



TIBCO® Managed File Transfer Internet Server

API Guide

*Version 8.5.0
March 2023*



Contents

Contents	2
JSON API Introduction	4
URL	4
Examples	6
JSON Data Returned	7
CSRF_NONCE Validation	8
gettree	8
createdir	12
deldir	13
delfile	14
rename	14
MFT REST Calls	16
HTTP Methods Used	17
Authentication	17
REST HTTP Return Codes	17
Resources Supported by REST	18
Command Center and Internet Server Administrative (Admin) REST Calls	18
Command Center Only REST Calls	33
Internet Transfer REST Calls	46
Performing File Transfers Using HTTP Protocol	48
File Transfers Using Java Applet	52
Required Concepts	52
Using the Java Wrapper	52
Creating Customer File Transfer Interface	54

Warnings for Using Custom Transfers	55
TIBCO Documentation and Support Services	56
Legal and Third-Party Notices	58

JSON API Introduction

TIBCO® Managed File Transfer Internet Server provides 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 on to TIBCO Managed File Transfer (MFT) Internet Server, a program can navigate through the TIBCO MFT Internet Server directory structure using `jsondirtree.jsp`. Parameters are passed to the server in the URL and 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.

i Note: To use the `jsondirtree` API to process file structure, you must first log on 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 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 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 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.
org.apache.catalina.filters.CSRF_NONCE	The CSRF nonce is used to protect against

Name	Description
	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	<code>https://mft.acme.com/cfcc/control?view=view/filetransfer/jsondirtree.jsp&action=gettree&path=/&org.apache.catalina.filters.CSRF_NONCE=0BC0207D47896DBF5EEA627EEA716193</code> or <code>https://mft.acme.com/cfcc/control?view=view/filetransfer/jsondirtree.jsp&action=gettree&path=/Dir1/FY2018&org.apache.catalina.filters.CSRF_NONCE=0BC0207D47896DBF5EEA627EEA716193</code>
createdir	<code>https://mft.acme.com/cfcc/control?view=view/filetransfer/jsondirtree.jsp&action=createdir&path=/Dir1/FY2018&org.apache.catalina.filters.CSRF_NONCE=0BC0207D47896DBF5EEA627EEA716193</code>
deldir	<code>https://mft.acme.com/cfcc/control?view=view/filetransfer/jsondirtree.jsp&action=deldir&path=/Dir1/FY2018&org.apache.catalina.filters.CSRF_</code>

Parameter	Sample URLs
	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

i 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 onto 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.

