme.com/cfcc/control?view=view/filetransfer/jsondirtree.jsp&action=delfile&path=/Dir1/FY2018/acct.xls&org.apache.catalina.filters.CSRF_NONCE=0BC0207D47896DBF5EEA627EEA716193
`rename`	https://mft.acme.com/cfcc/control?view=view/filetransfer/jsondirtree.jsp&action=deldir&path=/Dir1/FY2018/file1.txt&dest=/Dir1/FY2013/acct.file1.txt&org.apache.catalina.filters.CSRFNONCE=0BC0207D47896DBF5EEA627EEA716193

Note: Only the file or directory name of the `dest` parameter is used. The parent path is ignored.

JSON Data Returned

Data is returned from the jsondirtree request in JSON format.

The following list shows the returned data:

- Information about the call
- CSRF_NONCE that should be included in the URL of the next call. For more information about CSRF_NONCE checking, see the "CSRF_NONCE Validation" section.
- An array of data describing files and folders returned

To see how this works, you can log in to TIBCO MFT Internet Server and issue the jsondirtree command shown earlier. The browser displays the JSON data returned. You can navigate through the directory tree by updating the `path` parameter.

Note: Additional fields might be returned by JSON and fields that are not documented must be ignored.

CSRF_NONCE Validation

MFT has enhanced the security of the JSONDIRTREE by including a check for a CSRF nonce. The check protects jsondirtree requests against CSRF attacks by including a CSRF token (referred to as CSRF_NONCE) on each request.

Here is how this works: The first jsondirtree request must be run without any parameters. This request does not require a CSRF_NONCE and returns a CSRF_NONCE that must be used in the next jsondirtree request.

For example:

https://localhost:8443/cfcc/control?view=view/filetransfer/jsondirtree.jsp

This returns an output similar to the following response:

```
{"vmgrname":"null","deldir":"N","errmsg":"\"Unknown action:
null\"","changemode":"","delfile":"N","readfid":"","token":"-
8923610393526332683","rc":1,"path":"\/","writefid":"","rename":"N","

action":null,"create":"N","csrftoken":"
&org.apache.catalina.filters.CSRF_
NONCE=0BC0207D47896DBF5EEA627EEA716193
","currentmode":"","currentcrlf":""}
```

Note: The CSRF_NONCE parameter is highlighted in bold. This parameter is the CSRF_NONCE that must be used on the next jsondirtree request.

https://localhost:8443/cfcc/control?view=view/filetransfer/jsondirtree.jsp&action=gettree&path=/&org.apache.catalina.filters.CSRF_NONCE=0BC0207D47896DBF5EEA627EEA716193

Each jsondirtree request returns a CSRF_NONCE. The CSRF_NONCE must be included in the URL of the next jsondirtree call in the `&org.apache.catalina.filters.CSRF_NONCE=nonceReturnedOnLastCall` format.

gettree

You can use this method to return an array of files or folders in the defined path.

Sample URLs

https://mft.acme.com/cfcc/control?view=view/filetransfer/jsondirtree.jsp&action=gettree&path=/&org.apache.catalina.filters.CSRF_NONCE=0BC0207D47896DBF5EEA627EEA716193

or

https://mft.acme.com/cfcc/control?view=view/filetransfer/jsondirtree.jsp&action=gettree&path=/Dir1/FY2018&org.apache.catalina.filters.CSRF_NONCE=0BC0207D47896DBF5EEA627EEA716193

Fixed Parameters Returned

The following table lists the data returned:

Parameter	Description
"token"	The token to use for all file transfer requests. It is changed on each request.
"rc"	The valid values are as follows: • 0: success. • 1: failure.
"errmsg"	Message.
"path"	The path value sent to the server. Forward slash (/) means the root level.
"action"	The action requested.
"delfile"	Defines whether the transfer definition allows a delete file action. The valid values are as follows: • "Y": yes. • "N": no.

Parameter	Description
"deldir"	Defines whether the transfer definition allows a delete directory action. The valid values are as follows: • "Y": yes. • "N": no.
"rename"	Defines whether the transfer definition allows a rename action. The valid values are as follows: • "Y": yes. • "N": no.
"create"	Defines whether the transfer definition allows a create directory action. The valid values are as follows: • "Y": yes. • "N": no.
"readfid"	The download transfer ID associated with this file or folder. This parameter is used when uploading or downloading files.
"writefid"	The upload transfer ID associated with this file or folder. This parameter is used when uploading or downloading files.
"vmgrname"	This parameter is for internal use only and can be ignored.
"changemode"	This parameter is for internal use only and can be ignored.
"currentcrlf"	This parameter is for internal use only and can be ignored.
"currentmode"	This parameter is for internal use only and can be ignored.

Array of Children Returned

The following table lists the array of children returned:

Parameter	Description
"entries"	A JSON array for files and folders in the parent directory. There is one entry in the array for each file or folder in the path.
"isdir"	Defines whether the entry is a directory. The valid values are as follows: • "Y": yes. • "N": this is a file.
"size"	The file size when *isdir* is not equal to "Y" (`isdir != "Y"`).
"modified"	The date of last modification: seconds since 1 January 1970.
"name"	The file or folder name.
"readrestart"	Defines whether download restart is supported. The valid values are as follows: • "Y": yes. • "N": no.
"writerestart"	Defines whether upload restart is supported. The valid values are as follows: • "Y": yes. • "N": no.
"delfile"	The parameter echoes information from the root entry.
"deldir"	The parameter echoes information from the root entry.
"rename"	The parameter echoes information from the root entry.
"create"	The parameter echoes information from the root entry.

Parameter	Description
"readfid"	The parameter echoes information from the root entry.
"writefid"	The parameter echoes information from the root entry.
"readcompress"	The parameter is for internal use only and can be ignored.
"writecompress"	The parameter is for internal use only and can be ignored.
"remainingpath"	The parameter is for internal use only and can be ignored.
"changemode"	The parameter is for internal use only and can be ignored.
"currentmode"	The parameter is for internal use only and can be ignored.
"currentcrlf"	The parameter is for internal use only and can be ignored.

createdir

You can use this method to create a directory in the defined path.

Sample URL

https://mft.acme.com/cfcc/control?view=view/filetransfer/jsondirtree.jsp&action=createdir&path=/Dir1/FY2018&org.apache.catalina.filters.CSRF_NONCE=0BC0207D47896DBF5EEA627EEA716193

Fixed Parameters Returned

The following table lists the parameters returned:

Parameters	Description
"rc"	The valid values are as follows:

Parameters	Description
	• `0`: success • `1`: failure
`"errmsg"`	Message
`"path"`	The path value sent to the server.
`"action"`	The action requested.

deldir

You can use this method to delete a directory.

Sample URL

https://mft.acme.com/cfcc/control?view=view/filetransfer/jsondirtree.jsp&action=deldir&path=/Dir1/FY2018&org.apache.catalina.filters.CSRF_NONCE=0BC0207D47896DBF5EEA627EEA716193

Fixed Parameters Returned

The following table lists the parameters returned:

Parameters	Description
`"rc"`	The valid values are as follows: • `0`: success. • `1`: failure.
`"errmsg"`	Message.
`"path"`	The path value sent to the server.

