



TIBCO® Managed File Transfer Platform Server for UNIX User's Guide

Version 8.0.1

April 2022



Contents

Product Overview	6
TIBCO® Managed File Transfer Suite	6
TIBCO Managed File Transfer (MFT) Platform Server	6
MFT Platform Server Modes of Operation	7
Administration Commands	8
Starting TIBCO MFT Platform Server	8
Stopping TIBCO MFT Platform Server	8
Verifying TIBCO MFT Platform Server	9
Displaying Version Information of TIBCO MFT Platform Server	9
TLS/SSL Certificates Setup	10
Generating a Certificate Request	10
Parameters in the sslutility.exe Utility	12
Viewing a Certificate	13
Encrypting a Private Key Password	14
Configuring a Certificate	14
Transfer Commands	16
File to File Transfers	16
Examples: Transfer Using cfsend or cfrecv Command	17
Wildcards	17
Directory Transfers	17
Directory Transfer Parameters	18
Directory Tokens	18
Examples: Directory Transfers	19
File Name Tokens	20
Examples: Transfer Using File Name Tokens	25
Post Processing Actions (PPA)	25
PPA Command Format	25
PPA Substitutable Parameters	26
PPA Error Codes	27
File to Job Transfers	28
Running Remote Commands	28
Error Handling: Running Remote Commands	29
File to Print Transfers	29
Transfer Using Nodes	31
Creating Nodes	31
Deleting Nodes	34

Listing Nodes	34
Node Parameters	35
Examples: Transfer Using Nodes	39
Transfer Using Profiles	40
Local Profile	40
Creating Local Profiles	40
Listing Local Profiles	41
Deleting Local Profiles	41
Local Profile Parameters	41
Responder Profile	43
Creating Responder Profiles	43
Listing Responder Profiles	44
Deleting Responder Profiles	44
Responder Profile Parameters	44
Transfer Using Distribution Lists	46
Configuring Distribution Lists	46
Distribution List Parameters	46
Examples: Transfer Using Distribution Lists	47
Transfer Using Templates	48
Sample Templates: TSEND and TRECVC	49
Transfer Parameters	52
Minimum Transfer Parameters	52
Optional Transfer Parameters	53
z/OS Specific Transfer Parameters	62
TIBCO Accelerator Parameters	68
Configuration Parameters	70
Server Configuration	70
Server Configuration Parameters	71
Server SSL Communications Parameters	75
Client Configuration	78
Client Configuration Parameters	79
Client SSL Communications Parameters	82
Common Configuration	83
Common Configuration Parameters	84
Extended Features	91
Checkpoint Restart	92
Checkpoint Restart Example	92
Conversion Tables/Custom Code Conversion	93
ASCII to EBCDIC Conversion Table Example	95

Directory Named Initiation (DNI)	96
fusing Utility	96
fusutil Utility	97
Configured Post Processing	98
Argument Substitution	99
Configured Post Processing Command Examples	100
CfAlias	100
CfAlias Parameters	100
Substitutable Parameters	101
Examples: Using CfAlias	102
Auditing (cfinq Utility)	103
Log Files	103
cfinq Parameters	103
Examples: Using cfinq Utility	106
Access Control	108
Access Control Parameters	108
Examples: Access Control	112
Default Access Control Entries	113
Access Control Format	113
CRL Support	114
CRL Configuration	114
Configuring a Sample CRL	114
Configured SSL Authorization	115
Configured SSL Authorization Format	115
SSL Authorization Parameters	116
Examples: SSL Authorization	117
User Exits	118
CfXitData Structure	118
cfunix2dos.exe Utility	123
TIBCO Accelerator	123
TIBCO Accelerator Ports	123
Using TIBCO Accelerator within TIBCO MFT Platform Server for UNIX	124
Managing TIBCO Accelerator	124
Configuring the TIBCO Accelerator Server	125
Configuring TIBCO Accelerator Client	125
Example 1: Transfer from Linux to Linux through TIBCO Accelerator	126
Example 2: Transfer from z/OS to Windows through TIBCO Accelerator on Linux	127
Appendix: PAM Authentication	129
Configuring PAM Authentication	129

Configuring PAM Authentication Service	129
TIBCO Documentation and Support Services	131
Legal and Third-Party Notices	132

Product Overview

TIBCO® Managed File Transfer Platform Server for UNIX is an integration product to provide you secure and reliable file transfers based on your platform.

Managed File Transfer (MFT) refers to software or services that manage secure transfers of data from one computer to another through a network. MFT applications are designed for enterprises as an alternative to using other file transfer solutions, such as FTP, HTTP and others.

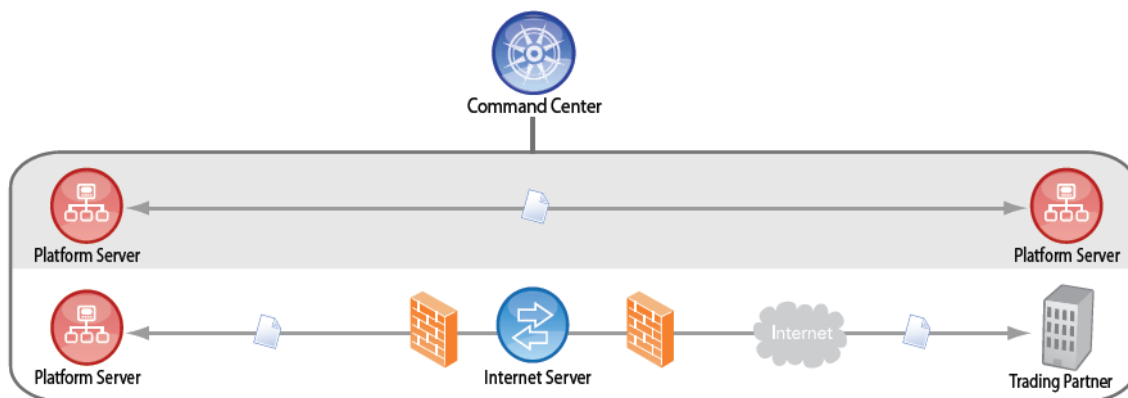
TIBCO® Managed File Transfer Suite

With TIBCO Managed File Transfer (MFT) Suite, you can exchange files quickly, securely, and economically across all major operating systems and geographical boundaries.

TIBCO MFT Suite provides a complete file transfer solution, which is composed of the following components:

- TIBCO® Managed File Transfer Platform Server enables an organization to secure and control activities to transfer files on all major platforms throughout an enterprise.
- TIBCO® Managed File Transfer Internet Server enables organizations to exchange information over the internet with complete security and control. This component is the portal through which all files are exchanged with external users.
- TIBCO® Managed File Transfer Command Center provides a single point of control to manage all of your enterprise file transfers, both inside and outside the enterprise, and across all major platforms.

The following figure shows how the components of TIBCO MFT Suite are connected together:



TIBCO Managed File Transfer (MFT) Platform Server

TIBCO MFT Platform Server enables an organization to control activities and to securely transfer files on all major platforms throughout the enterprise.

TIBCO MFT Platform Server is a robust solution for the following reasons:

- It supports all major platforms including Windows, UNIX (AIX and Solaris), Linux, z/Linux, iSeries, and IBM Mainframe.
- It offers a full range of encryption algorithms and built-in security and authentication options to secure any file transfer.
- It automates the process of file transfers and provides scheduling options and event-driven capabilities.

MFT Platform Server Modes of Operation

MFT Platform Server supports the following modes of operation for incoming and outgoing requests. This is for both file transfer requests and administrative requests, such as audit collection, server status as well as node and profile updates.

Mode of Operation	Description
Clear text mode	The password is encrypted using a proprietary encryption algorithm but the data is not encrypted.
AES 256 encryption	The password and data are encrypted using AES256. The asymmetric encryption key is generated through an algorithm on both the client and the server.
SSL (or TLS) mode	An asymmetric AES 256 encryption key is exchanged through a secure TLS connection after an SSL connection is established with the partner server. The AES 256 encryption key is used to encrypt and decrypt all data. A message digest and sequence number is added to each record to prevent man in the middle attacks.
Tunnel mode	<p>All data is sent over a negotiated TLS connection. Each transfer creates a new TLS connection.</p> <p>Tunnel mode is the most secure option, and it is strongly suggested when communicating to partners over the Internet. Tunnel mode requires MFT Internet Server V8.2 and MFT Platform Server V8.0 or higher.</p>

Administration Commands

You can use certain commands to administer TIBCO MFT Platform Server.

For more instructions, see the following tasks:

- [Starting TIBCO MFT Platform Server](#)
- [Stopping TIBCO MFT Platform Server](#)
- [Verifying TIBCO MFT Platform Server](#)
- [Displaying Version Information of TIBCO MFT Platform Server](#)

Starting TIBCO MFT Platform Server

You can start TIBCO MFT Platform Server by using the **cfstart** command.



The **cfstart** command must be used by the root user or the user that installed the platform server.

Each time after you use the **cfstart** command, a daemon (CyberResp) is started and a global server trace file is created. Multiple daemons can run at the same time. The trace file name is the *servername* with the current time as an extension. This file contains Process IDs (PIDs) of all started child processes.

Procedure

1. Optional: On the command line, navigate to the `$CFROOT/bin` directory.
2. Use the following commands to start TIBCO MFT Platform Server:
 - Enter `cfstart` if you want to start the platform server without an SSL certificate.
 - Enter `cfstart -ssl` if you want to start the platform server with an SSL certificate.
 - Enter `cfstart -tunnel` if you want to start the platform server with the TLS tunnel option.
 - Enter `cfstart -ipv6` if you want to start the platform server for IPv6 without an SSL certificate.
 - Enter `cfstart -ssl -ipv6` if you want to start the platform server for IPv6 with an SSL certificate.
 - Enter `cfstart -tunnel -ipv6` if you want to start the platform server for IPv6 with the TLS tunnel option.



If you want to start the platform server with the SSL or Tunnel option, ensure that you have set up an SSL certificate before starting TIBCO MFT Platform Server. For more information, see [TLS/SSL Certificates Setup](#).

If you want to start the platform server for IPv6, ensure that you have configured the IPv6 ports in the `config.txt` file. For more information, see [Configuration Parameters](#).

Stopping TIBCO MFT Platform Server

You can stop TIBCO MFT Platform Server by using the **cfstop** command.



The **cfstop** command must be used by the root user or the user that installed the platform server.

Procedure

1. Optional: On the command line, navigate to the `$CFROOT/bin` directory.
2. Use the following commands to stop TIBCO MFT Platform Server:

- Enter `cfstop` to stop the platform server without an SSL certificate.
- Enter `cfstop -ssl` to stop the platform server with an SSL certificate.
- Enter `cfstop -tunnel` to stop the platform server with the TLS tunnel option.
- Enter `cfstop -ipv6` to stop the platform server for IPv6 without an SSL certificate.
- Enter `cfstop -ssl -ipv6` to stop the platform server for IPv6 with the SSL option.
- Enter `cfstop -tunnel -ipv6` to stop the platform server for IPv6 with the TLS tunnel option.

Verifying TIBCO MFT Platform Server

You can verify if TIBCO MFT Platform Server is running by using the `ps | grep` command.

Procedure

1. Optional: On the command line, navigate to the `$CFROOT/bin` directory.
2. Enter `ps -ef | grep CyberResp | grep -v grep` to verify whether TIBCO MFT Platform Server is running.

See the following example of output after running the command:

```
root      3050      1  0 11:03 pts/0      00:00:00 /mftps720/CyberResp
root      3096      1  0 11:03 pts/0      00:00:00 /mftps720/CyberResp -6
root      3140      1  0 11:03 pts/0      00:00:00 /mftps720/CyberResp -ssl
root      3182      1  0 11:04 pts/0      00:00:00 /mftps720/CyberResp -ssl6
```

Displaying Version Information of TIBCO MFT Platform Server

You can display version information of TIBCO MFT Platform Server by using the `cfstart` command.

Procedure

1. Optional: On the command line, navigate to the `$CFROOT` directory.
2. Enter `./cfstart -v` to display the version information.

For example:

```
MFT Platform Server cfstart Version 8.0 build 7
```

TLS/SSL Certificates Setup

TIBCO MFT Platform Server supports TLS/SSL protocols, so that you can access TIBCO MFT Platform Server securely and encrypt all data involved in a file transfer.

You can perform the following tasks before performing TLS/SSL file transfers:

- [Generating a Certificate Request](#)
- [Viewing a Certificate](#)
- [Encrypting the SSL Private Key Password](#)
- [Configuring a Certificate in TIBCO MFT Platform Server](#)

Generating a Certificate Request

You can use the `sslutility.exe` utility to generate certificate request and private key files.

Prerequisites

Before using the utility, you must create a directory where you can save the generated files.



Ensure no spaces are embedded in the directory names and file names.

Procedure

1. On the command line, navigate to the `$CFROOT/util` directory.
2. Enter `./sslutility.exe`.
3. Enter 1 after the SSL Utilities menu is displayed.
4. Enter the required information to generate a specific certificate request.