i 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 executed 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": "" }
```

i 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.
"defile"	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 <i>isdir</i> is not equal to "Y" (<i>isdir</i> != "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 located 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 execute 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 the where the calls are executed.

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

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 top-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 execute an Admin

REST call, see *TIBCO® Managed File Transfer Command Center User Guide* or *TIBCO® Managed File Transfer Internet Server User's 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/v5/resource/{parameters}>
- Internet Server - <https://your.company.com:7443/cfcc/rest/admin/v5/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.
v5	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

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

Note that while the version corresponds with the MFT release number, this may not be the case in future releases.

For Admin REST calls, the following table lists the resource, the HTTP method, the description, and the URL that starts with the resource.

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= <string>&numberOfDays=<string>

Resource	HTTP Method	Description	URL Resource and Parameters
	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 authenticator.	ldap

Resource	HTTP Method	Description	URL Resource and Parameters
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}

Resource	HTTP Method	Description	URL Resource and Parameters
	GET	Gets a public key.	protocolpublickeys/ {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}

Resource	HTTP Method	Description	URL Resource and Parameters
	DELETE	Removes a protocol system key.	protocolsystemkeys/{keyType}/{keyId}
roles	GET	Gets all roles. You must have AdministratorRight, UpdateTransferUserRight to execute 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 execute this call.	roles/id/{roleId}
	GET	Gets all users in a role. You must have AdministratorRight, UpdateTransferUserRight to execute this call.	roles/users/id/{roleId}
server credentials	POST	Creates server credentials.	servercredentials

Resource	HTTP Method	Description	URL Resource and Parameters
	PUT	Updates server credentials.	servercredentials
	DELETE	Removes server credentials from a user.	servercredentials/ {userOrGroupType}/ {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>&remoteUserWindowsDo main= <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}

Resource	HTTP Method	Description	URL Resource and Parameters
	DELETE	Deletes a server.	servers/{servername}
	GET	Gets a server.	servers/{servername}
transfers	POST	Adds transfers.	transfers
	GET	Searches transfers.	transfers? transferId=<string>& serverFileName=<string>& authUserId=<string>& authGroupId=<string>& serverName=<string>& department=<string>& 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}

Resource	HTTP Method	Description	URL Resource and Parameters
	DELETE	Deletes expired transfers. You must have AdministratorRight, UpdateTransferDefinitionRight to execute 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}
transfer serversconfig	GET	Retrieves the AS2 transfer server configuration.	/transferserverconfig/as2/{hostName}
	PUT	Updates the AS2 transfer server configuration.	/transferserverconfig/as2/{hostName}
	GET	Retrieves FTP transfer server configuration.	/transferserverconfig/ftp/{hostName}
	PUT	Updates FTP transfer server configuration.	/transferserverconfig/ftp/{hostName}
	GET	Retrieves OFTP2 transfer server configuration.	/transferserverconfig/oftp2/{hostName}

Resource	HTTP Method	Description	URL Resource and Parameters
	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 serverstatus	PUT	Provides the status and starts or stops the transfer server.	/transferserverstatus/{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>&

Resource	HTTP Method	Description	URL Resource and Parameters
			AssignedRight=<string>& Group=<string>& EmailAddress=<string>& UserType=<string>& Department=<string>& fromExpirationDate=<string>& toExpirationDate=<string>& DisabledUsers=<string>& LockedUsers=<string>& canChangeOwnPassword=<string>& PasswordNeverExpires=<string>& PasswordExpired=<string>& lastUserId=<string>
	PUT	Changes Password	/user/password

Important Guidelines for REST Calls

- **Public Key / PGP Public Key REST Calls:** You must 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+0eowX/EhhluQQ628o/HV2l9JqpHqt75tNiVhed4YhApzGfpo
TAJGA05q8\r\n

mtN4nhhsyUZRA4tcIB5qSyDUQWY6Fd9yAabm+S0B6wIDAQABoxIwEDA0BgNVHQ8BAf8
EBAMCAvQw\r\n

DQYJKoZIhvcNAQEFBQADgYEA VRf7uTZ8MwMA0oc32BcToWj+xQwf+D3pJZ+XrDP/NDM
RcXwimJ0N\r\n

+zcQ07ZqgZ+Hy/PVEKSWb0BEwAZPtNuvJhLnkZtE9rq+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 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 `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 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 execute a Command Center REST call, see *TIBCO Managed File Transfer Command Center User's Guide*.

The URLs for the Admin REST calls are as follows:

```
https://your.company.com:8443/cfcc/rest/admincc/v5/resource/{parameters}
```

For Command Center REST calls, the following table lists the resource, the HTTP method, the description, and the URL that starts with the resource.

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

Resource	HTTP Method	Description	URL Resource and Parameters
		details.	
	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= <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	dni/{psServerName}/

Resource	HTTP Method	Description	URL Resource and Parameters
		to a Platform Server.	{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}
	GET	Gets a template of a Platform Server.	dni/{psServerName}/{templateName}
	DELETE	Deletes a template from a Platform Server.	dni/{psServerName}/{templateName}
psaudits	DELETE	Removes PS audits older than days.	psaudits/days
	DELETE	Removes PS audits older than date.	psaudits/date
	GET	Gets a Platform Server Audit using Audit ID.	psaudits/{auditId}

Resource	HTTP Method	Description	URL Resource and Parameters
	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>& fromTime=<string>& toDate=<string>& toTime=<string>& noOfDays=<string>& recordsReturned=<string>
	GET	Gets Platform Server audit record from Platform Server.	psaudits/ {localTransactionId}/psserver/ {psServerName}

Resource	HTTP Method	Description	URL Resource and Parameters
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
	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	psnodes/{nodeName}

Resource	HTTP Method	Description	URL Resource and Parameters
		From the bank.	
	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}
	GET	Gets all Platform Server user profiles from the bank.	psuserprofiles
	GET	Gets all Platform Server user profiles from server.	psuserprofiles/psserver/{psServerName}
	GET	Gets a user	psuserprofiles/{profileId}

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

Resource	HTTP Method	Description	URL Resource and Parameters
		from the bank.	
	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}
	DELETE	Deletes a Responder Profile from Platform Server.	psresponderprofiles/psserver/{psServerName}
platformserver transfers	POST	Adds a Platform Server transfer.	pstransfers
	PUT	Updates a	pstransfers/{transferId}

Resource	HTTP Method	Description	URL Resource and Parameters
		Platform Server transfer.	
	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 ID.	pstransfers/{transferId}
	DELETE	Deletes a Platform Server transfer from bank using transfer ID.	pstransfers/{transferId}
services	PUT	Gets status,	services

Resource	HTTP Method	Description	URL Resource and Parameters
		start, or stop a service.	
	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 an Platform Server active transfer.	activepstransfers/{serverName} / {transactionId}?wait=<string> &timeout=<string>
	PUT	Cancels an active Platform Server transfer.	activepstransfers/{serverName}/ {transactionId}
activeistransfers	GET	Retrieves active Internet transfers.	activeistransfers
	GET	Retrieves active Internet transfers for a server.	activeistransfers/{serverName}

Resource	HTTP Method	Description	URL Resource and Parameters
	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
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

Resource	HTTP Method	Description	URL Resource and Parameters
	GET	Gets a job using 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}
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's Guide*. The URL for the Internet Transfer REST calls is as follows:

```
https://your.company.com:8443/cfcc/rest/ft/v5/resource/{parameters}
```

For Internet Transfer REST calls, the following table lists the resource, the HTTP method, the description, and the URL that starts with the resource.

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

Resource	HTTP Method	Description	URL Resource and Parameters
bwtransfer	POST	Uploads file in rest mode.	bwtransfer/{filePath:. *}
	PUT	Uploads file in rest mode.	bwtransfer/{filePath:. *}
	GET	Download file in rest mode.	bwtransfer/{filePath:. *}
ftsession	PUT	Validates session.	ftsession
	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

Performing File Transfers Using HTTP Protocol

This section describes the process of performing an HTTP upload or download. This procedure should work in any programming language that has HTTP client classes. These steps are the same steps that the MFT Browser client uses. So you can use the browser client to test these steps. TIBCO recommends that initial tests use HTTP so you do not have to worry about any certificates. When using HTTPS, you may need to add the MFT server certificate to the Java cacerts file. Optionally, you might create a certificate manager to handle certificate exceptions.

Before you begin

TIBCO MFT Internet Server requires a NONCE for each jsontirtree request. The NONCE must be added to each request in the `org.apache.catalina.filters.CSRF_NONCE=085980EE44051B48CE865FF3999765F8` format.

Procedure

1. Authenticate to MFT Server.
 - a. Set the basic authentication header, and send request to:
`https://your.host:port/cfcc/login/servlet/login`. For example:
`https://10.1.2.3:443/cfcc/login/servlet/login`
The following HTTP codes indicate success: 200 302
 - b. Extract and save the JSESSIONID in the JSESSIONID cookie.



Note: The JSESSIONID cookie must be included in all requests documented below.

2. Get the MFT FTServiceSessionID cookie.
 - a. Use the JSESSIONID cookie in the HTTP session to go to:
`https://your.host:port/cfcc/control?view=view/filetransfer/browser/sart.jsp`. For example,