Parameters	Description
"action"	The action requested.

delfile

You can use this method to delete a file.

Sample URL

https://mft.acme.com/cfcc/control?view=view/filetransfer/jsondirtree.jsp&action=delfile&path=/Dir1/FY2018/acct.xls&org.apache.catalina.filters.CSRF_NONCE=0BC0207D47896DBF5EEA627EEA716193

Fixed Parameters Returned

The following table lists the parameters returned:

Parameters	Description
"rc"	The valid values are as follows: • 0: success • 1: failure
"errmsg"	Message
"path"	The path value sent to the server.
"action"	The action requested.

rename

You can use this method to rename a file or directory.

Sample URL

https://mft.acme.com/cfcc/control?view=view/filetransfer/jsondirtree.jsp&action=deldir&path=/Dir1/FY2018/file1.txt&dest=/Dir1/FY2013/acct.file1.txt&org.apache.catalina.filters.CSRF_NONCE=0BC0207D47896DBF5EEA627EEA716193

Note: Only the file or directory name that is part of the "`dest`" parameter, is used. The parent path is ignored.

Fixed Parameters Returned

The following table lists the parameters returned:

Parameters	Description
"`rc`"	The valid values are as follows: • `0`: success. • `1`: failure.
"`errmsg`"	Message.
"`path`"	The path value sent to the server.
"`action`"	The action requested.

MFT REST Calls

MFT supports REST calls for the most common administrative functions.

API Specification Files And Schema Used For REST Calls

The REST OpenAPI specification files in *MFT-Install*`/server/webapps/cfcc/public/docs` are as follows:

- *admin.json* - The JSON schema is used for all Admin REST calls supported by Command Center, and Internet Server when the Admin service is enabled.
- *admincc.json* - The JSON schema is used for Command Center REST calls. Internet Server does not support these REST calls.
- *ft.json* - The JSON schema is used for Internet Server File Transfer Rest calls. Command Center does not support these REST calls.

Note: These OpenAPI specification files can be imported in Postman, TIBCO ActiveMatrix BusinessWorks™ 6.4, and Swagger UI. For information on how to configure and run the REST calls, see *TIBCO ActiveMatrix BusinessWorks™ REST Reference*.

Before you use the JSON schemas, update the following line in the JSON files:

```
host : "localhost: 8443"
```

`localhost` must be changed to the host name of where the calls are run.

`8443` must be changed to the HTTP or HTTPS port of where the calls are run.

Redoc Used for API Documentation

MFT uses Redoc for API documentation.

Documentation for Command Center and Internet Server Administrative (Admin) REST calls can be accessed by clicking the Redoc link, which is at the upper-right corner of each Internet Server or Command Center admin page.

HTTP Methods Used

MFT uses the following HTTP methods.

Method	Description
GET	Used to retrieve a record or to search for multiple records.
POST	Used to create an object.
PUT	Used to update an object.
DELETE	Used to delete an object.

There are some calls that do not fit into these categories. For example, PUT is used to stop, start, return status of Transfer Servers.

Authentication

The basic authentication header should be included in the HTTP parameters. The user ID defined must have authorization to perform the desired function on the resource defined. MFT also supports certificate authentication for REST calls. To do this, you must configure the MFT Server HTTPS connector to request certificates.

Note:

- SAML and OIDC authentication is not supported for REST calls.
- When OIDC or SAML is configured, REST users must be configured in an exclusion list so that the users can be used in an HTTPs request without using SAML or OIDC.

REST HTTP Return Codes

The following table lists the REST HTTP return codes:

Return Code	Description
200	Successful
201	Successful Create
400	Invalid Parameter
401	Invalid Credentials
403	Forbidden
404	Not Found
409	Request Conflict This is also a catch-all when an error does not belong to another category. A message is generally returned with a description of the error.
500	Internal server Error - Unable to process the request

Resources Supported by REST

The resources supported by REST are provided in the following sections:

- Command Center and Internet Server Administrative (Admin) REST Calls
- Command Center Only Calls
- Internet Transfer REST Calls

Command Center and Internet Server Administrative (Admin) REST Calls

The Admin REST calls only work on Command Center or Internet Server when `admin-service-enabled` is true in the `web.xml` file. For information on rights to run an Admin

REST call, see *TIBCO® Managed File Transfer Command Center User Guide* or *TIBCO® Managed File Transfer Internet Server User Guide*.

MFT-Install/server/webapps/cfcc/public/docs/admin.json

REST URL: The URLs for the Admin REST calls are as follows:

- Command Center - `https://your.company.com:8443/cfcc/rest/admin/v7/resource/{parameters}`
- Internet Server - `https://your.company.com:7443/cfcc/rest/admin/v7/resource/{parameters}`

where,

Component	Description
https://your.company.com:8443	Defines the protocol, host name, and port
cfcc	Defines the MFT default context
rest	Defines that this is a REST call
admin	Defines that this is an admin REST call
v7	Defines the REST call version
resource	Defines the object of the REST call (like users and servers)
parameters	Defines the URL files required for the REST call

Note that the REST version is used to insulate users from changes in the REST calls. The following REST call versions are supported:

- v2: For MFT 8.2.x
- v3: for MFT 8.3.x
- v4, v4.1, v4.2: for MFT 8.4.x
- v5: for MFT 8.5.x
- v6: for MFT 8.6.x

- v7: for MFT 8.7.x

New MFT releases maintain support for older versions of REST calls.

> **Note:** While the version corresponds with the MFT release number, this may not be the case in future releases.

The following table lists the resource, the HTTP method, the description, and the URL that starts with the resource for some of the major Admin REST calls. For the complete list and their usage, click Redoc on the upper-right corner of the page.

Resource	HTTP Method	Description	URL Resource and Parameters
departments	POST	Adds departments	`departments`
	GET	Searches departments	`departments`
	GET	Gets a department	`departments/{deptName}`
	DELETE	Deletes a department	`departments/{deptName}`
	PUT	Updates a department	`departments/{deptName}`
errorEvents	GET	Searches for error events	`/errorevents?initiatingUser=<string>&serverHostName=<string>&sourceProtocol=<string>&destinationServer=<string>&destinationProtocol=<string>&errorType=<string>&searchStartDate=<string>&searchStartTime=<string>&searchEndDate=<string>&searchEndTime=`

Resource	HTTP Method	Description	URL Resource and Parameters
			`<string>&numberOfDays=<string>`
	GET	Retrieves error event details	`/errorevents/{errorEventId}`
groups	POST	Adds groups	`groups`
	GET	Searches groups	`groups?userId=<string>`
	GET	Gets a group	`groups/{groupId}`
	DELETE	Deletes a group	`groups/{groupId}`
	PUT	Updates a group	`groups/{groupId}`
	PUT	Adds a user to a group / removes a user from a group	`groups/{groupId}/{userId}`
	GET	Retrieves user IDs assigned to a group	`groups/{groupId}/users`
isaudits	GET	Searches Internet Server audit records	`/isaudits?auditId=<string>&localTransactionId=<string>& clientFileName=<string>&serverFileName=<string>& serverName=<string>&processName=`