Result

The `sslutility.exe` utility generates a certificate request file and a private key file. Then the certificate request file is sent to a certificate authority.

The following example shows how to use the **sslutility.exe** utility. For more information regarding the parameters of the utility, see [Parameters in sslutility.exe Utility](#).

```
SSL Utilities Menu
1. Generate a Certificate Request
2. View a Certificate
3. Exit

Please enter your choice: 1

Generate Certificate Request Menu

Please enter the certificate holder's name:
SystemA

Please enter the Organization Name:
OrgA

Please enter the Department Name:
Quality Assurance

Please enter the City:
Garden City

Please enter the State:
NY

Please enter the Country:
US

Please enter the Email Address:
qatest1@orga.com

Please select a key length:
1. 1024 ( default )
2. 2048
3. 4096
1

Please select signature algorithm:
1. sha1 ( default )
2. sha256
3. sha384
4. sha512
1

Please enter the location and file name for the Certificate Request that will be
created:
/mftps/certs/certreq.test

Please enter the location and file name for the Private Key that will be created:
/mftps/certs/privatekey.test

Please enter the password for the Private Key File:

Please re-enter the password for the Private Key File:

Please enter a directory to which you have write access
or hit enter for the default directory: [/tmp].






Generating RSA private key, 1024 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
.

**** Request successfully created. ****
SSL certificate request created in file: [/mftps/certs/certreq.test]
SSL private key file created in file: [/mftps/certs/privatekey.test]
```

Parameters in the sslutility.exe Utility

You can use the **sslutility.exe** utility to set parameters while generating an SSL certificate request.

The following table lists the parameters you must supply when running the **sslutility.exe** utility:

Parameter	Description
Certificate Holder's Name	Defines the IP address or host name of the machine that requires the certificate.
Organization	Defines the group or company with which the certificate holder is associated.
Organizational Unit	Defines the department within the organization that makes the request.
City	Defines the city of the certificate holder.
State	Defines the state of the certificate holder.
Country	Defines the country code of the certificate holder.  The value of a country code must be 2 characters.
Email address	Defines the email address that is associated with the certificate holder.  The maximum length of a email address is 256 characters.
Certificate Key Length	Defines the key length of the certificate. The valid key lengths are 1024, 2048 or 4096. The default value is 1024.
Certificate Signature Algorithm	Defines the signature algorithm of the certificate. The valid algorithms are sha1, sha256, sha384 or sha512 . The default value is sha1.
Certificate Request File Name	Defines the full name of the certificate request file.  Ensure that there are no spaces in the file name.
Private Key File Name	Defines the full name of the private key file.  Ensure that there are no spaces in the file name.
Private Key Password	Defines the password used to access a private key.  The maximum length of a private key password is 20 characters.

Parameter	Description
Directory to Place the Generated Files	Defines the directory where the certificate request file and private key file are. Press Enter to use the default /tmp directory.

Viewing a Certificate

You can use the **sslutility.exe** utility to view the details of a certificate.

Procedure

1. On the command line, navigate to the `$CFROOT/util` directory.
2. Enter `./sslutility.exe`.
3. Enter 2 after the SSL Utilities menu is displayed.
4. Enter the full path of the certificate file that you want to view.

Result

Detailed information regarding the specific certificate is displayed.

The following example shows the detailed information of a certificate:

```
SSL Utilities Menu
1. Generate a Certificate Request
2. View a Certificate
3. Exit

Please enter your choice: 2

View Certificate Menu

Please enter the Certificate Filename:
/mftps/certs/cert.AIX_NEW
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 90 (0x5a)
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=US, ST=NY, L=GC, O=OrgA, OU=Dev1A390, CN=a390.orga.com
    Validity
      Not Before: Jul  1 04:00:00 2009 GMT
      Not After : Jan  1 03:59:59 2016 GMT
    Subject: C=US, ST=NY, L=Garden City, O=OrgA, OU=QA, CN=QA Test/
emailAddress=qatest@orga.com
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (1024 bit)
      Modulus (1024 bit):
        00:f2:0a:92:e8:b7:ad:b9:8d:b4:21:7f:1b:bd:8c:
```

Encrypting a Private Key Password

You can use the **createPwd.exe** utility to encrypt a password for a private key, and save the password in a password file.

Prerequisites

Generate a certificate request file and a password file to save the encrypted password. For more instructions, see [Generating a Certificate Request](#).

Procedure

1. On the command line, navigate to the `$CFROOT/util` directory.
2. Enter `./createPwd.exe`.
3. Enter the password and a full path to the file where you want to place the encrypted password.

Result

After running the **createPwd.exe** utility, the private key password is encrypted and stored in the password file.

The following example shows how to encrypt an SSL private key password.

```
-bash-3.00$ ./createPwd.exe
Please enter your Password (Max: 20 characters)...
Please Reenter your Password to Confirm ...
Please specify a Path and File Name for the encrypted password to be saved in...
/mftps/certs/passwordfile
Thank you.....
Your encrypted Password has been saved in: /mftps/certs/passwordfile
```

Configuring a Certificate

You can modify specific parameters in the `config.txt` file to use an SSL certificate.

Prerequisites

Before modifying the `config.txt` file, ensure that you have received a certificate from your certificate authority.

Procedure

1. On the command line, navigate to the `$CFROOT/config` directory.
2. Open the `config.txt` file by using any text editor.
3. Navigate to the SSL Communication Additional Parameters part in both SERVER and CLIENT sections.
4. Configure the following parameters according to your certificate:
 - **CertificateFileName**
 - **PrivateKeyFileName**
 - **PrivateKeyPwdFileName**
 - **TrustedAuthorityFileName**

The following is an example of the SSL Communication Additional Parameters part in the SERVER section:

```
# SSL Communication Additional Parameters.
SSLPort: 56565
SSLPortIPv6: N { N, IPv6 Port }
TunnelPort: 58585
TunnelPortIPv6: N { N, IPv6 Port }
ClientVerification: N { N, Y }
CertificateFileName: /mftps/certs/cert.test
PrivateKeyFileName: /mftps/certs/privatekey.test
PrivateKeyPwdFileName: /mftps/certs/passwordfile
TrustedAuthorityFileName: /mftps/certs/certauth.all
AuthorizationFileName: N { N, FileName }
SSLTraceLevel: N { N, Y }
SSLTracePath: /mftps/trace/SSLResponder { N, Path }
CheckCRL: N { N, Y }
CAPath:
SSEnabledProtocols: TLSV1,TLSV1.1,TLSV1.2 { TLSV1,TLSV1.1,TLSV1.2 }
Ciphers: HIGH { openssl_cipher_list }
```

The following is an example of the SSL Communication Additional Parameters part in the CLIENT section:

```
# SSL Communication. Additional Parameters.
CertificateFileName: /mftps/certs/cert.test
PrivateKeyFileName: /mftps/certs/privatekey.test
PrivateKeyPwdFileName: /mftps/certs/passwordfile
TrustedAuthorityFileName: /mftps/certs/certauth.all
SSLTraceLevel: N { N, Y }
SSLTracePath: /mftps/trace/SSLInitiator { N,
Path }
CheckCRL: N { N, Y }
CAPath:
SSEnabledProtocols: TLSV1,TLSV1.1,TLSV1.2 { TLSV1,TLSV1.1,TLSV1.2 }
Ciphers: HIGH { openssl_cipher_list }
```

5. To make the changed config.txt file effective for responder requests, restart TIBCO MFT Platform Server. The changed config.txt is immediately effective for initiator (that is, cfsend and cfrecv) requests.

Transfer Commands

You can use the **cfsend** or **cfrecv** command to perform various file transfers.

TIBCO MFT Platform Server supports the following types of file transfers:

- [File to File Transfers](#)
- [File to Job Transfers](#)
- [Running Remote Commands](#)
- [File to Print Transfers](#)

Use the **cfsend** or **cfrecv** command, along with transfer parameters and specified options to perform transfers. For more information on transfer parameters, see [Transfer Parameters](#).

To simplify your transfer commands, you can also use nodes (as well as profiles and distribution lists with nodes) and templates:

- Using nodes, you can predefine a remote location in a node before performing a transfer. This can save you from typing repetitive location information. For more information, see [Transfer Using Nodes](#). You can also create profiles and distribution lists with created nodes:
 - Using profiles, you can define credentials to use when the configured user initiates a transfer to the configured node. For more information, see [Transfer Using Profiles](#).
 - Using distribution lists, you can perform file transfers to multiple locations. For more information, see [Transfer Using Distribution Lists](#).
- Using templates, you can configure all the transfer information in a template file before performing a transfer. This can save you from specifying transfer parameters on a command line. For more information, see [Transfer Using Templates](#).



In a **cfsend** or **cfrecv** command, if you use a node or a template when you also specify transfer parameters on a command line, the parameters specified on a command line take higher precedence over nodes and templates.

You can add special characters after the **cfsend** or **cfrecv** command to define transfers in different circumstances:

- To run the command in the background, add an ampersand (&) at the end of a command.
- To log off before the command is completed, add `nohup` before a command.
- To send the screen output to a specified file, add `> the file path 2>&1` at the end of the command.

File to File Transfers

You can use the **cfsend** or **cfrecv** command to send or receive a file.

At a minimum, you must specify the following parameters to perform a file to file transfer:

- **Node**
- **LocalFileName**
- **RemoteFileName**
- **UserId** (If you define a profile, this parameter is not needed.)
- **Password** (If you define a profile, this parameter is not needed.)

For more descriptions on these parameters, see [Minimum Transfer Parameters](#).

If you set the **RequiredNodeDefinition** parameter to **y** in the **SERVER** section of the `config.txt` file, you must define nodes for all transfer requests. For more information, see [Transfer Using Nodes](#) and [Transfer Using Profiles](#).

If you set the security policy to **HIPAA** in either the `config.txt` file or in a node definition, this setting always takes precedence.

If you set the **Encryption** parameter to **Never** and the **Compression** parameter to **Never** in a node definition on the initiator side, these settings always take precedence.

Examples: Transfer Using `cfsend` or `cfrecv` Command

To perform a file transfer on the command line, you can use the **cfsend** or **cfrecv** command with specific parameters and options.

Use equal signs (=) or colons (:) to separate the transfer parameters from their values.

The examples below assume that a profile has been configured for the defined node.

- To send a file to a remote system, use the **cfsend** command. For example:

```
cfsend node:NYServer lf:/home/usr/file rf:dataset.name
```
- To receive a file from a remote system, use the **cfrecv** command. For example:

```
cfrecv node:NYServer lf:/home/usr/file rf:"c:\temp\test.txt"
```
- To send a file using SSL to a remote system, you must set the **ssl** parameter to **y** and define the **sport** parameter in the **cfsend** command. For example:

```
cfsend n:NYTunnelNode lf:/home/usr/file rf:dataset.name ssl:Y
```
- To run a command in the background, add an ampersand (&) at the end of the command. For example:

```
cfrecv n:ParisNode lf:/home/usr/file rf:"c:\temp\test.txt" &
```
- To log off before a command is completed, add a prefix **nohup** at the beginning of the command. For example:

```
nohup cfrecv n:ParisNode lf:/home/usr/file rf:"c:\temp\test.txt" > /tmp/file 2>&1
```
- To send screen output of a command to a specific file, add the file path at the end of the command. For example:

```
cfsend n:LondonNode lf:/home/usr/file rf:dataset.name
```

Wildcards

To perform transfers for multiple files, you can use the wildcards: asterisks (*) and question marks (?).

For UNIX platforms, * means any number of symbols in a file name and ? means any single symbol in a file name. They have exactly the same meaning on each platform as they do in the operating system. Only files with file names that satisfy the selection criteria can be matched and transferred. To restrict matched files, use any combination and amount of these wildcards and alpha numeric characters. If you want to transfer an entire directory, add a single * after the directory path.

For example, you can use the file name `/home/johndoe/r?t*` to match `/home/johndoe/returns` and `/home/johndoe/ratelist`, but not `/home/johndoe/report`.

You must put wildcards after the last forward slash (/) to interpret them. Wildcards before the last forward slash (/) are ignored and will cause errors.

Directory Transfers




By specifying certain transfer parameters and options, you can also send and receive entire directories.

For more information on the directory transfer parameters and options, see [Directory Transfer Parameters](#).

You can use tokens and wildcards to define paths of directories and files in a more efficient way. For more information, see [File Name Tokens](#) and [Wildcards](#).