```
https://10.1.2.3.4:443/cfcc/control?view=view/filetransfer/browser/s
tart.jsp
```

The following HTTP codes indicate success: 200 302

- b. Extract and save the FTServiceSessionID in the FTServiceSessionID cookie.

i Note: The value of the cookie must be included in the SessionID parameter of the upload and download URL defined in Step 4.

3. Get the File ID and virtual alias of the file to be uploaded or downloaded.

The `jsondirtree.jsp` call has the following purposes:

- Extract the BToken used in the upload and download request.
- Extract the NONCE used by subsequent `jsondirtree` requests for TIBCO MFT Internet Server 8.1 and later.
- Navigate through the directory structure of a remote transfer definition.

- a. Issue a call to return the session token and the NONCE:

```
https://your.host:port/cfcc/control?view=view/filetransfer/jsondirtree.jsp. (Do not include any URL parameters on this request.)
```

This request returns two important fields in the JSON Response:

- `csrftoken: &org.apache.catalina.filters.CSRF_NONCE=B805F69A7A796585C5B64094C5AA5412`
- `token: -470748818111506828`

The "token" must be passed to the upload and download URL links defined in Step 4 using the URL parameter BToken. If a NONCE is returned, it must be included in each subsequent call to `jsondirtree` by appending the value to the URL. For example:

```
https://10.1.2.3:443/cfcc/control?view=view/filetransfer/jsondirtree.jsp&
amp;action=gettree&path=/&org.apache.catalina.filters.CSRF_
NONCE=085980EE44051B48CE865FF3999765F8.
```

You can hard-code the FileID, or you can navigate through the directory structure to return the FileID for a particular virtual alias.

i Note: Versions of TIBCO MFT Internet Server prior to 8.1 do not return a NONCE.

4. Perform the file transfer.

- a. Decide whether the transfer is an upload or download. Call the proper Java or Python HTTP classes to perform an HTTP upload or download and pass this class to one of the URLs as shown here.

- Download URL:

```
https://your.host:port/cfcc/control?view=servlet/fileTransfer&FileID=transferId&BrowserXfer=Y&BToken=token&AliasInPath=Y&ClientFileName=clientfilename&SessionID=MFTSessionId&ServerFileName=/serveralias/filename
```

For example:

```
https://10.1.2.3:443/cfcc/control?view=servlet/fileTransfer&FileID=FA0550000001&BrowserXfer=Y&BToken=8707202069055907714&AliasInPath=Y&ClientFileName=c:\temp\test.file&SessionID=a0401bcfa9264a68e0c5cda1ead7a0587f9d37c9&ServerFileName=/local/test.file
```



Note: You must fill in the following fields:

- FileID: from Step 3: jsondirtree call
- ClientFileName: Is the fully qualified file name where the file is to be saved
- BToken: from Step 3: jsondirtree call
- SessionID: from Step 2. This is the FTServiceSessionID.
- ServerFileName: Include the Virtual Alias and the file name to be downloaded. If the file is in a subdirectory, you must include the fully qualified path name. The download request can use method GET or POST. (An upload request has to be with method POST and the http header "Content-Type" has to be multipart/form-data

```
Content-Type:
multipart/form-data;
```

- Upload URL:

```
https://your.host:port/cfcc/control?view=servlet/fileTransfer&AliasInPath=Y&FileID=transferId&BToken=token&BrowserXfer=Y&ClientFileName=clientfilename&SessionID=MFTSessionId&ServerFileName=/serveralias/
```

For example:

```
https://10.1.2.3:443/cfcc/control?view=servlet/fileTransfer&AliasInPath=Y&FileID=FA0550000002&BToken=8707202069055907714&BrowserXfer=Y&ClientFileName=mywordfile.docx&SessionID=a0401bcfa9264a68e0c5cda1ead7a0587f9d37c9&ServerFileName=/local/
```



Note: You must fill in the following fields:

- FileID: from Step 3: jsondirtree call
- ClientFileName: Is the fully qualified file name where the file is to be saved
- BToken: from Step 3: jsondirtree call
- SessionID: from Step 2. This is the FTServiceSessionID.
- ServerFileName: Include the Virtual Alias and the file name to be downloaded. If the file is in a subdirectory, you must include the fully qualified path name. The upload request must be with method POST and the http header "Content-Type" must be multipart/form-data

Content-Type:

multipart/form-data;

Result

The following HTTP codes indicate success: 200



Note: The response body is a JSON object string.

File Transfers Using Java Applet

TIBCO MFT Internet Server includes a Java applet that can do file transfers using a proprietary HTTP protocol. To leverage this transfer API, you can write your own code to follow the MFT protocol, and upload and download files. Note that this API requires Java and is documented for compatibility purposes only.

New applications should use the FT REST interface to upload or download files. FT REST calls are simpler to execute and can be initiated from any language that supports REST. These calls can also be initiated by ActiveMatrix BusinessWorks™ or by applications such as Postman.

Required Concepts

You need to understand how MFT works in order to use this API. You need to create the necessary user, server and transfer definitions to upload or download files. You must also understand the differences between file and directory transfers. Hint: Configure the transfer definitions as directory transfers.

We suggest using the MFT browser transfer client to upload or download files to verify that the user, server, and transfer definitions have been made correctly. Once the browser client can perform the file transfer, the Java API should also work.

Using the Java Wrapper

For ease of use, TIBCO MFT Internet Server provides wrapper classes that hide the proprietary protocol details. A sample program is supplied with TIBCO MFT Internet Server that explains how this code works.

The sample program is called `APIExample.java` and is distributed in the following files. The source code for both Windows and UNIX is identical; you can use either file.

- UNIX: *MFT-Install*/distribution/InternetTransfer/InternetTransferClient.tar
- Windows: *MFT-*

Install/distribution/InternetTransfer/InternetTransferClient.zip

The `APIExample.java` file references all the wrapper classes and demonstrates how to use them. Follow the instructions in this file to compile and run your Java implementations.

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 Internet Server 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 [TIBCO Product Documentation](#) website, mainly in HTML and PDF formats.

The [TIBCO Product Documentation](#) website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The following documentation for TIBCO® Managed File Transfer Internet Server is available on the [TIBCO® Managed File Transfer Internet Server](#) Product Documentation page.

- TIBCO® Managed File Transfer Internet Server *Managed File Transfer Overview*
- TIBCO® Managed File Transfer Internet Server *Installation*
- TIBCO® Managed File Transfer Internet Server *Quick Start Guide*
- TIBCO® Managed File Transfer Internet Server *User Guide*
- TIBCO® Managed File Transfer Internet Server *Utilities Guide*
- TIBCO® Managed File Transfer Internet Server *API Guide*
- TIBCO® Managed File Transfer Internet Server *Transfer and File Share Clients User Guide*
- TIBCO® Managed File Transfer Internet Server *Desktop Client User Guide*
- TIBCO® Managed File Transfer Internet Server *Security Guide*
- TIBCO® Managed File Transfer Internet Server *Container Deployment*
- TIBCO® Managed File Transfer Internet Server *Release Notes*

How to Contact TIBCO Support

Get an overview of [TIBCO Support](#). You can contact TIBCO Support in the following ways:

- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the [TIBCO Support](#) website.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to [TIBCO Support](#) website. If you do not have a user name, 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 TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

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

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of Cloud Software Group, Inc.

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.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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.

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.

This 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 this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SOFTWARE GROUP, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

This and other products of Cloud Software Group, Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 2003-2023. Cloud Software Group, Inc. All Rights Reserved.