Resource	HTTP Method	Description	URL Resource and Parameters
			<string>& userData= <string>&transferUserId= <string>& transferStatus= <string>&department= <string>& fromDateAndTime= <string>&toDateAndTime= <string>& noOfDays= <string>&recordsReturned= <string>&pageNumber= <string>& as2MDNStatusSuccess= <string>& as2MDNStatusFailure= <string>& as2MDNStatusPending= <string>& transferId= <string>&virtualAlias= <string>
	GET	Gets Internet Server audit record	isaudits/{auditId}
	DELETE	Deletes audits by date	isaudits/date
	DELETE	Deletes audits by number of days	isaudits/days
ldap	PUT	Synchronizes a user or an	ldap

Resource	HTTP Method	Description	URL Resource and Parameters
		authenticator	
lockout release	PUT	Releases an invalid login lock (s)	`lockoutrelease`
mftversion	GET	Gets the version	`mft/version`
pgp publickeys	POST	Adds PGP public keys	`pgppublickeys`
	GET	Searches PGP public keys	`pgppublickeys?userServerFlag=<string>&status=<string>&name=<string>&keyId=<string>&lastKeyId=<string>`
	GET	Gets a PGP public key	`pgppublickeys/{keyId}`
	PUT	Updates a PGP public key	`pgppublickeys/{keyId}`
	DELETE	Deletes a PGP public key	`pgppublickeys/{keyId}`
pgp systemkeys	POST	Adds PGP system key	`pgpsystemkeys`

Resource	HTTP Method	Description	URL Resource and Parameters
	PUT	Imports a PGP system key	`pgpsystemkeys`
	PUT	Updates a PGP system key	`pgpsystemkeys/{keyId}`
	GET	Searches all the PGP system keys	`pgpsystemkeys`
	GET	Gets details of a PGP system key	`pgpsystemkeys/{keyId}`
	DELETE	Removes a PGP system key	`pgpsystemkeys/{keyId}`
protocol publickeys	POST	Adds a public key	`protocolpublickeys`
	GET	Searches public keys	`protocolpublickeys?keyType=<string>&userServerFlag=<string>& keyStatus=<string>&userOrServerName=<string>& lastKeyId=<string>`
	PUT	Updates a public key	`protocolpublickeys/{keyType}/{keyId}`
	DELETE	Deletes a public key	`protocolpublickeys/{keyType}/{keyId}`
	GET	Gets a public key	`protocolpublickeys/`

Resource	HTTP Method	Description	URL Resource and Parameters
			`{UserServerFlag}/{keyType}/{keyId}`
	PUT	Retrieves public keys for a server	`protocolpublickeys/server`
protocol systemkeys	POST	Adds a protocol system key	`protocolsystemkeys`
	PUT	Imports a protocol system key	`protocolsystemkeys`
	GET	Exports a protocol system key	`protocolsystemkeys/:keyId?serverFileName=<string>&keyType=<string>`
	PUT	Updates a protocol system key	`protocolsystemkeys/{keyType}/{keyId}`
	GET	Searches all the protocol system keys	`protocolsystemkeys?keyType=<string>&description=<string>& lastKeyId=<string>`
	GET	Gets the details of a protocol system key	`protocolsystemkeys/{keyType}/{keyId}`
	DELETE	Removes a protocol system key	`protocolsystemkeys/{keyType}/{keyId}`

Resource	HTTP Method	Description	URL Resource and Parameters
roles	GET	Gets all roles. You must have `AdministratorRight`, `UpdateTransferUserRight` to run this call.	`roles`
	PUT	Adds a user to a role / removes a user from a role.	`roles/{userId}`
	GET	Gets roles for a user	`roles/{userId}`
	GET	Gets a role. You must have `AdministratorRight`, `UpdateTransferUserRight` to run this call.	`roles/id/{roleId}`
	GET	Gets all users in a role. You must have `AdministratorRight`, `UpdateTransferUserRight` to run this call.	`roles/users/id/{roleId}`
server credentials	POST	Creates server credentials	`servercredentials`
	PUT	Updates server credentials	servercredentials
	DELETE	Removes server credentials	`servercredentials/{userOrGroupType}/`

Resource	HTTP Method	Description	URL Resource and Parameters
		from a user	`{userOrGroupId}/{serverName}`
	GET	Gets server credentials for a user	`servercredentials/{userOrGroupType}/{userOrGroupId}/{serverName}`
	GET	Searches server credentials	`servercredentials?userOrGroupType=<string>&userOrGroupId=<string>&serverName=<string>&remoteUserId=<string>&remoteUserWindowsDomain=<string>`
servers	POST	Adds servers	`servers`
	GET	Searches servers	`servers?serverName=<string>&ipName=<string>&serverType=<string>&serverPlatform=<string>&department=<string>`
	PUT	Updates a server	`servers/{servername}`
	DELETE	Deletes a server	`servers/{servername}`
	GET	Gets a server	`servers/{servername}`

Resource	HTTP Method	Description	URL Resource and Parameters

tags	GET	Searches tags	`tags?`
			`name=<string>&`
			`value=<string>&`
			`resources=<array of strings>&`
			`resourceid=<string>`

	POST	Add tags	`tags`

	PUT	Updates a tag	`tags?`
			`resource=<string>&`
			`resourceid=<string>`

	DELETE	Deletes a tag	`tags?`
			`resource=<string>&`
			`resourceid=<string>`

transfers	POST	Adds transfers	`transfers`

	GET	Searches transfers	`transfers?`
			`transferId=<string>&`
			`serverFileName=<string>&`
			`authUserId=<string>&`
			`authGroupId=<string>&`
			`serverName=<string>&`
			`department=<string>&`

Resource	HTTP Method	Description	URL Resource and Parameters

			`description=< string>&`
			`virtualAlias=<string>&`
			`expiredFlag=<string>&`
			`processName=<string>&`
			`userData=<string>&`
			`lastFileId=<string>`

	GET	Lists all transfer definitions of a user	`users/{userId}/transfers`
	GET	Gets a transfer	`transfers/{transferId}`
	PUT	Updates a transfer	`transfers/{transferId}`
	DELETE	Deletes a transfer	`transfers/{transferId}`
	DELETE	Deletes expired transfers. You must have `AdministratorRight`, `UpdateTransferDefinitionRight` to run this call.	`transfers/expired`
transferservers	PUT	Provides the status, starts, or stops the transfer server. **Note:** The transferservers REST call has been deprecated. It has been replaced by the transferserverstatus REST call.	`transferservers/{hostname}`