Directory Transfer Parameters

The following table shows parameters you can use to perform directory transfers:

Parameter (Alternate Specification)	Description
DirTransfer0Files (d0)	<p>Defines the status of the directory transfer when zero files are in the directory.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> F: When no files are transferred on a directory transfer, the transfer is a failed transfer. S: When no files are transferred on a directory transfer, the transfer is a successful transfer. <p>The default value is F.</p>
ScanSubDir (ssd)	<p>Defines whether all subdirectories from the file path are scanned.</p> <div>  This parameter only applies to directory transfers. </div> <p>The valid values are Y or N.</p>
StopOnFailure (sonf)	<p>Defines whether to stop transferring the rest of files in the directory when the current file transfer fails.</p> <div>  This parameter applies to directory transfers and distribution list transfers. </div> <p>The valid values are Y or N.</p>
Test	<p>Defines whether to display the file names without doing the actual transfers, so that you can verify whether the file names are correct before the actual transfers.</p> <div>  This parameter applies to directory transfers and distribution list transfers. </div> <p>The valid values are Y or N.</p>


Directory Tokens

To perform directory transfers, you can enter directory names with tokens to save your time.



Tokens are case sensitive.

The following table lists the tokens that you can use in directory transfers:

Token	Description
\$(SDIR)	<p>You can use this token:</p> <ul style="list-style-type: none"> • In the LocalFileName parameter when using a cfrecv command. • In the RemoteFileName parameter when using a cfsend command.
\$(MEMBER)	<p>You can use this token to make names of received files in the local system the same as member names in a remote z/OS system.</p> <div>  <p>Use this token only when you perform a receive transfer from a remote z/OS system.</p> </div> <p>If there is no \$(MEMBER) token in the file name from the z/OS side, this token might be ignored.</p> <p>For example, if you use <code>/mftps/\$(MEMBER)/filename</code> as a file path, then the file path can be resolved to <code>/mftps/data/filename</code>.</p>

Examples: Directory Transfers

You can use wildcards and tokens to perform different directory transfers.

1. To receive all the files in a directory folder, you can use a wildcard on the command line. For example:

```
cfrecv LocalFileName: '/home/johndoe/data/$(RemoteFileName)' RemoteFileName: '/mftps/data/*' n:MFTNode
```

As a result of this example, all files with the same file names in the `/mftps/data/` directory are received.

You can also use the special `*` token in place of the `$(RemoteFileName)` token:

```
cfrecv LocalFileName: '/home/johndoe/data/*' RemoteFileName: '/mftps/data/*'
```

2. To send the files in the subdirectories for a directory transfer, you can set **ScanSubDir** to **Y** on the command line. For example:

```
cfsend ScanSubDir:Y n:PhoenixNode LocalFileName: /home/johndoe/data/* RemoteFileName: '/mftps/data/$(LocalFileName)
```

As a result of this example, all files and subdirectories in the `/home/johndoe/data` directory are sent.

You can also use the special `*` token in place of the `$(LocalFileName)` token:

```
cfsend ScanSubDir:Y n:PhoenixNode LocalFileName: /home/johndoe/data/* RemoteFileName: '/mftps/data/*'
```

3. To receive a directory and also duplicate the directory structure from the remote system, you can use the **\$(SDIR)** token:

```
cfrecv ScanSubDir:Y LocalFileName: '/home/johndoe/data/$(SDIR)/$(RemoteFileName)' RemoteFileName: '/mftps/data/*' n:MontrealNode
```

As a result of this example, all files and subdirectories in the `/mftps/data/` directory are received with the same structure.

File Name Tokens

You can use file name tokens to name transferred files or to match files that you want to transfer, on either a responder (server) or an initiator (client).

A string of tokens are characters that contain a mixture of literal and substitution values. You can use file name tokens to format file names based on date, time, user information and so on. Therefore, instead of entering a standard file name, you can enter a name that consists of tokens to save time.

There is one special token: `*`.

- If you perform a `cfsend` and include a `*` on the remote file name, the `cfsend` command will substitute the local file name for the `*`.
- If you perform a `cfrecv` and include a `*` on the local file name, the `cfrecv` command will substitute the remote file name for the `*`.

The format of the file name using the token is `$(tokenname)`. All tokens are converted before being sent to the remote system, except for **RemoteTransactionNumber**.




When you use the dollar sign (\$) or the forward slash sign (/) on the command line, some UNIX systems might require a backslash (\) before these signs. Optionally, you can put single quotes around the entire parameter that contains the token. Tokens are case sensitive. A list of file name tokens can be displayed by the `cfsend` or `cfrecv` commands by entering the following commands:

```
cfsend /htoken
```



```
cfrecv /htoken
```



The following table shows the file name tokens that are supported:

Token	Description	Generated Value
JDate	Julian date	YYYYDDD
Time	Local time	HHMMSSMSS
Time1	Local time	HHMMSS
Time2	Local time	HHMMSST
Date	Local date	YYYYMMDD
Date1	Local date	YYMMDD
Date2	Local date	MMDDYY
Date3	Local date	DDMMYY
DateUS	Local date	MMDDYYYY

Token	Description	Generated Value
LocalFileBase	The base name of the local file.	<p>For example:</p> <ul style="list-style-type: none"> Local file: /home/usr/temp/files/testfile1.txt Remote file: c:\target\[LocalFileBase] <p>The resolution to this token is testfile1.</p>
LocalFileExt	The extension of the local file.	<p>For example:</p> <ul style="list-style-type: none"> Local file: /home/usr/temp/files/testfile1.txt Remote file: c:\target\[LocalFileExt] <p>The resolution to this token is txt.</p>
LocalFileName	The local file name including the extension.	<p>For example:</p> <ul style="list-style-type: none"> Local file: /home/usr/temp/files/testfile1.txt Remote file: c:\target\[LocalFileName] <p>The resolution to this token is testfile1.txt.</p> <div>  <p>You must use the full path of the remote file, otherwise, the file might be transferred to the \$CFROOT directory.</p> </div>
LocalFL##	<p>The file qualifier computed from left-justified position specified by ##.</p> <div>  <p>The range of ## is within 1 to 16.</p> </div>	<p>For example:</p> <ul style="list-style-type: none"> Local file: /home/usr/temp/files/testabc.aaa.bb.c.txt Remote file: c:\target\[LocalFL01] <p>The resolution to this token is testabc.</p>
LocalFR##	<p>The file qualifier computed from right-justified position specified by ##.</p> <div>  <p>The range of ## is within 1 to 16.</p> </div>	<p>For example:</p> <ul style="list-style-type: none"> Local file: /home/usr/temp/files/testabc.aaa.bb.c.txt Remote file: c:\target\[LocalFR01] <p>The resolution to this token is txt.</p>

Token	Description	Generated Value
NoLocalFileBase	The local file name excluding the base name.	<p>For example:</p> <ul style="list-style-type: none"> Local file: /home/usr/temp/files/a.b.c.txt Remote file: c:\target\[NoLocalFileBase] <p>The resolution to this token is b.c.txt.</p>
NoLocalFileExt	The local file name excluding the extension.	<p>For example:</p> <ul style="list-style-type: none"> Local file: /home/usr/temp/files/a.b.c.txt Remote file: c:\target\[NoLocalFileExt] <p>The resolution to this token is a.b.c.</p>
LocalUserId	The local user ID being used for the file transfer.	<p>For example:</p> <ul style="list-style-type: none"> Local user ID: TESTLAB\cfuser1 Local file: /home/usr/temp/files/testfile1.txt Remote file: c:\target\file1\$(LocalUserId).txt <p>The resolution to this token is d:\target\file1cfuser1.txt.</p>
TransactionNumber	Local transaction number	<p>For example:</p> <ul style="list-style-type: none"> Local file: /home/usr/temp/files/testfile1.txt Remote file: c:\target\[TransactionNumber].testfile1.txt <p>The resolution to this token is IA18100117.testfile1.</p>
RemoteUserId	Remote user ID used in the file transfer.	<p>For example:</p> <ul style="list-style-type: none"> Remote user ID: TEST\cfuser1 Local file: /home/usr/files/file1.\$(RemoteUserId).txt Remote file: c:\source\directory\testfile1.txt <p>The resolution to this token is /home/usr/files/file1.cfuser1.txt.</p>

Token	Description	Generated Value
RemoteTransactionNumber	Remote transaction number.	<p>For example:</p> <ul style="list-style-type: none"> Local file: /home/usr/temp/files/testfile1.txt Remote file: c:\target\[RemoteTransactionNumber].testfile1.txt <p>The resolution to this token is RA18100052.testfile1.</p>
 The following tokens can be only used in receive transfers.		
RemoteFileBase	The base name of the remote file.	<p>For example:</p> <ul style="list-style-type: none"> Local file: /home/usr/temp/[RemoteFileBase] Remote file: c:\source\directory\testfile1.txt <p>The resolution to this token is testfile1.</p>
RemoteFileExt	The extension of the remote file.	<p>For example:</p> <ul style="list-style-type: none"> Local file: /home/usr/files/[RemoteFileExt] Remote file: c:\source\directory\testfile1.txt <p>The resolution to this token is txt.</p>
RemoteFileName	The remote file name including the extension.	<p>For example:</p> <ul style="list-style-type: none"> Local file: /home/usr/files/[RemoteFileName] Remote file: c:\source\directory\testfile1.txt <p>The resolution to this token is testfile1.txt.</p> <div>  You must use the full path of the remote file, otherwise, the file might be transferred to the \$CFROOT directory. </div>

Token	Description	Generated Value
RemoteFL##	<p>The file qualifier computed from left-justified position specified by ##.</p>  <p>The range of ## is within 1 to 16.</p>	<p>For example:</p> <ul style="list-style-type: none"> Local file: /home/usr/files/\$ (RemoteFL01) Remote file: c:\source\directory\testabc.aaa.bb.c.txt <p>The resolution to this token is testabc.</p>
RemoteFR##	<p>The file qualifier computed from right-justified position specified by ##.</p>  <p>The range of ## is within 1 to 16.</p>	<p>For example:</p> <ul style="list-style-type: none"> Local file: /home/usr/files/\$ (RemoteFR01) Remote file: c:\source\directory\testabc.aaa.bb.c.txt <p>The resolution to this token is txt.</p>
NoRemoteFileBase	The remote file name excluding the base name.	<p>For example:</p> <ul style="list-style-type: none"> Local file: /home/usr/temp/files\$ (NoRemoteFileBase) Remote file: c:\source\directory\ a.b.c.txt <p>The resolution to this token is b.c.txt.</p>
NoRemoteFileExt	The remote file name excluding the extension.	<p>For example:</p> <ul style="list-style-type: none"> Local file: /home/usr/temp/files\$ (NoRemoteFileBase) Remote file: c:\source\directory\ a.b.c.txt <p>The resolution to this token is a.b.c.</p>
UserData	The user data defined by the cfsend or cfrecv commands.	<p>For example:</p> <p>cfsend ud:AcctFile</p> <p>The resolution to this token is AcctFile.</p>
\$(SYYYY)	The year to be specified using 4 digits. The valid values are from 0000 to 9999.	For example: 2018
\$(SYY)	The year to be specified using the last two digits of the year. The valid values are from 00 to 99.	For example: 18 for the year 2018

Token	Description	Generated Value
\$(SMON)	The month of the year to be specified using the first three letters and upper case.	For example: JAN for the month of JANUARY
\$(SMon)	The month of the year to be specified using the first three letters and lower case.	For example: Jan for the month of January
\$(SMM)	The month of the year to be specified as a number. The valid values are from 01 to 12.	For example: 02 for the month of February
\$(SDD)	The date or day of the month to be specified. The valid values are from 01 to 31.	For example: 05 for the fifth day of the month
\$(SJ)	The Julian day(ddd).	For example: 320

Examples: Transfer Using File Name Tokens

To send a file from a UNIX system to a Windows system, you can use the **cfsend** command with tokens. For example:

```
cfsend lf:/home/usr/file rf:'c:\reports\$(LocalFileBase).\$(Date1)-\$(Time1).\$(LocalFileExt)' node:TorontoNode
```

In the Windows system, the result is expected to be the `revenue.080929-095201.txt` file in the `c:\reports` directory.

Post Processing Actions (PPA)

With PPA, you can use up to four commands after a file transfer is completed either successfully or unsuccessfully.

See the following PPA parameters you can use:

- **Post_Action1** or **ppa1**
- **Post_Action2** or **ppa2**
- **Post_Action3** or **ppa3**
- **Post_Action4** or **ppa4**

For more information, see [Optional Transfer Parameters](#).

PPA Command Format

You must follow the command format to perform post processing actions after file transfers.

The PPA command format is:

```
Post_Action<number>="S|F,L|R,COMMAND|CALLJCL|CALLPGM|SUBMIT,<command data>"
```

Use a comma (,) to separate each of the following sections of a PPA command:

- **S|F**

Defines if you want to run the action when the file transfer is successful or unsuccessful.

- **L|R**

Defines if you want to run the action in the local system (initiator) or in the remote system (responder).

- **COMMAND | CALLJCL | CALLPGM | SUBMIT**

Defines which action you want to run.



You can run **CALLJCL**, **CALLPGM** and **SUBMIT** if the remote system is mainframe, otherwise you can only run **COMMAND**.

- **<command data>**

Defines the full path and file name of a command with associated parameters. This section is limited to 256 bytes.