Resource	HTTP Method	Description	URL Resource and Parameters
transfer serversconfig	GET	Retrieves the AS2 transfer server configuration	`/tranferserverconfig/as2/{hostName}`
	PUT	Updates the AS2 transfer server configuration	`/tranferserverconfig/as2/{hostName}`
	GET	Retrieves FTP transfer server configuration	`/tranferserverconfig/ftp/{hostName}`
	PUT	Updates FTP transfer server configuration	`/tranferserverconfig/ftp/{hostName}`
	GET	Retrieves OFTP2 transfer server configuration	`/tranferserverconfig/oftp2/{hostName}`
	PUT	Updates OFTP2 transfer server configuration	`/tranferserverconfig/oftp2/{hostName}`
	GET	Retrieves Platform Server configuration	`/tranferserverconfig/ps/{hostName}`
	PUT	Updates Platform Server configuration	`/tranferserverconfig/ps/{hostName}`
	GET	Retrieves SSH transfer server configuration.	`/tranferserverconfig/ssh/{hostName}`
	PUT	Updates SSH transfer server configuration	`/tranferserverconfig/ssh/{hostName}`
transfer	PUT	Provides the status and starts	`/transferserverstatus/`

Resource	HTTP Method	Description	URL Resource and Parameters
serverstatus		or stops the transfer server	{hostname}
users	POST	Adds users	users
	GET	Gets a user	users/{userId}
	PUT	Updates a user	users/{userId}
	DELETE	Deletes a user	users/{userId}
	GET	Searches users	users? userId=<string>& FullName=<string>& AssignedRight=<string>& Group=<string>& EmailAddress=<string>& UserType=<string>& Department=<string>& fromExpirationDate=<string>& toExpirationDate=<string>& DisabledUsers=<string>& LockedUsers=<string>& canChangeOwnPassword=<string>& PasswordNeverExpires=<string>&

Resource	HTTP Method	Description	URL Resource and Parameters
			`PasswordExpired=<string>&` `lastUserId=<string>`
	PUT	Changes Password	`/user/password`

Important Guidelines for REST Calls

- **Public Key / PGP Public Key REST Calls:** Enter key data with \r\n characters. Example:

```
-----BEGIN CERTIFICATE-----\r\n

MIIBqzCCARSgAwIBAgIGAV8GPboIMA0GCSqGSIb3DQEBBQUAMA8xDTALBgNVBAMMBHR
lc3QwHhcN\r\n

MTcxMDEwMTIyMzQ5WhcNMjIxMDA5MTgzMDAwWjAPMQ0wCwYDVQQDDAR0ZXN0MIGfMA0
GCSqGSIb3\r\n

DQEBAQUAA4GNADCBiQKBgQCD8/xhSXzHHD8D0rRaALRE1l7PwjPKTD3vb1LWSGNMTd1
s/FiC/Yeb\r\n

LD4iZ2dpywhwojqtm4+OeowX/EhhluQQ628o/HV2l9JqpHqt75tNiVhed4YhApzGfpo
TAJGAO5q8\r\n

mtN4nhhsyUZRA4tciB5qSyDUQWy6Fd9yAabm+S0B6wIDAQABoxIwEDAOBgNVHQ8BAf8
EBAMCAvQw\r\n

DQYJKoZIhvcNAQEFBQADgYEAVRf7uTZ8MwmA0oc32BcToWj+xQwf+D3pJZ+XrDP/NDM
RcXwimJ0N\r\n

+zcQO7ZqgZ+Hy/PVEKSWbOBEwAZPtNuvJhLnkZtE9rq+TxfDR4wn1zWPKmJKLLuCSdx
NK535xI4n\r\n
F84cED1MC0ZFh5wZdB3VFh8mXchZw9RnPM/kWP2hZls=\r\n
-----END CERTIFICATE-----
```

  In TIBCO ActiveMatrix BusinessWorks™, `\r\n` characters must be replaced with `&crlf;` character.

- **Search REST Calls:** The wildcard character used for search requests is an asterisk (*).

- **Search User and Transfer REST Calls:** The search user REST calls return only 100 records at a time. To retrieve the next 100 records, you must set the `lastUserId` parameter to the last user id value that was returned in the previous call. Similarly, for the search transfers rest calls, you must set the `lastFileId` parameter to the last transfer id value that was returned in the previous call.
- **Update Server REST Calls:** You can reset to the default value for the following fields as shown.

Fields	Default Value	Reset Request
FTPKeepAliveInterval	The default value is 0	"ftpKeepAliveInterval": ""
ftpPoolingIdleTimeout	The default value is 5	"ftpPoolingIdleTimeout": ""
sshPoolingIdleTimeout	The default value is 5	"sshPoolingIdleTimeout": ""

Example:

1. If you want to update the field values to 20, 60, and 60 respectively, enter the request as follows:

   "ftpKeepAliveInterval": "20"

   "ftpPoolingIdleTimeout": "60"

   "sshPoolingIdleTimeout": "60"

2. If you want to reset the fields to their default values, enter the request as follows:

   "ftpKeepAliveInterval": "",

   "ftpPoolingIdleTimeout": "",

   "sshPoolingIdleTimeout": ""

   Note: The response for the above entries is set to default values only if the value is blank and enclosed between double quotation marks ("").

Note:
If you want to reset the values of PPA using the update transfer definition, then you must pass "postActionType1": "NONE" in the update request.

```
{
 "transferId": "F204K0001807",
 "postActionType1": "NONE",
 "postActionType2": "NONE",
 "postActionType3": "NONE",
 "postActionType4": "NONE"
}
```

Command Center Only REST Calls

`MFTCC-Install/server/webapps/cfcc/public/docs/admincc.json`

The Command Center REST calls only work on a Command Center. For information on rights to run a Command Center REST call, see the *TIBCO Managed File Transfer Command Center User Guide*.

The URLs for the Command Center REST calls are as follows:

```
https://your.company.com:8443/cfcc/rest/admincc/v7/resource/{parameters}
```

The following table lists the resource, the HTTP method, the description, and the URL that starts with the resource for some of the major Command Center REST calls. For the complete list and their usage, click Redoc on the upper-right corner of the page.

Resource	HTTP Method	Description	URL Resource and Parameters
alerts	GET	Retrieves login event alert details.	/alerts/logon/{alertId}

Resource	HTTP Method	Description	URL Resource and Parameters
	PUT	Updates an existing login event alert definition.	/alerts/logon/{alertId}
	POST	Adds a login event alert.	/alerts/logon
	GET	Retrieves transfer event alert details.	/alerts/transferEvent/{alertId}
	PUT	Updates an existing transfer event alert definition.	/alerts/transferEvent/{alertId}
	POST	Adds a transfer event alert.	/alerts/transferEvent
	GET	Retrieves transfer non-event alert details.	/alerts/transferNonEvent/{alertId}
	PUT	Updates an existing transfer non-event alert definition.	/alerts/transferNonEvent/{alertId}
	POST	Adds a transfer non-event alert.	/alerts/transferNonEvent
	GET	Lists all alerts.	/alerts?transferEvent= <string>&logonEvent= <string>&transferNonEvent=