Ensure no spaces are embedded in a PPA command. If one PPA parameter is used on the command line, the command definition goes into the next undefined PPA parameter.

See the following command for an example of how to use PPA with an executable command:

```
ppa1="S,L,COMMAND,batchjob.exe"
```

PPA Substitutable Parameters

When using PPA substitutable parameters, you can modify the **<command data>** section in any PPA command, to save you from specifying the **LocalFileName** or **RemoteFileName** parameters.



If you use PPA commands, then you cannot use standard tokens. Standard tokens are not supported because they are relatively long, so using PPA substitutable parameters can save bytes.

Use the percentage sign (%) as the escape character when using PPA substitutable parameters.

A list of PPA tokens can be displayed by the **cfsend** or **cfrecv** commands by entering the following commands:

- **cfsend /hppa**
- **cfrecv /hppa**

The following table lists the substitutable parameters that are supported:

Parameter	Description	Resolved Name Examples (Based on C:\a\b\c\d\config.txt)
%DIR	Directory name without the file name and drive name	a\b\c\d
%DRIVE	Drive name	C
%FILE	File name without the directory	config.txt
%HDIR	High level directory	a
%HLQ	High level qualifier of file	config
%LFILE	File name with directory	C:\a\b\c\d\config.txt
%LLQ	Low level qualifier of file (data after the last period mark)	txt

Parameter	Description	Resolved Name Examples (Based on C:\a\b\c\d\config.txt)
%NODRIVE	File name without drive name	a\b\c\d\config.txt
%NOHDR	Directory name without high level directory	b\c\d
%NOSDIR	Directory name without lowest level directory	a\b\c
%SDIR	Lowest level directory	d
%GDATE	Gregorian date (yymmdd)	
%GDATEC	Gregorian date with century (ccymmdd)	
%JDATE	Julian date (YYDDD)	
%JDATEC	Julian date with century (CCYYDDD)	
%LUSER	Local user ID	
%PROC	Process name	
%RUSER	Remote user ID	
%TIME	Time (hhmmss)	
%TRN	Transaction number	
%UDATA	User data	



You can use multiple PPA substitutable parameters within a single PPA command. Each substitutable parameter must be processed one at a time.

Because there is no drive in UNIX systems, if you use the %**DRIVE** parameter in a UNIX environment, then the % might be removed and **DRIVE** might be used as a substitution.

PPA Error Codes

For PPA executed on the local machine, the PPA return codes are saved in the audit file. PPA error messages are recorded if you set the trace level to medium (M) or high (H). A trace file is created for each file transfer and is located according to the **TracePath** parameter you define in the `config.txt` file.

For remotely used PPA commands, you can determine if a PPA command has failed or not by checking if the trace file has a positive error code.

For locally used PPA commands, the output of the PPA commands can be displayed in the system terminal, so you do not have to check the trace file.

File to Job Transfers

You can use the **cfSEND** or **cfrecv** command to send or receive a file and run it as a job remotely or locally.

To perform a file to job transfer, you have to set the **TransferType** parameter to J (J stands for job). This parameter can be set on the command line or in templates.

File to job transfers can be performed in either direction. You can receive a file from a remote system and run it locally, or send a file to a remote system and run it remotely. The file name can be either local or remote, depending on transfer directions.

- If you receive a file from a remote system and run it in the local system, specify only the name of the remote file, which is an executable file. You do not have to specify a local file name because the output is not written to any local file.
- If you send a file to a remote system and run it in the remote system, specify only the local file name, which is an executable file. You do not have to specify a remote file name because the output is not written to any remote file.

Examples: File to Job Transfers

The following example is a file to job transfer using the **cfSEND** command:

```
cfSEND lf:/home/usr/job n:LANode trtype:j
```

Running Remote Commands

You can use the **cfSEND** command on the command line to send a command and run it in a remote system.

To run a command remotely, specify the **TransferType** parameter as C (C stands for command). This parameter can be set on a command line or in templates.

You have to specify both the type of command and the actual command you want to use.

Remote Command Types

Use different parameters to indicate the type of a command:

- If the remote system is a Windows or UNIX system, the parameter is:

rcmd | **remotecommand**

- If the remote system is a z/OS system, the parameters are:

- **e**: | **exec**: and **re**: | **rexxexec**: used for an executable.
- **sj**: | **subjcl**: used for submitting job control language.
- **cj**: | **calljcl**: used for calling programs with JCL linkage.
- **cpg**: | **callpgm**: used for calling a program with standard linkage.



Each of the parameters must be followed by the command to be used.

Examples: Running Remote Commands

The following example is to run a remote command using the **cfSEND** command:

```
cfSEND n:UNIXNode trtype:C rcmd:"ls -la"
```

Error Handling: Running Remote Commands

If you define a local file name when running a command remotely, the output of the transfer, whether successful or unsuccessful, is returned and stored with error codes.

When running a remote command in a Windows or UNIX system, you can define the **LocalFileName** parameter in the **cfsend** command, to store the output of the remote command. Otherwise, the output is written to your terminal. The **RemoteFileName** parameter is invalid and ignored. Based on how streams work in UNIX, stdout data is printed first, followed by the last 256 characters of stderr.

The types of error codes are listed as follows:

- When the function is successful, the return code is set as 0, and any output data is returned to the caller, in the same way as any other command.
- When the function is unsuccessful, the return code is set to a non-zero value, and a send error is returned to the caller along with a message indicating the cause of the failure.
- For network errors, the return code is normally 4, and the function can be retried.
- For severe errors, the return code is normally 8, and the function cannot be retried.

See the following list of errors that are not retried:

- Access control errors.
- Bad user ID or password.
- Checkpoint errors.
- CfAlias errors.
- Errors while starting a conversation.
- Errors while writing to a PQF file.
- Errors while trying to run the **chmod** command on the PQF file.
- Failure to connect SSL port without proper handshakes.
- Failure to apply a umask.
- Failure to run the **cfdir** or **fusutil** command.
- Failure to open files.
- Invalid encryption type for international version.
- Invalid encryption for HIPAA.
- Malloc errors (when system runs out of memory).
- Node is not defined when **requirednodedefinition** is set as Y.
- CRL authentication errors.
- SSL authentication errors.
- Severity-1 errors from the remote system.
- Security violation during Negotiation/Control record (when SSL and encryption is used).
- Tokens and wildcard character errors.

File to Print Transfers

You can send a file to a remote system, and run it as a print job.

To perform a file to print transfer, you have to set the **TransferType** parameter to P (P stands for print). This parameter can be set on a command line or in templates.

When performing a file to print transfer, the value of **LocalFileName** parameter is the name of the file you want to print, and the **RemoteFileName** defines the printer name on the remote system.

Transfer Using Nodes

Using pre-defined node definitions for remote systems, you do not have to constantly provide information to TIBCO MFT Platform Server when conducting transfers with remote systems. We suggest using node and profile definitions to pre-define the connectivity and authentication parameters. This makes the product more secure and less prone to errors.

The settings for each remote system are stored in a clear text file named `cfnode.cfg` located in the `$CFROOT/config` directory. Once a node definition is created, you can specify the name of the node to perform a transfer as opposed to using numerous transfer parameters to get the same results from a file transfer. TIBCO MFT Platform Server checks the definition for the specified node to obtain the required parameters to perform a transfer.

Node definitions define default parameters required by TIBCO MFT Platform Server to interact with a remote system (node). This information includes:

- Node name
- System type
- IP address or host name for TCP/IP transfers
- Port number for TCP/IP transfers
- (Optional) Security compliance level (used for HIPAA and FIPS mode transfers)
- (Optional) Netmask for remote IP address
- (Optional) Netmask6 for remote IPv6 address
- (Optional) Use of TLS/SSL for secure communications
- (Optional) Default compression type
- (Optional) Default encryption type
- (Optional) Default local translation file
- (Optional) Default remote translation file
- (Optional) Whether responder profiles are used
- (Optional) Whether verified users are accepted
- (Optional) Text description for the node
- (Optional) CRC checking
- (Optional) Supported Command Center functions

Creating Nodes

You can use the **cfnode** utility to add, update, and delete node definitions.



Only the superuser (root) or members of the `cfadmin` group can create nodes.

Procedure

1. Optional: On the command line, navigate to the `$CFROOT/bin` directory.
2. Enter `cfnode prompt:Yes`.

You are prompted for both the required and optional parameters to define a remote system (node).



Without the `prompt:YES` option (or with the `prompt:NO` option), you can define values for the required parameters and only the optional parameters you want to use. See the following sample command:

```
cfnode n:dataserverA s:Windows
net:255.255.255.0 h:192.168.0.43 p:46464
ssl:Y c:RLE e:NEVER security:Default v:Y r:D
d:"This is a sample node definition"
ccc:TRANSFER prompt:NO
```

3. On the prompt, set the values for both the required and optional parameters.



If the node name you define already exists, you are prompted as to whether you want to edit the defined node.

See the following sample of creating a Windows node with the `cfnode prompt:YES` command.


```

> cfnode prompt:YES
Enter a valid node name: dataServerB
Enter a System Type for Node[dataServerB]:
1: HPUX
2: SUNOS/SOLARIS
3: AIX
4: LINUX
5: Windows
6: IBMi
7: z/OS
8: Command_Center
9: Other
: 5
Enter a valid IP address for Node[dataServerB]: 192.168.0.44
Would you like to specify netmask for remote IpAddress:
1: Yes
2: No
: 2
Enter the port for which Node[dataServerB] is configured to use: 46464
Enter the Security Compliance level for file transfers:
1: Default ( use Security Policy setting from config.txt )
2: None
3: HIPAA
: 1
Should TLS/SSL be used:
1: NO
2: YES (TLS/SSL compatibility mode)
3: TLS TUNNEL
: 1
What should be the default encryption used:
1: DES
2: 3DES
3: BF
4: BFL
5: RIJN(AES)
6: AES128
7: No default encryption
8: Never use encryption
: 3
What should be the default compression used:
1: LZ
2: RLE
3: ZLIB
4: No default compression
5: Never use compression
: 2
Would you like to specify local translation file:
1: Yes
2: No
3: NONE (Caution! If uncertain, refer to User Guide.)
: 1
Please enter local translation file:
: MyComtblg.dat
Would you like to specify remote translation file:
1: Yes
2: No
: 2
Accept Verified Users from this node?
1: Yes
2: No
3: Do not define
: 1
Use Responder Profiles for this node?
1: Yes
2: No
3: Dual
4: Do not define
: 3
Would you like to add a description:
1: Yes

```

```

2: No
: 2
Enter the MFT Command Center parameters this node will support:
1: All
2: None
3: Audit
4: Node
5: Ping
6: Profile
7: Transfer
: 1
Do CRC checking for this node?
1: Yes
2: No
3: Default
: 2
A Node definition was created for:
[new_1]
SystemType = SUNOS/SOLARIS
Protocol = tcpip
HostName = 10.102.52.14
Server = 46464
TLS = N
Compression = NO
Encryption = NO
Security Policy = None
AcceptVerifiedUser = N
ResponderProfile = D
CommandSupport = ALL
CRC = No
[root@rkundu-xw6200 ~] #

```

Node definitions are stored in the `cfnode.cfg` file located in the `$CFROOT/config` directory. After a node definition is created, you can specify the name of the node you want to use for the transfer. In addition to working with a remote TIBCO MFT Platform Server, you can also conduct transfers for this server using TIBCO MFT Command Center. For more information, see TIBCO MFT Command Center documentation.

Deleting Nodes

You can delete nodes using the `cfnode a:delete node:nodename` command.

Procedure

1. Optional: On the command line, navigate to the `$CFROOT/bin` directory.
2. Enter `cfnode a:delete node:nodename`.
The specified node is deleted.

Listing Nodes

You can list nodes using the `cfnode a:list` command.

Procedure

1. Optional: On the command line, navigate to the `$CFROOT/bin` directory.
2. Enter `cfnode a:list`.
The nodes defined for TIBCO MFT Platform Server are listed.

Node Parameters


The following tables list parameters supported for the **cfnode** command.

Required Node Parameters



If any of the required parameters is not defined and the **prompt** parameter is set to No, the **cfnode** command fails.

The following table lists the required parameters for the **cfnode** command.


Parameter (Alternate Specification)	Description
HostName (h)	<p>Defines the IP name or IP address of the node.</p> <p>This value can be either the IP address of the remote machine or a resolvable host name or DNS entry.</p> <p>Multiple IP names or IP address can be defined, separated by a comma. When more than one IP name or IP address is defined, the following rules apply:</p> <ul style="list-style-type: none"> For initiator requests (i.e. cfsend/cfrecv), only the first IP Name or IP Address is used. For responder requests, when the Platform Server responder is checking for a match on the Node name, each IP name or IP address is checked for a match. This feature is typically used when you want to use the same node and responder profile for incoming request from multiple platform servers.
Node (n)	<p>Defines the node name you want to add or update to the <code>cfnode.cfg</code> file.</p> <p>The maximum length of the defined value is 256 characters, and it cannot contain any spaces.</p> <div>  <p>The node name is not case sensitive.</p> </div>
Port (p)	Defines the port number on which the remote node is listening for incoming transfer requests.
SystemType (s)	<p>Defines the system type of the node.</p> <p>The valid values are SUNOS/SOLARIS, AIX, LINUX, Windows, IBMi, z/OS, Command_Center, and other system types.</p>


Optional Node Parameters



The **cfnode** command does not require any of the optional parameters to be defined if the **prompt** parameter is set to No.

The following table lists the optional parameters for the **cfnode** command.

Parameter (Alternate Specification)	Description
Action (a)	<p>Defines the action you want to take with the specified node.</p> <p>The valid values are Delete, List or Add.</p>
CommandSupport (ccc)	<p>Defines the actions that the TIBCO MFT Command Center can perform on this node.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • ALL: NODE, PROFILE, AUDIT, PING and TRANSFER are supported on this node. • NONE: no Internet Server function is supported on this node. This is the default value. • AUDIT: requests that access to the TIBCO MFT Platform Server audit file are supported on this node. • NODE: node list and update functions are supported on this node. • PING: the TIBCO MFT Platform Server fusing request is supported on this node. • PROFILE: profile list and update functions are supported on this node. • TRANSFER: TIBCO MFT Command Center transfer function that initiates file transfers is supported on this node.
Compression (c)	<p>Defines the compression type for all transfers with this node.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • LZ: Lempel-Zev (LZ) compression. • RLE: Run Length Encoding (RLE) compression. • ZLIB: ZLIB1 through ZLIB9. • NO: no default compression. • NEVER: never uses compression. <div>  <p>NEVER is the only value that cannot be overridden by cfsend or cfrecv on the command line.</p> </div>
CRC	<p>Defines whether a Cyclic Redundancy Check (CRC) is performed for transfers initiated to this node.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • Yes: performs CRC checking. • No: bypasses CRC checking. The default value is No.
Description (d)	<p>Defines a text description of the node.</p> <p>The maximum length of the defined value is 256 characters. If the description contains spaces, then the description must be encapsulated in double quotation marks.</p>

Parameter (Alternate Specification)	Description
Encrypt (e)	<p>Defines the encryption type for all transfers with this node.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • DES: 56-bit encryption. • 3DES: triple DES, 112-bit encryption. • BF: Blow Fish encryption, 56-bit encryption. • BFL: Blow Fish Long, 128-bit encryption. • RIJN(AES): AES/Rijndael, 256-bit encryption. • AES128: 128-bit encryption. • NO: no encryption. • NEVER: never uses encryption. <div>  <ul style="list-style-type: none"> • NEVER is the only value that cannot be overridden by cfsend or cfrecv on the command line. • AES128 is not supported for products with earlier versions. </div>
LocalCTFile (lct)	<p>Defines the name of the local conversion table (local translation file).</p> <p>This parameter is used to translate data on the local side.</p> <p>The valid values are <i>filename</i> or <i>No</i>. The maximum length of the defined value is 16 characters.</p>
NetMask (net)	<p>Defines the netmask that is applied to the remote IP address.</p> <p>This parameter is defined so that you can use any IP in the specified subnet.</p>
NetMask6 (net6)	<p>Defines the netmask that is applied to the remote IPv6 address.</p> <p>The valid values are a number between 8 and 128 and a multiple of 8.</p>
Prompt	<p>Defines whether to activate the cfnode command to an interactive mode.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • Yes: cfnode prompts the user for all information required to create a node. This is the default value. • No: cfnode does not prompt the user for all information required to create a node.
RemoteCTFile (rct)	<p>Defines the name of the remote conversion table (remote translation file).</p> <p>This parameter is used to translate the data remotely. Valid values are <i>filename</i> and <i>No</i>. The maximum length of the defined value is 16 characters.</p>

Parameter (Alternate Specification)	Description
Responder (r)	<p>Defines whether to use a responder profile for this node.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • Yes: Check the responder profiles and do not try to log on with the remote user ID and password that are sent with the transfer request. • No: Not check the responder profiles at all and try to log on with the remote user ID and password that are sent with the transfer request. • Dual D: Substitute a real user ID when <code>cfrprofile</code> exists and a match is found. If there is no match found, TIBCO MFT Platform Server tries to log on with the remote user ID and password sent with the transfer request. • Do not define:
Security (sl)	<p>Defines the security policy this node complies with.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • Default: Follow what is defined for the Security Policy parameter in the <code>\$CFROOT/config/config.txt</code> file. • HIPAA: Comply with HIPAA standards. • None: Not comply with any security policy.
SSL	<p>Defines whether SSL is used for TCP/IP communications.</p> <p>The SSL parameter is not required if the prompt parameter is set to No. For more information, see SSL Certificates Setup.</p>
Verify (v)	<p>Defines whether a remote verified user can log on to TIBCO MFT Platform Server using only the remote user ID without a password.</p> <p>TIBCO MFT Platform Server recognizes the client as verified if the client sends an internal password inside the password field.</p> <p>To enable this feature, the user on the client side has to provide the following parameters:</p> <ul style="list-style-type: none"> • Userid: *VER • Password: this field must be left blank.
/? or -?	Displays online help for the cfnode command.

Examples: Transfer Using Nodes

After you define the required parameters for file transfers, such as the remote IP address and port number, in the node definition, you can only provide the node name of the remote system with which you want to perform transfers.

In the following example, two nodes called `zos` and `windows` are defined. By using nodes, the first two examples given in [Examples: Transfers Using Command Line](#) can be simplified to the following:

```
cfSEND lf:/home/usr/file rf:dataset.name n:zos zOS:y
uid:zremote_user pwd:zremote_password

cfrecv lf:/home/usr/file rf:"c:\temp\test.txt"
n:windows uid:wremote_domain\wremote_userid
pwd:wremote_password
```



If transfer parameters are provided on the command line, they override equivalent parameters provided by the node definition. Exception to this rule occurs only when **Compression** or **Encryption** is set to `NEVER`, or when **Security** is configured to `default` to follow the FIPS140 standards as specified in the `config.txt` file. In these cases, the command line cannot override any of these options.

- Place an ampersand (&) at the end of the command to run the command in the background:

```
cfrecv lf:/home/usr/file rf:"c:\temp\test.txt"
n:windows uid:wremote_domain\wremote_userid
pwd:wremote_password &
```

- Prefix the command with `nohup` to log off before the `cfrecv` command is completed:

```
nohup cfrecv lf:/home/usr/file
rf:"c:\temp\test.txt" n:windows
uid:wremote_domain\wremote_userid
pwd:wremote_password
```

- Add the file path in the command to send the screen output to a file. In the following example, the output writes to `/tmp/file`:

```
cfrecv lf:/home/usr/file rf:"c:\temp\test.txt"
n:windows uid:wremote_domain\wremote_userid
pwd:wremote_password > /tmp/file 2>&1
```

Transfer Using Profiles

You can define local or responder profiles based on a created node. Each local or responder profile corresponds to a local or remote user ID.

Local and responder profiles are used for different transfer situations.

- **Local Profile:** local profiles are used when you initiate transfers from a local machine.
- **Responder Profile:** responder profiles are used when you receive transfer requests.

Local Profile

A local profile defines a remote user name and remote password that can be used by a local user. Each local profile definition is defined in a clear text file named `cfprofile.cfg` located in the `$CFROOT/config` directory.

A local profile contains the following information:

- Node: the remote system with which the local profile is associated
- Local user name: the name of the local user who can use this local profile
- Remote user name: the remote user name used to log on to the node (remote system)
- Remote password: the remote user password used to log on to the node (encrypted)

You can add or update local profiles through the `cfprofile` command. Before you update any information in `cfprofile.cfg`, a backup file called `cfprofile.bak` is created. You can activate local profiles by simply specifying the node for a transfer.

Creating Local Profiles

When a node is supplied on a command line or in a template, a local profile is chosen based on the current user. The information in this local profile is used to log on to the remote system.



If you are the root account or a member of the `cfadmin` group, you can create your own profile as well as profiles for other users.

Procedure

1. Optional: On the command line, navigate to the `$CFROOT/bin` directory.
2. Enter `cfprofile prompt:YES`.

You are prompted for the required and optional parameters to define a local profile.



Without the `prompt:YES` option (or with the `prompt:NO` option), you can define values for the required parameters and only the optional parameters you want to use. See the following sample command:

```
cfprofile n:dataserverA u:kenny p:apple l:uk
```

3. On the prompt, set the values for the required parameters.
See the following sample command:

```
cfprofile prompt:YES
Enter a valid Node Name: dataserverB
Add profile as local user ROOT?
1: Yes
2: No
: 2
Enter new local user: johndoe
Enter a valid Remote User: bob
Enter a valid Remote Password:
```


This example creates a local profile for the local user johndoe. This local user johndoe can initiate a transfer request to the remote server defined for the `dataserverB` node without having to know the user ID and password in the remote system. With this profile, johndoe only has to define the following on the command line:

```
cfSEND lf:/home/usr/file rf:dataset.name n:dataserverB
```

or

```
cfRECV lf:/home/usr/file rf:"c:\temp\test.txt" n:dataserverB
```

Listing Local Profiles

You can list local profiles using the `cfprofile a:list` command.

Procedure

1. Optional: On the command line, navigate to the `$CFROOT/bin` directory.
2. Enter `cfprofile a:list`.
This lists the local profiles defined for TIBCO MFT Platform Server.

Deleting Local Profiles

You can delete local profiles using the `cfprofile a:delete node:nodename luser:username` command.

Procedure

1. Optional: On the command line, navigate to the `$CFROOT/bin` directory.
2. Enter `cfprofile a:delete node:nodename luser:username`.
This deletes a local profile previously defined.

Local Profile Parameters

The following tables list parameters supported for the `cfprofile` command.


Required Local Profile Parameters



If any of the required parameters is not defined and the `prompt` parameter is set to No, the `cfprofile` command fails.

The following table lists the required parameters for the `cfprofile` command.

Parameter (Alternate Specification)	Description
Node (n)	<p>Defines the name of the node with which the local profile is associated.</p> <div> <p>The node name is not case sensitive. The node must already exist so you can successfully add or update a local profile.</p> </div>
Password (p)	Defines the password used to log on to the node (remote system).



Parameter (Alternate Specification)	Description
User (u)	<p>Defines the user name used to log on to the node (remote system).</p> <div>  <p>If the node is a Windows system, the domain must also be specified using either of the following formats: <i>domain\username</i> or <i>domain/username</i>. The maximum length of the defined value is 64 characters.</p> </div>

Optional Local Profile Parameters



The **cfprofile** command does not require any of the optional parameters to be defined if the **prompt** parameter is set to No.

The following table lists the optional parameters for the **cfprofile** command.

Parameter (Alternate Specification)	Description
Action (a)	<p>Defines the action you want to take.</p> <p>The valid values are: Delete, List, or Add.</p>
LocalUser (l luser)	<p>Defines the identity of a different local user in the local system.</p> <p>For example, if user A has defined this parameter, user A can add a user profile for user B without having to log on as user B.</p> <div>  <p>Only the root account or members of the cfadmin group can use this option.</p> </div> <p>When the LocalUser parameter is not defined, the prompt parameter is set to YES, and you log on as the root account or a member of the cfadmin group, you are prompted whether you want to define another local user.</p> <p>If this parameter is set to *ALL, this local profile can be used by all local users who want to perform transfers with the defined node.</p>
Prompt	<p>Defines whether to activate the cfprofile command to an interactive mode.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> YES: cfprofile prompts you for all required information to create or update a local profile. This is the default value. <div>  <p>You are prompted as to whether you want to create cfprofile.cfg when it is not found.</p> </div> <ul style="list-style-type: none"> No: if you do not want to be prompted, you can set this parameter to No.
-? (/?)	Displays online help for the cfprofile command.

Responder Profile

A responder profile defines a local user id that can be used to substitute the incoming user name and password. Each responder profile definition is defined in a clear text file named `cfrprofile.cfg` located in the `$CFROOT/config` directory.

A responder profile contains the following information:

- **Node:** the remote system with which the responder profile is associated.
- **Remote user name:** the user name supplied by the remote system initiating the transfer. It does not have to be a valid user name in the local system.
- **Remote password:** the password supplied by the remote system initiating the transfer. If the remote user is a verified user, this parameter must be set to `*VER` in the responder profile.
- **Local user name:** the local user name used to process a transfer request on your local machine from a specified remote user.

You can add or update responder profiles through the `cfrprofile` command. Before you update any information in `cfrprofile.cfg`, a backup of this file called `cfrprofile.bak` is created. You can activate responder profiles by simply specifying the node for a transfer.

Creating Responder Profiles

By using responder profiles, a remote transfer user does not have to know a local user name and password on your local machine to initiate a transfer. A responder profile is chosen based on the remote user name, password, and the node definition associated with the incoming request. The information in this responder profile is used to log on to the local system.