Resource	HTTP Method	Description	URL Resource and Parameters
			<string>&description= <string>&severity= <string>&enabled= <string>&userids= <string>&department= <string>&transferType= <string>&transferStatus= <string>&serverName= <string>&processName= <string>&transferDescription= <string>&lastAlertId=<string>
	DEL	Deletes an alert.	/alerts/{alertId}
dni	POST	Adds a template to a Platform Server.	dni/{psServerName}/{templateName}
	PUT	Gets status, start, stop, or update a template of a Platform Server.	dni/{psServerName}/{templateName}
	GET	Lists DNI daemons.	dni
	GET	Lists templates of a Platform Server.	dni/{psServerName}

Resource	HTTP Method	Description	URL Resource and Parameters
	GET	Gets a template of a Platform Server.	dni/{psServerName}/{templateName}
	DELETE	Deletes a template from a Platform Server.	dni/{psServerName}/{templateName}
events	GET	Searches for events	/events?eventid= <string>&requestType= <string>&eventType= <string>&userId= <string>&department= <string>&eventStatus= <string>&jobName= <string>&fromDate= <string>&fromTime= <string>&toDate= <string>&toTime= <string>&numberOfDays= <string>&lastEventId=<string>
	GET	Retrieves event details	/events/{eventId}
psaudits	DELETE	Removes PS	psaudits/days

Resource	HTTP Method	Description	URL Resource and Parameters
		audits older than days.	
	DELETE	Removes PS audits older than date.	psaudits/date
	GET	Gets a Platform Server Audit using Audit ID.	psaudits/{auditId}
	GET	Searches Platform Server audits.	psaudits? auditId=<string>& localTransactionId=<string>& clientFileName=<string>& serverFileName=<string>& serverName=<string>& processName=<string>& userData=<string>& transferUserId=<string>& transferStatus=<string>& department=<string>& fromDateAndTime=<string>& toDateAndTime=<string>& noOfDays=<string>& pageNumber=<string>& recordsReturned=<string>& nodeName=<string>
	GET	Searches Platform Server audits on Platform Server.	psaudits/psserver? psServerName=<string>& nodeName=<string>& localTransactionId=<string>& localFileName=<string> &transferUserId=<string>& transferStatus=<string>& fromDate=<string>&

Resource	HTTP Method	Description	URL Resource and Parameters
			fromTime=<string>& toDate=<string>& toTime=<string>& noOfDays=<string>& recordsReturned=<string>
	GET	Gets Platform Server audit record from Platform Server.	psaudits/ {localTransactionId}/psserver/ {psServerName}
psnodes	POST	Adds a new Platform Server node to the bank.	psnodes
	PUT	Updates a Platform Server node in the bank.	psnodes/{nodeName}
	PUT	Adds or updates node on Platform Server from the bank.	psnodes/{nodeName}/psserver/ {psServerName}
	PUT	Adds or updates node on Platform Server.	psnodes/psserver/ {psServerName}
	GET	Gets all Platform Server nodes from the bank.	psnodes

Resource	HTTP Method	Description	URL Resource and Parameters
	GET	Gets node from Platform Server.	psnodes/{nodeName}/psserver/{psServerName}
	GET	Gets all nodes from Platform Server.	psnodes/psserver/{servername}
	GET	Gets node from the bank.	psnodes/{nodeName}
	DELETE	Deletes node From the bank.	psnodes/{nodeName}
	DELETE	Delete node from Platform Server.	psnodes/{nodeName}/psserver/{psServername}
psuserprofiles	POST	Adds user profile to the banks.	psuserprofiles
	PUT	Adds/Updates a Platform Server user profile in the server.	psuserprofiles/psserver/{psServerName}
	PUT	Updates a user profile in the bank.	psuserprofiles/{profileId}
	PUT	Updates a user profile in Platform Server from the bank.	psuserprofiles/{profileId}/psserver/{psServerName}

Resource	HTTP Method	Description	URL Resource and Parameters
	GET	Gets all Platform Server user profiles from the bank.	psuserprofiles
	GET	Gets all Platform Server user profiles from the server.	psuserprofiles/psserver/{psServerName}
	GET	Gets a user Profile from the bank using Profile ID.	psuserprofiles/{profileId}
	DELETE	Deletes a user profile from Platform Server.	psuserprofiles/psserver/{psServerName}
	DELETE	Deletes a user profile from the bank.	psuserprofiles/{profileId}
psresponder profiles	POST	Adds a new Responder Profile to the bank.	psresponderprofiles
	PUT	Adds or updates a Platform Server Responder Profile in the server.	psresponderprofiles/psserver/{psServerName}

Resource	HTTP Method	Description	URL Resource and Parameters
	PUT	Updates a Platform Server Responder Profile in the bank.	psresponderprofiles/{profileId}
	PUT	Adds or updates a Platform Server Responder Profile in Server from the bank.	psresponderprofiles/{profileId}/psserver/{psServerName}
	GET	Gets all Platform Server Responder Profiles from the server.	psresponderprofiles/psserver/{psServerName}
	GET	Gets a Platform Server Responder Profile from the bank using Profile ID.	psresponderprofiles/{profileId}
	GET	Gets all Platform Server Responder Profiles from the bank.	psresponderprofiles
	DELETE	Deletes a Responder Profile from the bank.	psresponderprofiles/{profileId}

Resource	HTTP Method	Description	URL Resource and Parameters
	DELETE	Deletes a Responder Profile from Platform Server.	psresponderprofiles/psserver/{psServerName}
platformserver transfers	POST	Adds a Platform Server transfer.	pstransfers
	PUT	Updates a Platform Server transfer.	pstransfers/{transferId}
	PUT	Executes a Platform Server transfer from bank using Transfer ID.	pstransfers/{transferId}/execute
	PUT	Executes a Platform Server transfer.	pstransfers/execute
	GET	Searches for Platform Server transfers.	pstransfers? description=<string>& serverName=<string>& initiatorFileName=<string>& responderFileName=<string>& responderHostName=<string>& transferId=<string>& transferType=<string>
	GET	Gets a Platform Server transfer from the bank using transfer	pstransfers/{transferId}

Resource	HTTP Method	Description	URL Resource and Parameters
		ID.	
	DELETE	Deletes a Platform Server transfer from bank using transfer ID.	pstransfers/{transferId}
services	PUT	Gets status, start, or stop a service.	services
	PUT	Gets status, start, or stop a service or scheduler on a host, or hold a scheduler.	services/{hostname}
activepstransfers	GET	Retrieves active Platform Transfers for a server.	activepstransfers/{serverName}
	GET	Retrieves details of a Platform Server active transfer.	activepstransfers/{serverName} / {transactionId}?wait=<string> &timeout=<string>
	PUT	Cancels an active Platform Server transfer.	activepstransfers/{serverName}/ {transactionId}

Resource	HTTP Method	Description	URL Resource and Parameters
activeistransfers	GET	Retrieves active Internet transfers.	activeistransfers
	GET	Retrieves active Internet transfers for a server.	activeistransfers/{serverName}
	GET	Retrieves details of an active transfer.	activeistransfers/details/{auditId}?hostName=<string>&wait=<string>&timeout=<string>
hoststats	GET	Retrieves host status.	hoststats
istransfer	PUT	Performs Internet Server file transfer.	istransfer/file
	PUT	Performs Internet Server Alias to Alias Transfer.	istransfer/virtualalias
	PUT	Performs Internet Server JMS transfer.	istransfer/jms