Only if you are the root account or a member of the `cfadmin` group, you can create profiles for other users.

Procedure

1. Optional: On the command line, navigate to the `$CFROOT/bin` directory.
2. Enter `cfrprofile prompt:YES`.

You are prompted for the required parameters to define a responder profile.



You can use the `cfrprofile` command without being prompted for the required parameters. See the following sample command:

```
cfrprofile n:dataServerA r:abc rp:abc l:johndoe prompt:NO
```

3. On the prompt menu, set the values for the required parameters. See the following sample command:

```
cfrprofile prompt:YES
Enter a valid Node Name: dataServerA
Enter a valid Remote User: abc
Enter a valid Remote Password: abc
Re-enter Remote Password:
Enter a valid Local User: johndoe
```

This example creates a responder profile for the remote user `abc`. This remote user can initiate a transfer request from the `dataServerA` node without having to know the user ID and password in the local system. The transfer is processed on the local machine using the local user ID `johndoe`.

Listing Responder Profiles

You can list responder profiles using the `cfrprofile a:list` command.

Procedure

1. Optional: On the command line, navigate to the `$CFROOT/bin` directory.
2. Enter `cfrprofile a:list`.
This lists in the console the responder profiles defined for TIBCO MFT Platform Server.

Deleting Responder Profiles

You can delete responder profiles using the `cfrprofile a:delete node:nodename ruser:username` command.

Procedure

1. Optional: On the command line, navigate to the `$CFROOT/bin` directory.
2. Enter `cfrprofile a:delete node:nodename ruser:username`.
This deletes a responder profile that was previously defined.

Responder Profile Parameters

The following tables list parameters supported for the `cfrprofile` command.

Required Responder Profile Parameters



If any of the required parameters is not defined and the **prompt** parameter is set to No, the `cfrprofile` command fails.

The following table lists the required parameters for the `cfrprofile` command.



Parameter (Alternate Specification)	Description
Node (n)	<p>Defines the name of the node with which the responder profile is associated.</p> <div> <p>The node name is not case sensitive. The node must already exist so you can successfully add or update a responder profile.</p> </div>
Rpass (rp)	<p>Defines the password supplied by the remote initiator.</p> <p>If this responder profile is associated with a verified user, Rpass must be set to *VER.</p>
Ruser (r)	<p>Defines the user name supplied by the remote initiator.</p> <div> <p>The maximum length of the defined value is 64 characters. If the remote user is located on a mainframe, this parameter cannot contain more than 8 characters.</p> </div>

Optional Responder Profile Parameters



The **cfprofile** command does not require any of the optional parameters to be defined if the **prompt** parameter is set to No.

The following table lists the optional parameters for the **cfprofile** command.

Parameter (Alternate Specification)	Description
LocalUser (1 luser)	<p>Defines the local user used to process the transfer request initiated by the defined remote user.</p> <p> Only members of the cfadmin group or the root user can define this parameter. Otherwise, this parameter is automatically set to the current user.</p>
Action (a)	<p>Defines the action you want to take.</p> <p>The valid values are: Delete, List and Add.</p>
Prompt	<p>Defines whether to activate the cfprofile command to an interactive mode.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • YES: cfprofile prompts you for all required information to create or update a user profile. This is the default value. <p> You are prompted as to whether you want to create cfprofile.cfg when the file is not found.</p> <ul style="list-style-type: none"> • No: if you do not want to be prompted, you can set this parameter to No.
-? (/?)	Displays online help for the cfprofile command.

Transfer Using Distribution Lists

With distribution lists, you can use a single command to send a single or multiple files to multiple destinations.

In a distribution list, you can define multiple nodes and the default directory to which you want to perform **cfsend** transfers.

Configuring Distribution Lists

You can configure the `cflist.cfg` file located in the `$CFROOT/config` directory to define distribution lists. When you use a distribution list for **cfsend** transfers, you can specify a single destination for multiple nodes or specify different destinations for different nodes. The host connection information is retrieved from your node configurations.



You can only use distribution lists when you perform a send transfer.

Procedure

1. On the command line, navigate to the `$CFROOT/config` directory .
2. Open the `cflist.cfg` file with any text editor.
3. Set the values for the required parameters and save the file.


See the following two sample distribution lists included in the default `cflist.cfg` file:

```
[AccList]
# Distribution list : Acclist
Node=Store1, Store2,
Directory = /tmp/prod/data
Node=Store5

[Stores]
# Distribution list : Stores
Node=Linux
Node=Windows, windowsWM
Directory=c:/tmp/yz
```

Distribution List Parameters

The following tables list required parameters for the distribution lists.

Parameter	Description
<i>distribution_list_name</i>	<div>Defines the name of the distribution list.</div> <div>This is a required parameter. Specify the distribution list name between square brackets. It can be from 1 to 32 characters and cannot contain any spaces. Any name longer than 32 characters is truncated.</div> <div> The pound sign (#) cannot be used within a distribution list name and it has special meaning in the UNIX environment.</div>

Parameter	Description
Node	<p>Defines a single or multiple nodes to conduct transfer requests with when this distribution list is used.</p> <p>This is a required parameter. Multiple nodes defined on 1 line must be delimited by a comma.</p>
Directory	<p>Defines a destination directory on the specified node.</p> <p>If no directory is specified, then the directory defined on the command line is used. However, if a directory is defined in the distribution list, it overrides a directory that is defined in the transfer window or on the command line.</p>

Examples: Transfer Using Distribution Lists

In the following example, AccList is a distribution list defined in the `cflist.cfg` file. The example presupposes that each node defined in AccList has the same user ID and password in each system.



If a node has a different user ID and password, a local profile must be configured. Local profiles can be configured locally for each node to map the local user who initiates the file transfer to the correct remote user ID and password. Alternatively, the administrator of the remote server can configure responder profiles to map the incoming user ID.

```
cfsend list:AccList lf:/home/user/file
rf:"c:\temp\test.txt"
```

You can define other optional transfer parameters for distribution lists. For more information, see [StopOnFailure](#) and [RunPPAEndDirTx](#).

Transfer Using Templates

You can perform file transfers by defining all required transfer parameters and options in transfer templates.

Using templates is a method to save your time. After creating a template, specify the **Template | t** parameter in the **cfsend** or **cfrecv** command.

The format for using a template is as follows:

- `cfsend t:TemplateName`
- `cfrecv t:TemplateName`

Two sample templates called TSEND and TRECVC are installed and located in the `$CFROOT` directory by default. For more details of the sample templates, see [Sample Templates: TSEND and TRECVC](#).

In a template, by specifying the **TransferType (trtype)** parameter, you can perform the following types of transfers:

- You can simply send or receive a file. To perform such transfers, specify the **TransferType | trtype** parameter as F; or you do not have to set the **TransferType** parameter because this is the default transfer type.
- You can send or receive an executable file, and run the file as a job. To perform such transfers, set the **TransferType | trtype** parameter as J.



The **LocalFileName** and the **RemoteFileName** parameter in the template are to define the name of the executable file.

- You can send a command, and run it in a remote system. To perform such transfers, set the **TransferType | trtype** parameter as C.



If the remote system is Windows or UNIX, the **LocalFileName** parameter in the template is to define the name of the file to place command output. The **RemoteCommand** parameter is to define the command you want to use in the remote system.

- You can send a file, and run it as a print job. To perform such transfers, set the **TransferType | trtype** parameter as P.



The **LocalFileName** parameter is to define name of the file to print. The **RemoteFileName** parameter is to define the printer name on the remote system.

When using the **cfsend** or **cfrecv** command with a template, you can also specify other transfer parameters on the command line. The parameters specified on a command line take higher precedence over parameters specified in a template.

See the following examples for your reference:

- `cfsend t:TSEND ip:10.1.1.130`

In this example, the file defined in TSEND is the file sent to a file to the IP address 10.1.1.130, even if another IP address has been defined in the TSEND template.

- `cfsend t:TSEND n:dataServerB`

In this example, the file defined in TSEND are sent to the dataServerB node, even if another node has been defined in TSEND template.



If you specify both a node and a template on a command line, the parameters specified in a node can override parameters specified in a template. As an exception, the **IpName/Address** and **Port** parameters specified in a template can override the **HostName|h** and **Port|p** parameters specified in a node. Therefore under this circumstance, it is good practice to mark the **IpName/Address** and **Port** parameters as comments in the template.

Sample Templates: TSEND and TRECV

Use the sample templates as a guide to create your own templates.

For more information on configuring template parameters, see [Transfer Parameters](#).

You can add the number sign (#) at the beginning of a line to make the line a comment.

See the following example of the TSEND template:

```
# Sample template file for cfsend command

LocalFileName:          /home/user/file
RemoteFileName:         c:\tmp\unix.txt
# ConfigFileName:       /mftps/config/config.txt
# IpName/Address:       127.0.0.1
Node:                   N                               { N, Node Name }
Port:                   46464
UserId:                 uid                             { *VER }
Password:               pwd

# Additional File Transfer Parameters
CR_LF:                  Y                               { CRLF|Y,LF,N,CRLFY }
ASCII_to_EBCDIC:        N                               { N|Binary, Y|Text, A|Ascii }
ConvTbl:                N                               { N, FileName }
CreationOption:         CR                             { C,R,A,CR,CA,CRN }
TryNumber:               1                             { N|1, 0|U|Unlimited, 2 - 10 }
RetryInterval:          N                               { N, Y|1, # of min more than 1 }
CheckPointInterval:     N                               { N, Y|1, # of min more than 1 }
Compression:            N                               { N, RLE|Y, LZ, ZLIB1-9 }
EncryptionType:         N { N,DES,3DES,BlowFish|BF,BlowFishLong|BFL,Rijndael|RIJN|RJ|
AES,AES128 }
LocalCTFile:            N                               { N, NONE, FileName }
RemoteCTFile:           N                               { N, FileName }
TLS:                    N                               { T, Y|S, N }
TLSPort:               N
TransferType:           F                               { F|File, J|Job, C|Command, P|Print }
RemoteCommand:          N                               { N, Command to be executed }
RemotePrinterName:      N                               { N, printer name }
ProcessName:            N                               { N, string, $(TIME) }
UserData:               N                               { N, string }
ExitPrgm:               N                               { N, FileName }
SecurityAttribTemplate: N                               { N, any name }
PermittedActions:       N                               { N; E,Z,S,H,A,R,C }
UnixPermissions:        N                               { N, 3 digit number }
EmailSuccess:           N                               { N, email address }
EmailFailure:           N                               { N, email address }
Post_Action1:           N                               { N, parameters }
Post_Action2:           N                               { N, parameters }
Post_Action3:           N                               { N, parameters }
Post_Action4:           N                               { N, parameters }
SilentMode:             N                               { N, Y }
Timeout:                120                             { Transfer timeout in min }
ClassOfService:         Default                         { from cfcos.cfg }
CRC:                    N                               { N, Y }
DirTransfer0Files:      F                               { Failure|F, Success|S }

# Additional Directory\List Transfer Parameters
ScanSubDir:             N                               { N, Y }
StopOnFailure:          Y                               { N, Y }
Test:                   N                               { N, Y }
DistributionList:       N                               { N, List Name }

# Additional Accelerator Transfer Parameters
Accelerate:             N                               { N, Y }
ACCProtocol:            PDP                             { TCP, UDP, PDP }
ACCEncryption:          N                               { N, Y }
ACCCompression:         N                               { N, Y | Best, Default, Fast }
```

```

ACCMaxSpeed:          1000000          { 256 - 1000000 kbps }
ACCHost:              N                { N, Host }
ACCPort:              N                { N, Port }

# Optional Parameters For zOS Transfers follow:
# In Order For Them To Take Effect, Set zOS Parameter to Y.
zOS:                  N                { Y, N }
DELIM:                N                { CRLF|Y,LF,N,CRLFY }
REMOVETRAIL:          N                { Y, N }
RECFM:                FB              { F,FB,VB,V,U,FBA,FA,FBM,FM,VBA,VA,VBM,VM }
LENGTH:              80              { 1 - 32760 }
BLKSIZE:              0               { 0 - 32760 }
ALLOC_TYPE:           K               { T,C,M,K }
ALLOC_PRI:            0               { 0 - 32000 }
ALLOC_SEC:            0               { 0 - 32000 }
VOLUME:               N               { N, VolumeName }
UNIT:                 SYSALLDA        { N, UnitName }
AVAIL:                Immediate       { Immediate|I, Deferred|D }
EXEC:                 N               { N, OS390 Command to be executed }
CALLJCL:              N               { N, OS390 program be called }
CALLPROG:             N               { N, OS390 program be called }
SUBMIT:               N               { N, OS390 JCL to be submitted }
DATACLASS:            N               { N, DataClass }
MGMTCLASS:            N               { N, MgtClass }
STORCLASS:            N               { N, StorClass }
RetenPeriod_ExpDate:  N               { N, # of days, yyyy/ddd }
SysOutClass:          N               { N, SysOutClass }
SysOutFcb:            N               { N, SysOutFcb }
SysOutForms:          N               { N, SysOutForms }
SysOutCopies:         N               { N, SysOutCopies }
SysOutWriter:         N               { N, SysOutWrites }
SysOutDestination:    N               { N, SysOutDestination }
SysOutUserName:       N               { N, SysOutUserName }
MaintainBDW:          N               { Y, N }
MaintainRDW:          N               { Y, N }
Truncate              Y               { Y, N, W }
UTF8BOM               N               { A, R, B, N }