Resource	HTTP Method	Description	URL Resource and Parameters
jobs	POST	Adds a job.	jobs
	PUT	Updates a job.	jobs/jobName/{jobName}/groupName/{groupName}
	PUT	Executes a job now.	jobs/jobName/{jobName}/groupName/{groupName}/execute
	GET	Gets a job using the job name and group name.	jobs/jobName/{jobName}/groupName/{groupName}
	GET	Retrieves all active jobs.	jobs/activejobs?hostName=<string>
	GET	Searches jobs or gets all jobs.	jobs?jobName=<string>&groupName=<string>&description=<string>&jobType=<string>&enabled=<string>&batchName=<string> &department=<string>&authorizedUserId=<string>&authorizedGroupId=<string>&jobScheduledDate=<string>
	DELETE	Deletes a job using job name and group name.	jobs/jobName/{jobName}/groupName/{groupName}

Resource	HTTP Method	Description	URL Resource and Parameters
serverstatus	GET	Retrieves server status.	serverstatus
	GET	Retrieves server status details.	serverstatus/{serverName}
statistics	GET	Searches statistics by date range.	statistics? startDate={startDate}&endDate={endDate}

Internet Transfer REST Calls

`MFTIS-Install/server/webapps/cfcc/public/docs/ft.json`

The Internet Transfer REST calls only work on an Internet Server. For information on rights to execute Internet Transfer REST calls, see *TIBCO® Managed File Transfer Internet Server User Guide*. The URL for the Internet Transfer REST calls is as follows:

```
https://your.company.com:7443/cfcc/rest/ft/v7/resource/{parameters}
```

The following table lists the resource, the HTTP method, the description, and the URL that starts with the resource for some of the major Internet Transfer REST calls. For the complete list and their usage, click Redoc on the upper-right corner of the page.

Resource	HTTP Method	Description	URL Resource and Parameters
navigation	POST	Creates a directory.	navigation/{filePath:.*}
	PUT	Renames a file	navigation/{fromPath:.*}

Resource	HTTP Method	Description	URL Resource and Parameters
		or directory.	
	GET	Gets one level directory tree.	navigation/{filePath:.*}
	DELETE	Deletes a directory or file.	navigation/{filePath:.*}
transfer	POST	Uploads file in streaming mode.	transfer/{filePath:.*}
	PUT	Uploads file in streaming mode.	transfer/{filePath:.*}
	GET	Downloads file in streaming mode.	transfer/{filePath:.*}
bwtransfer	POST	Uploads file in rest mode.	bwtransfer/{filePath:.*}
	PUT	Uploads file in rest mode.	bwtransfer/{filePath:.*}
	GET	Downloads file in rest mode.	bwtransfer/{filePath:.*}
ftsession	PUT	Validates	ftsession

Resource	HTTP Method	Description	URL Resource and Parameters
		session.	
	GET	Gets session.	ftsession
	DELETE	Closes session.	ftsession
mft	GET	Gets version.	mft/version
ftuser	PUT	Changes password.	ftuser/password
	GET	Gets password expiring days.	ftuser/passwordexpiringdays
	GET	Gets transfers.	ftuser/transfers
	GET	Gets audits.	ftuser/audits
virtualaliastransfer	PUT	Performs Internet Server Alias to Alias Transfer	/istransfer/virtualalias

MFT File Transfer REST Calls

MFT supports uploading, downloading and moving files from one server to another using a REST call to an Internet Server.

The following two distinct modes for uploading or downloading a file are available:

- **Streaming**: This is designed for larger files and necessitates multiple REST calls for the complete file upload/download.
- **Non-streaming**: This involves transmitting the complete file data within a single request (upload) or response (download) body.

The Internet Server Virtual Alias to Virtual Alias REST call moves files from one server to another.

Upload a File in Streaming Mode

REST URL

`/rest/ft/v7/transfer/{filePath}`

`{filePath}` is the virtual alias of the transfer definition and the file name.

HTTP Method

POST or PUT

URL Parameters

Parameter	Description	Value
position	Upload starting position	Position for this chunk of data; long value >= 0. No default value.

packet	Packet sequence	Sequence for this chunk of data; integer value >= 1, incremented by one for each packet. No default value.
final	Is this the last packet in this upload	`true` or `false` No default value.
encode	How is the data encoded	base64, hex, or do not include parameter • Base64-the server decodes the data as base64. • Hex - the server interprets the data as hex (2 hex digits equal one byte). Default - data is not decoded.

Return values

```
Success:
HTTP 202 for partial upload, 200 for the final package with json message
containing audit id

Errors:
HTTP 4xx with a message in the json body
HTTP 500 with a generic message
```

Examples:

1. Uploading base64 data:
    a. First Request:

```
POST
/cfcc/rest/ft/v7/transfer/temp/myfile.txt?position=0&packet=1&
final=false&encode=base64 HTTP/1.1
Host: localhost:7443
Content-Type: application/octet-stream
Authorization: Basic Z2VvcmdlbDpwcm9naW5ldA==
Content-Length: 8

SGVsbG8=
```

```
Server Response
202
Set-Cookie JSESSIONID=5F1C4407DE51B105204D5F1FCA61AA68
{"message":"Audit id: A812P0001B69"}
```

b. Second request:

```
POST
/cfcc/rest/ft/v7/transfer/temp/myfile.txt?position=5&packet=2&
final=true&encode=base64 HTTP/1.1
Host: localhost:7443
Content-Type: application/octet-stream
Authorization: Basic Z2VvcmdlbDpwcm9naW5ldA==
Cookie: JSESSIONID=5F1C4407DE51B105204D5F1FCA61AA68
Content-Length: 8

IFdvcmxk
```

```
Server Response
200
{"message":"Audit id: A812P0001B69"}
```

```
Contents of File myfile.txt
Hello World
```

2. Uploading hex data:

a. First Request:

```
POST
/cfcc/rest/ft/v7/transfer/temp/myfile.txt?position=0&packet=1&
final=false&encode=hex HTTP/1.1
Host: localhost:7443
Content-Type: application/octet-stream
Authorization: Basic Z2VvcmdlbDpwcm9naW5ldA==
Content-Length: 10

48656c6c6f
```

Server Response

```
202
Set-Cookie JSESSIONID=678F60540574667A7DEB1B338C7F43E1
{"message":"Audit id: A812P0001B70"}
```

b. Second request:

```
POST
/cfcc/rest/ft/v7/transfer/temp/myfile.txt?position=5&packet=2&
final=true&encode=hex HTTP/1.1
Host: localhost:7443
Content-Type: application/octet-stream
Authorization: Basic Z2VvcmdlbDpwcm9naW5ldA==
Cookie: JSESSIONID=678F60540574667A7DEB1B338C7F43E1
Content-Length: 12

20576f726c64
```