```

See the following example of the TREC template:

Sample template file for cfrcv command

```

LocalFileName:        /home/user/file
RemoteFileName:        c:\tmp\unix.txt
# ConfigFileName:      /mftps/config/config.txt
# IpName/Address:      127.0.0.1
Node:                  N                { N, Node Name }
Port:                  46464
UserId:                uid              { *VER }
Password:              pwd

# Additional File Transfer Parameters
CR_LF:                 Y                { CRLF|Y,LF,N,CRLFY }
ASCII_to_EBCDIC:       N                { N|Binary, Y|Text, A|Ascii }
ConvTbl:               N                { N, FileName }
CreationOption:         CR              { C,R,A,CR,CA,CRN }
TryNumber:              1               { N|1, 0|U|Unlimited, 2 - 10 }
RetryInterval:          N               { N, Y|1, # of min more than 1 }
CheckPointInterval:     N               { N, Y|1, # of min more than 1 }
Compression:           N               { N, RLE|Y, LZ, ZLIB1-9 }
EncryptionType:         N { N,DES,3DES,BlowFish|BF,BlowFishLong|BFL,Rijndael|RIJN|RJ|
AES,AES128 }
LocalCTFile:           N                { N, NONE, FileName }
RemoteCTFile:          N                { N, FileName }
TLS:                   N                { T, Y|S, N }
TLSPort:               56565
TransferType:           F                { F|File, J|Job, C|Command, P|Print }
ProcessName:           N                { N, string, $(TIME) }
UserData:              N                { N, string }
ExitPrgm:              N                { N, FileName }

```

```

UnixPermissions:      N                                { N, 3 digit number }
EmailSuccess:         N                                { N, email address }
EmailFailure:         N                                { N, email address }
Post_Action1:         N                                { N, parameters }
Post_Action2:         N                                { N, parameters }
Post_Action3:         N                                { N, parameters }
Post_Action4:         N                                { N, parameters }
SilentMode:           N                                { N, Y }
Timeout:              120                             { Transfer timeout in min }
ClassOfService:       Default                         { from cfcos.cfg }
CRC:                  N                                { N, Y }
DirTransfer0Files:    F                                { Failure|F, Success|S }

# Additional Directory Transfer Parameters
ScanSubDir:           N                                { N, Y }
StopOnFailure:        Y                                { N, Y }
Test:                 N                                { N, Y }

# Additional Accelerator Transfer Parameters
Accelerate:           N                                { N, Y }
ACCProtocol:          PDP                             { TCP, UDP, PDP }
ACCEncryption:        N                                { N, Y }
ACCCompression:       N                                { N, Y | Best, Default, Fast }
ACCMaxSpeed:          1000000                         { 256 - 1000000 kbps }
ACCHost:              N                                { N, Host }
ACCPort:              N                                { N, Port }

# Optional Parameters For zOS Transfers follow:
# In Order For Them To Take Effect, Set zOS Parameter to Y.
zOS:                  N                                { Y, N }
DELIM:                LF                             { CRLF|Y,LF,N,CRLFY }
REMOVETRAIL:          N                                { Y, N }
UTF8BOM               N                                { A, R, B, N }

```

Transfer Parameters



You must supply the required parameters when you run the `cfsend` or `cfrecv` command.


See the following list of different types of parameters you can use.

- [Minimum Transfer Parameters](#)
- [Optional Transfer Parameters](#)
- [z/OS Specific Transfer Parameters](#)
- [TIBCO Accelerator Transfer Parameters](#)

Minimum Transfer Parameters

You must supply at least the following parameters when you run the `cfsend` or `cfrecv` command.


Parameter (Alternate Specification)	Description
LocalFileName (lf)	Defines the name of the local file you want to transfer.
Node (n)	<p>Defines the name of the node to or from which you perform the transfer.</p> <p>The valid values are <code>N</code> or the name of the node.</p> <p>For more information, see Transfers Using Nodes.</p>
Password (pwd)	<p>Defines the password for the remote user ID.</p> <p>This password is used by the remote system to validate if the user has credentials to access the remote file.</p> <div>  <p>This field must be left blank when you define verified users.</p> </div>
RemoteFileName (rf)	<p>Defines the name of the file or data set at the remote location.</p> <div>  <p>When specifying the remote file name on a command line:</p> <ul style="list-style-type: none"> • If a backslash (\) is used in the file name, such as in a file transfer to a Windows system, you must enclose the remote file name in double quotation marks (""). • If you use a member name, such as in a data set transfer to a mainframe system, you must enclose the entire data set in double quotation marks (""). Otherwise, you must use backslashes before the parentheses (). </div>


Parameter (Alternate Specification)	Description
UserId (uid)	<p>Defines the remote user ID for the file transfer.</p> <div>  <p>This ID must exist in the remote system and have permission to access a remote file.</p> </div> <p>For Windows security systems, you can use the format <i>domain \username</i> or <i>domain/username</i>. If this parameter is not specified, TIBCO MFT Platform Server treats the user as a verified user (*VER).</p>




Optional Transfer Parameters



You can supply the optional transfer parameters as you want when you run the **cfsend** or **cfrecv** command.


Parameter (Alternate Specification)	Description
UTF8BOM (bom)	<p>Defines if files need to be encoded in UTF-8.</p> <p>The default value is N.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • A: add BOM. • R: remove BOM. • B: add/remove BOM. • N: None.
CheckpointInterval (cpint)	<p>Defines how often a checkpoint is taken.</p> <p>The default value is N. If the transfer fails after taking a checkpoint, the transfer continues from that last checkpoint.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • N: no checkpoint. • Y 1: the checkpoint is taken every minute. • 2 to 90: the checkpoint is taken at this specified interval.
ClassOfService (cos)	<p>Defines the buffer size for the platform server in a transfer.</p> <p>The valid values are the different levels of SBUFSIZE and RBUFSIZE specified in the configuration file named cfcos.cfg. The transfer performance can be improved as server and client using different level of classes.</p>





Parameter (Alternate Specification)	Description
Compression (cmp)	<p>Defines the type of compression you want to use. We strongly suggest using ZLIB or RLE compression. LZ compression uses a lot of CPU and the compression results are not as good as with ZLIB.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • R Y: Run Length Encoding (RLE) compression. • LZ: Lempel-Zev (LZ) compression. • ZLIB1 through ZLIB9: levels of zlib compression. <p>Level 1 is fast but provides the lowest ratio of compression. Level 7 to 9 produce the best quality of compression, but are much slower. ZLIB2 typically provides the best balance between compression and speed.</p> <ul style="list-style-type: none"> • N: no default compression. • NEVER: never uses any compression.
ConfigFileName (cf)	<p>Defines the path of the configuration file on the local machine.</p> <p>The valid values are any valid paths.</p> <div>  This parameter is normally left unspecified. </div>
CONVTBL(ct)	<p>Defines the path of the conversion table.</p> <p>This parameter is not used if the LocalCTFile parameter is specified.</p> <p>For more information, see Conversion Tables/Custom Code Conversion.</p>




Parameter (Alternate Specification)	Description
CR_LF (crlf)	<p>Defines the carriage return (CR) and line feed (LF) control for transfers using the cfsend or cfrecv command between UNIX and Windows.</p> <p> When you perform transfers to or from z/OS, you can use the DELIM parameter.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • Y CRLF: CR is deleted when you receive a file on UNIX. CR is added before the LF (line feed) when you send a file on UNIX. • L LF: records are delimited by LF. This is typically used when you transfer text data to z/OS. Note that line conversion is performed on z/OS. No processing is performed by TIBCO MFT Platform Server for UNIX. • CRLFY: CR is not added to LF when you send a file on UNIX. Likewise, CR is not removed when you receive a file on UNIX. This applies to the rare case when a UNIX file contains CRLF, or if the application requires CRLF instead of LF. • N: no record delimiter is applied in the file. This typically applies for a binary transfer.
CreationOption (co)	<p>Defines the remote file creation options.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • R: replaces an existing file. This only works when the remote file already exists. • A: appends to an existing file. This only works when the remote file already exists. • C: creates a new file at the remote location. This only works when the remote file does not exist. • CR X: creates a new file or replace an existing file. This is the default option. • CA Y: creates a new file or append to an existing file. • CRN Z: creates a new file, and if necessary create the directory path to this file or replace an existing file.
DirTransfer0Files (d0)	<p>Defines the status of the directory transfer when zero files are in the directory.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • F: When no files are transferred on a directory transfer, the transfer is a failed transfer. • S: When no files are transferred on a directory transfer, the transfer is a successful transfer. <p>The default value is F.</p>


Parameter (Alternate Specification)	Description
EmailFailure (emf)	<p>Defines the email address or addresses to which TIBCO MFT Platform Server sends a message when a transfer fails. You can specify multiple email addresses by delimiting the email addresses with a comma.</p> <div>  <p>In a directory transfer, when EmailFailure is specified, if any of those file transfers fails, an individual email is sent for that failed transfer and a summary email is sent when the directory transfer is completed. If all the files within the directory are transferred successfully, nothing is sent.</p> <p>Ensure your email settings are configured in the <code>config.txt</code> file.</p> </div>
EmailSuccess (ems)	<p>Defines the email address or addresses to which TIBCO MFT Platform Server sends a message when a transfer is successful. You can specify multiple email addresses by delimiting the email addresses with a comma.</p> <div>  <p>In a directory transfer, when EmailSuccess is specified, if all the files within the directory are transferred successfully, only a summary email is sent. If any of those file transfers fails, nothing is sent.</p> <p>Ensure your email settings are configured in the <code>config.txt</code> file.</p> </div>
EncryptionType (en)	<p>Defines the type of encryption that is used for the transfer.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • N 0: no encryption • DES 1 : DES encryption • 3DES 2 : triple DES encryption • Blowfish BF 3: Blowfish encryption • BlowfishLong BFL 4: Blowfish Long (448 Bit) encryption • AES Rijndael RIJN RJ 5: AES 256 bit encryption • AES128: AES 128 encryption <div>  <p>AES128 is not supported for products with earlier versions.</p> </div>
ExitPrgm (ep)	<p>Defines the path to the exit program on the local machine.</p> <p>You can use the exit program to customize post processing.</p> <p>For more information, see User Exits.</p>
IpName/Address (ip)	<p>Defines the host name or the IP address of the remote system.</p>


Parameter (Alternate Specification)	Description
LIST (l)	<p>Defines the distribution list you want to use for the transfer request.</p> <p>The maximum length of the defined value is 32 characters.</p> <p>For more information, see Distribution Lists.</p>
LocalCTFile (lct)	<p>Defines the name of the local conversion table file.</p> <div data-bbox="699 470 740 516"></div> <p>You must set the ASCII_to_EBCDIC parameter to Y to use this parameter.</p> <p>For more information, see Conversion Tables/Custom Code Conversion.</p>
PermittedActions (pa)	<p>Defines the permitted actions for the transferred files.</p> <div data-bbox="699 701 740 747"></div> <p>The following values are Windows specific and are only valid when you transfer files to Windows.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • S SYSTEM_FILE: a system file that can only be viewed by the operating system and not the user. • H HIDDEN_FILE: a hidden file that cannot be seen by the user. • R READ_ONLY: a file that can only be viewed by the user. Users cannot modify the file. • C NTFS_COMPRES: a file that can be compressed in the remote system. This option is only valid on NTFS partitions. Otherwise, it is ignored. • Z EOF_CRLF: with this option, a CR/LF (0x0d, 0x0a) is appended to the end of the file, followed by the DOS End of File character, Control Z (0x1a). If a trailing Control Z or CR/LF already exists, they are not added again. This option is only valid when CR/LF processing is enabled. • E EOF: with this option, a DOS End of File character, Control Z (0x1a), is appended to the file.
Port	<p>Defines the IP port on which TIBCO MFT Platform Server listens for incoming requests.</p> <p>The valid values are from 1024 to 65535.</p>