Server Response

```
200
{"message":"Audit id: A812P0001B70"}
```

Contents of File myfile.txt

Hello World

3. Uploading unencoded data

   a. First Request

```
POST
/cfcc/rest/ft/v7/transfer/temp/myfile.txt?position=0&packet=1&
final=false HTTP/1.1
Host: localhost:7443
Content-Type: application/octet-stream
Authorization: Basic Z2VvcmdlbDpwcm9naW5ldA==
Content-Length: 5

Hello
```

Server Response

```
202
Set-Cookie JSESSIONID=803C7E248C991835EB4D3E110221D998
{"message":"Audit id: A812P0001B71"}
```

   b. Second Request

```
POST
/cfcc/rest/ft/v7/transfer/temp/myfile.txt?position=5&packet=2&
final=true HTTP/1.1
Host: localhost:7443
Content-Type: application/octet-stream
Authorization: Basic Z2VvcmdlbDpwcm9naW5ldA==
Cookie: JSESSIONID=803C7E248C991835EB4D3E110221D998
```

```
Content-Length: 6

World
```

Server Response
200
{"message":"Audit id: A812P0001B71"}

Contents of File myfile.txt
Hello World

Best Practices

To upload a file, data is sent through a series of REST calls. The entire file can be sent in a single request or be broken down into multiple chunks, each sent in a separate request.

- **Session Management**: A `JSessionId` HTTP cookie must be included in every request. This ensures that all chunks are associated with the same file upload session.
- **Content-Type**: The content-type header for all requests must be set to application/octet-stream.
- **Request Body**: The file data (or chunk of data) must be placed directly in the request body. The data should not be formatted as JSON.

Query Parameters

Each upload request must include the following query parameters as needed.

Parameter	Type	Description	Required
`packet`	Integer	A 1-based sequential identifier for each data chunk. The first packet is `1`, the second is `2`, and so on.	Yes
`position`	Integer	A 0-based byte offset for the current chunk. For a chunk size of 500 bytes, the first packet's position is `0`, the second is `500`, and so forth.	Yes
`encode`	String	Specifies the encoding method used on the data (example: `base64`, `hex`). This parameter should be omitted if the data is not encoded.	No

Data Encoding

The requirement for data encoding depends on the file type.

- Text Data: No encoding is necessary. The `encode` parameter should not be included in the request.
- Binary Data: Data must be encoded before being sent.
    - Base64: This is the recommended encoding method.
    - Hex: Alternatively, data can be hex-encoded, where each character is converted to its ASCII hexadecimal equivalent (example: the character 'a' becomes `61`).

Download a File in Streaming Mode

REST URL

`/rest/ft/v7/transfer/{filePath}`

`{filePath}` would be the virtual alias of the transfer definition and the file name.

HTTP Method

`GET`

URL Parameters

Parameter	Description	Values
`position`	Download starting position	Position for this chuck of data long value >= 0. No default value.
`encode`	How is the data encoded	base64, hex, or do not include parameter • Base64 - the server decodes the data as base64. • Hex - the server interprets the data as hex (2 hex digits equal one byte). Default - data is not decoded.

`zipdownload`	Zip a directory before downloading	true or false

Return Values

```
Success:
HTTP 200 and body contains file data
```

```
Errors:
HTTP 4xx with a message in the json body
HTTP 500 with a generic message
```

Examples

- Downloading base64 data

  Request:

  ```
  GET
  /cfcc/rest/ft/v7/transfer/temp/localfile.txt?position=0&encode=base
  64 HTTP/1.1
  Host: localhost:7443
  Authorization: Basic Z2VvcmdlbDpwcm9naW5ldA==
  ```

  ```
  Server Response
  200
  Body
  YWJjZGVmZ2hpamtsbW5vcHFyc3R1dnd4eXo=
  ```

  ```
  Decodes to
  abcdefghijklmnopqrstuvwxyz
  ```

- Downloading hex data

  Request:

  ```
  GET
  /cfcc/rest/ft/v7/transfer/temp/localfile.txt?position=0&encode=hex
  HTTP/1.1
  Host: localhost:7443
  Authorization: Basic Z2VvcmdlbDpwcm9naW5ldA==
  ```

  ```
  Server Response
  200
  Body
  616263646566768696a6b6c6d6e6f707172737475767778797a
  ```

```
Decodes to
abcdefghijklmnopqrstuvwxyz
```

- Downloading unencoded data

  Request:

  ```
  GET /cfcc/rest/ft/v7/transfer/temp/localfile.txt?position=0
  HTTP/1.1
  Host: localhost:7443
  Authorization: Basic Z2VvcmdlbDpwcm9naW5ldA==
  ```

  ```
  Server Response
  200
  Body
  abcdefghijklmnopqrstuvwxyz
  ```

- Downloading zip file

  Request:

  ```
  GET /cfcc/rest/ft/v7/transfer/temp/demo?position=0&zipdownload=true
  HTTP/1.1
  Host: localhost:7443
  Authorization: Basic Z2VvcmdlbDpwcm9naW5ldA==
  ```

  ```
  Server Response
  200
  Body
  Binary data that should be saved a zip file
  ```

  In this example, temp is the virtual alias name and demo is a directory found in that virtual alias.

Standard Download

In a standard file download, the user can specify the starting point for the data retrieval using a `position` parameter. However, it is not possible to specify the size of the data chunk returned by the server.

Zipped Directory Download

This mode is enabled by using the `zipdownload` parameter and has the following specific constraints:

- **Position**: The position parameter must be set to 0.
- **Encoding**: Any encode parameter in the request are ignored.
- **Content**: The resulting download is a single zip file containing all files and subdirectories located at the target path.

The primary functional difference between the download modes is that the zipped download mode provides the capability to retrieve an entire directory structure as a single archive.

Upload a File in Non-Streaming Mode

REST URL

`/rest/ft/v7/bwtransfer/{filePath}`

`{filePath}` is the virtual alias of the transfer definition and the file name.

HTTP Method

POST or PUT

Body

```
JSON format, data is base64 encoded

{
“base64Data”: “Hello World”

}
```

Return values

```
Success:
HTTP 200 for partial upload, 200 for the final package with json message
containing audit id
```

```
Errors:
HTTP 4xx with a message in the json body
HTTP 500 with a generic message
```

Examples:

Request:

```
POST /cfcc/rest/ft/v7/bwtransfer/temp/myfile.txt HTTP/1.1
Host: localhost:7443
Content-Type: application/json
Authorization: Basic Z2VvcmdlbDpwcm9naW5ldA==
Content-Length: 42

{
"base64Data": "SGVsbG8gV29ybGQ="
}
```

```
Server Response
200
{"message":"Audit id: A813P00003F9"}
```

```
Contents of File myfile.txt
Hello World
```

> **Note:** This file upload mode facilitates the transmission of data via a single REST call. The request body is to be formatted in JSON, and the file data must be Base64 encoded. The content-type header should be set to `application/json`.

Download a File in Non-Streaming Mode

REST URL

`/rest/ft/v7/bwtransfer/{filePath}`

`{filePath}` would be the virtual alias of the transfer definition and the file name.

HTTP Method

`GET`

Return values

```
Success:
HTTP 200 with json body containing file data
```

```
Errors:
HTTP 4xx with a message in the json body
HTTP 500 with a generic message
```

Example:

Request:

```
GET /cfcc/rest/ft/v7/bwtransfer/temp/localfile.txt HTTP/1.1
Host: localhost:7443
Authorization: Basic Z2VvcmdlbDpwcm9naW5ldA==
```

```
Server Response
200
{
"base64Data": "YWJjZGVmZ2hpamtsbW5vcHFyc3R1dnd4eXo="
}
```

Note: This file download mode facilitates the transmission of data via a single REST call. The body of the response is in JSON format. The data is returned in base64 encoded.

Internet Server Alias to Alias Transfer

REST URL

/rest/ft/v7/istransfer/virtualalias

HTTP Method

PUT

Body

JSON format

Parameter	Type	Description
isHostName	string	Internet Server host name
sourceVirtualAlias	string	Virtual alias name for a download transfer definition
sourceFileName	string	File name (supports regex)
targetVirtualAlias	string	Virtual alias name for an upload transfer definition
targetFileName	string	File name (supports regex)
dataType	string	Text or Binary
messageDelimiter	string	None, CRLF, LF
waitForCompletion	string	Yes or No
waitTimeout	string	Integer from 1 - 1400, which represents the number of minutes
fourEyes	boolean	true or false

Return values

```
Success:
HTTP 200 with json message containing audit id

Errors:
HTTP 4xx with a message in the json body
HTTP 500 with a generic message
```

Example

Request:

```
PUT /cfcc/rest/ft/v7/istransfer/virtualalias HTTP/1.1
Host: localhost:7443
Content-Type: application/json
Authorization: Basic Z2VvcmdlbDpwcm9naW5ldA==
Content-Length: 278

{
"isHostName": "MyIS",
"sourceVirtualAlias": "gcp-ssh",
"sourceFileName": "*",
"targetVirtualAlias": "temp",
"targetFileName": "*",
"dataType": "Binary",
"messageDelimiter": "None",
"waitForCompletion": "Yes",
"waitTimeout": "30"
}
```

```
Server Response
200
{
"message": "SUCCESS",
"id": "A814P0002772"
}
```

Virtual Alias to Virtual Alias Transfer: File Name Patterns

Use this REST call to move data between servers.

To use this call, enter the following information:

- For source, specify the name of the virtual alias from a download transfer definition. The source file name supports file name, directories, subdirectories and regular expressions.
- For target, specify the name of the virtual alias from an upload transfer definition. The target file name can be * for directory transfers (the target file name matches the source) or a specific file name for single file transfers.
- Specify the file names.

The following table describes the patterns to use to distinguish expressions.

Pattern	Description

*	Specifies all files in the directory.
**	Specifies all files in the directory and its subdirectories.
[]	Specifies a regular expressions. Regular expressions are added to the end of the remote file name.

Examples: Source File Name and Target File Name

- Use a regex to download files only with a certain extension

```
"sourceFileName": "/* [^.*\\.(?i)(doc|docx|pdf|txt|xls|xlsx|jpg)$]"
“targetFileName”: “*”
```

- All files in a directory:

```
"sourceFileName": "/aaa/*"`
“targetFileName”: “*”`
```

- All files in a directory and all subdirectories:

```
"sourceFileName": "/aaa/**"
“targetFileName”: “*”
```

- Rename a single file:

```
"sourceFileName": "/aaa/bbb.txt"
“targetFileName”: “ccc.txt”
```

Creating Customer File Transfer Interface

Through the use of a well-defined interface, you can write your own module in Java to support a protocol not currently supported by TIBCO MFT. Only outbound transfers are supported.

Note: The documentation of the Java interface and example code are found in the *MFT Install*/server/webapps/cfcc/example/customTransfers directory on the server. The javadoc folder is found in the docs directory.

Procedure

1. Write your implementation, compile the code, and package it into a .jar file.
2. Copy the .jar file and any dependent jar files into the following directory on each of the Internet Servers: *MFT Install*/server/webapps/cfcc/WEB-INF/lib
3. Copy the mft-custom-transfer.jar file from *MFT Install*/server/webapps/cfcc/example/customTransfers to *MFT Install*/server/webapps/cfcc/WEB-INF/lib on each of the Internet Servers.
4. Restart the MFT Servers.
5. Add a server that is defined as a "custom server".
6. Enter the class name that implements the com.tibco.mft.transfers.custom.CustomTransfer interface.
7. Add any extra configuration you may need in a text-based format.

   Here is the sample build script from the examples directory:

```
rmdir /S/Q build
mkdir build
javac -classpath mft-custom-transfer.jar;json-simple-1.1.1.jar -d
build src\com\example\transfer\CustomXferImpl.java
cd build
jar -cvf MyCustomXfer.jar com\example\transfer\CustomXferImpl.class
move MyCustomXfer.jar ..\
cd ..
```

```
rmdir /S/Q build
```

Warnings for Using Custom Transfers

Here are some things to keep in mind when you use custom transfers.

- You are responsible for all changes to support this code.
- You are responsible for verifying that the third-party code is up to date and has no security vulnerabilities.
- You must note that the third-party software supplied by TIBCO MFT Command Center is subject to change.
- You are responsible for implementing and testing the custom transfer change correctly.
- If you are the MFT Admin, you must limit access to the server where the software is located.
- You must install the custom transfer software on each Internet Server that can execute the custom transfer.

TIBCO Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the Product Documentation website, mainly in HTML and PDF formats.

The Product Documentation website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The documentation for this product is available on the TIBCO® Managed File Transfer Command Center Documentation page.

How to Contact Support for TIBCO Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our product Support website.
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the product Support website. If you do not have a username, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature

requests from within the TIBCO Ideas Portal. For a free registration, go to TIBCO Community.

Legal and Third-Party Notices

SOME CLOUD SOFTWARE GROUP, INC. (“CLOUD SG”) SOFTWARE AND CLOUD SERVICES EMBED, BUNDLE, OR OTHERWISE INCLUDE OTHER SOFTWARE, INCLUDING OTHER CLOUD SG SOFTWARE (COLLECTIVELY, “INCLUDED SOFTWARE”). USE OF INCLUDED SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED CLOUD SG SOFTWARE AND/OR CLOUD SERVICES. THE INCLUDED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER CLOUD SG SOFTWARE AND/OR CLOUD SERVICES OR FOR ANY OTHER PURPOSE.

USE OF CLOUD SG SOFTWARE AND CLOUD SERVICES IS SUBJECT TO THE TERMS AND CONDITIONS OF AN AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER AGREEMENT WHICH IS DISPLAYED WHEN ACCESSING, DOWNLOADING, OR INSTALLING THE SOFTWARE OR CLOUD SERVICES (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH LICENSE AGREEMENT OR CLICKWRAP END USER AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE SAME TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

TIBCO, the TIBCO logo, the TIBCO O logo, and Slingshot are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only. You acknowledge that all rights to these third party marks are the exclusive property of their respective owners. Please refer to Cloud SG’s Third Party Trademark Notices (https://www.cloud.com/legal) for more information.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: https://scripts.sil.org/OFL

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.