Parameter (Alternate Specification)	Description
Post_Action (ppa) Post_Action1 (ppa1) Post_Action2 (ppa2) Post_Action3 (ppa3) Post_Action4 (ppa4)	<p>Defines the command you want to use upon the completion of a transfer.</p> <p>You can define this command up to four times with the following command:</p> <pre>Post_Action="S F,L R,COMMAND,command data"</pre> <p>Where:</p> <p>S F means success or failure.</p> <p>L R means local or remote.</p> <p>COMMAND refers to the command you want to use after the file transfer is completed. This is the only option currently supported by a UNIX responder.</p> <p><i>command data</i> refers to the absolute path and file name of the command and any parameters to be used. This is limited to 256 bytes.</p> <p>You cannot use any spaces in the Post_Action command. If the remote system is a mainframe, then CALLJCL, CALLPGM, and SUBMIT parameters are also supported besides COMMAND. For more information, see <i>TIBCO Managed File Transfer Platform Server for z/OS User's Guide</i>.</p> <p>If you transfer files to TIBCO MFT Platform Server for Windows:</p> <ul style="list-style-type: none"> You can append a # sign to the end of the entered data: TIBCO MFT Platform Server for Windows launches the PPA and waits for the return code of the action. You can append a & sign to the end of the data entered: TIBCO MFT Platform Server for Windows launches the PPA and does not wait for the action to finish. This is the default behavior. <p>For more information, see Post Processing Actions.</p>
ProcessName (pn)	<p>Defines the process name used for the file transfer.</p> <p>The maximum length of the defined value is 8 characters.</p>
RemoteCommand (rcmd)	<p>Defines the command you want to use in the remote system.</p> <p>The default value is No. The valid values are N or the command you want to use.</p>
RemoteCTFile (rct)	<p>Defines the name of the remote conversion table file.</p> <div>  <p>When defining this parameter, you have to set the LocalCTFile as NULL, to ensure no translation takes place locally.</p> </div> <p>For more information, see Conversion Tables/Custom Code Conversion.</p>
RemotePrinterName (rp)	<p>Defines the name of the remote printer to which you send the job.</p>

Parameter (Alternate Specification)	Description
RetryInterval (ri)	<p>Defines the interval in minutes after which a failed transaction can be retried.</p> <p> This parameter only applies when the value of the TryNumber parameter is greater than 1.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • N: the failed transaction is retried immediately. • Y 1: the failed transaction is retried after one minute. • <i>number_of_minutes</i> (more than 1): the failed transaction is retried after this defined interval.
ScanSubDir (ssd)	<p>Defines whether all subdirectories from the file path are scanned.</p> <p> This parameter only applies to directory transfers.</p> <p>The valid values are Y or N.</p>
SecurityAttribTemplate (sa)	<p>Defines the file that the remote Windows platform uses as a template for Access Control List (ACL).</p> <p>The ACL of this file is copied to the ACL of the destination file.</p> <p> For the access control feature to function properly on Windows, the file specified must be readable by the partner that receives the file to file transfer, and the created file must be located on an NTFS drive.</p>
SilentMode (sm)	<p>Defines whether the progress message (in bytes) is displayed on the output screen on the initiator side.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • Y: the progress message (in bytes) is not displayed on the output screen on the initiator side. • N: the progress message (in bytes) is displayed along with the typical output on the screen. This is the default value. <p> You can best observe the transmission progress in bytes when the file you transfer is large.</p>
TLS/SSL	<p>Defines whether you use TLS/SSL communication for transfers.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • N : does not use SSL/TLS Encryption • Y: uses TLS/SSL encryption • T : uses TLS tunnel encryption <p>For more information, see TLS/SSL Certificates Setup.</p>

Parameter (Alternate Specification)	Description
SSLPort (tport , sport , tlsport)	<p>Defines the port that MFT Platform Server uses to connect to the target system when TLS or tunnel requests are initiated.</p> <p>Valid values are from 1024 to 65535 and must match the port that the target server is listening on.</p> <p>This parameter should be used when the TLS parameter is set to Y (TLS Mode) or T (Tunnel Mode).</p> <p>If TLS=Y or TLS=T is defined and this parameter is not set, Platform Server will use the port defined by the port parameter.</p> <p>If TLS=Y is specified and port, sport and tport are not defined, port 56565 will be used.</p> <p>If TLS=t is specified and port, sport and tport are not defined, port 58585 will be used.</p> <p>For more information, see TLS/SSL Certificates Setup.</p>
StopOnFailure (sonf)	<p>Defines whether to stop transferring the rest of files in the directory when the current file transfer fails.</p> <div data-bbox="699 890 740 936"></div> <p>This parameter applies to directory transfers and distribution list transfers.</p> <p>The valid values are Y or N.</p>
Template (t)	<p>Defines the file name of the transfer template.</p> <div data-bbox="699 1087 740 1134"></div> <p>This parameter cannot be set inside a transfer template.</p> <p>For more information, see Transfers Using Templates.</p>
Test	<p>Defines whether to display the local and remote file names to verify if the file names are correct, instead of doing the actual transfers.</p> <div data-bbox="699 1304 740 1350"></div> <p>This parameter applies to directory transfers and distribution list transfers.</p> <p>The valid values are Y or N.</p>
Timeout (to)	<p>Defines the amount of time in minutes a connection stays open when waiting for a response from the remote side.</p> <p>Once this amount of time is reached, the connection ends.</p>

Parameter (Alternate Specification)	Description
TransferType (trtype)	<p>Defines the type of transfer you want to perform.</p> <p>The default value is <code>File</code>.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • <code>F File</code>: you can send your local file to the remote file. • <code>J Job</code>: you can send your local file to the remote system. The remote server runs it and sends you an error message if the running fails. You can receive the remote file and run it on your side, and get an error message if running fails. • <code>C Command</code>: you can send the command to the remote system and receive a result. If you specify LocalFileName, the result of the command is saved there, otherwise the result is printed out. If the remote command fails, you receive an error message with a return code describing the reason of the failure. <div data-bbox="738 779 781 827"></div> <div data-bbox="850 779 1482 842">The <code>C Command</code> option only works with the cfsend command.</div> <ul style="list-style-type: none"> • <code>P Print</code> : you can send a local file to a remote system. The partner executes the file as a print job. <p>For more information, see Transfer Commands.</p>
Truncate (trunc)	<p>The valid values are:</p> <ul style="list-style-type: none"> • <code>YES</code>: when a record longer than the z/OS LRECL is received, the record is truncated. • <code>NO</code>: when a record longer than the z/OS LRECL is received, the transfer is terminated with an error. • <code>WRAP</code>: when a record longer than the z/OS LRECL is received, the record is split into multiple records.
TryNumber (trynum)	<p>Defines the number of times the transfer can be attempted.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • <code>0 U Unlimited</code>: the transfer can be attempted 9999 times or until it is successful. It restarts the transfer from the beginning unless the Checkpoint/Restart option is set. • <code>N 1</code>: the transfer can be attempted only one time. The default value is 1. • <code>2 to 9998</code>: the transfer can be attempted for the specified number of times.

Parameter (Alternate Specification)	Description
UNIXPermissions (uperm)	<p>Defines the UNIX permissions for the file.</p> <p>When a file is created in UNIX, TIBCO MFT Platform Server can set the UNIX permissions on the file. UNIX permissions are defined by a three digit number such as 777 (the same as the chmod command).</p> <p>The default value for this parameter is the file permissions of the file transferred. This parameter works differently for a send transfer than a receive transfer.</p> <p>If a send transfer is initiated and the UNIXPermissions parameter is defined, this value is passed to the remote system. If this parameter is not defined, the permissions of the file can be in the control record. If no values are passed in the control record, the responder uses the system default permissions.</p> <div>  <p>You can only set up the permissions for the file when the file is created. For example, UNIXPermissions works only with the Create, CreateReplace and CreateReplaceNew options when the file is created.</p> </div>
UserData (ud)	<p>Defines the description for the transfer in the local and remote system.</p> <p>This is 25 character field for user comments, and it can contain any alphabetic, numeric, or national characters.</p>


z/OS Specific Transfer Parameters



You must supply z/OS specific transfer parameters when you perform transfers with a z/OS system.







To use any z/OS specific transfer parameters, set the parameter **zOS** to Y on the command line, otherwise all z/OS specific transfer parameters are ignored.





Parameter (Alternate Specification)	Description
ALLOC_PRI (ap)	<p>Defines the quantity of the remote file primary allocation.</p> <p>The valid values are from 0 to 32000.</p> <p>TIBCO MFT Platform Server supports automatic assignment for ALLOC_PRI when ALLOC_TYPE is set to M or K. If you set this value to zero, then the appropriate number of megabytes or kilobytes are assigned respectively.</p>
ALLOC_SEC (as)	<p>Defines the quantity of the remote file secondary allocation.</p> <p>The valid values are from 0 to 32000.</p> <p>TIBCO MFT Platform Server supports automatic assignment for ALLOC_SEC when ALLOC_TYPE is set to M or K. If you set this value to zero, then the appropriate number of megabytes or kilobytes are assigned respectively.</p>

Parameter (Alternate Specification)	Description
ALLOC_TYPE (at)	<p>Defines the type of remote file allocation.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • T: data set size is allocated in tracks. • C: data set size is allocated in cylinders. • M: data set size is allocated in megabytes. • K: data set size is allocated in kilobytes.
ASCII_to_EBCDIC (eb)	<p>Defines the type of data translation that is required for the remote system.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • Binary N 0: the file is binary and does not require any translation. • ASCII A 1: the file is ASCII and does not require translation, but can require CR/LF (carriage return, line feed) insertion. • Text Y 2: the file is ASCII and the remote system requires EBCDIC, the data is translated by TIBCO MFT Platform Server for UNIX. This value is typically used for transfers to a z/OS system. <div>  <p>This parameter is used when sending files from UNIX to z/OS.</p> </div>
AVAIL (da)	<p>Defines the remote file volume availability.</p> <p>The valid values are I Immediate or D Deferred.</p>
BLKSIZE blocksize (obs)	<p>Defines the remote file block size.</p> <p>The valid values are from 0 to 32760.</p>
CALLJCL (cj)	<p>Defines whether to call any z/OS program with JCL linkage.</p> <p>The valid values are N or z/OS program.</p>
CALLPROG (cp)	<p>Defines whether to call any z/OS program.</p> <p>The valid values are N or z/OS program.</p>

Parameter (Alternate Specification)	Description
DATACLASS (dc)	<p>Defines the z/OS data class as specified in the Data Facility/System Managed Storage.</p> <p>You can use the Data Facility/System Managed Storage to indicate the media type of the host file, the backup, restore, and archive policies of the installation.</p> <p>The maximum length of the defined value is 8 characters, it can contain numeric, alphabetic, or national characters (\$, #, @).</p> <div>  <p>The first character must be an alphabetic or national character.</p> </div> <p>In addition, you can use this parameter to indirectly select file attributes such as record format and logical record length.</p>
DELIM (cr)	<p>Defines the file delimiter.</p> <div>  <p>This parameter is only valid when the remote system is a z/OS system. If you perform transfers to a Windows system, use the CR_LF parameter.</p> </div> <p>The valid values are:</p> <ul style="list-style-type: none"> • Y CRLF: CR (carriage return) is deleted when you receive a file on UNIX. CR is added before the LF (line feed) when you send a file on UNIX. • L LF: records are delimited by LF. This is typically used when you transfer text data to z/OS. Note that the line conversion is performed on z/OS. No processing is performed by TIBCO MFT Platform Server for UNIX. • CRLFY: CR is not added to LF when you send a file on UNIX. Likewise, CR is not removed when you receive a file on UNIX. This applies in rare cases when a UNIX file contains CRLF, or if the application requires CRLF instead of LF. • N: no record delimiter is applied in the file. This typically applies for transfers of binary files.
EXEC REXXEXEC (re)	Defines the z/OS command that you want to use.
LENGTH (orl lrecl)	<p>Defines the remote file record length.</p> <p>The valid values are from 1 to 32760.</p>
MaintainBDW (mbdw)	<p>Defines whether to maintain the Block Descriptor Word (BDW) when sending or receiving variable block binary files to z/OS.</p> <p>If the data being sent or received is not in the proper BDW format, the transfer will fail.</p>
MaintainRDW (mrdw)	<p>Defines whether to maintain the Record Descriptor Word (RDW) when sending or receiving variable block binary files to z/OS.</p> <p>If the data being sent or received is not in the proper RDW format, the transfer will fail.</p>



Parameter (Alternate Specification)	Description
MGMTCLASS (mc)	<p>Defines the z/OS management class as specified in the Data Facility/System Managed Storage.</p> <p>You can use the Data Facility/System Managed Storage to indicate the media type of the host file, the backup, restore, and archive policies of the installation.</p> <p>The maximum length of the defined value is 8 characters, it can contain numeric, alphabetic, or national characters (\$, #, @). The first character must be an alphabetic or national character.</p>
RECFM (or f)	<p>Defines the remote file record format.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • F: Fixed • FA: Fixed ASA • FB: Fixed Blocked • FBA: Fixed Blocked ASA • FBM: Fixed Blocked Machine • FM: Fixed Machine • V: Variable • VA: Variable ASA • VB: Variable Blocked • VBA: Variable Blocked ASA • VBM: Variable Blocked Machine • VM: Variable Machine • U: Undefined <p>The A extension indicates the use of ASA control characters on z/OS, and the M extension indicates the use of machine control characters on z/OS.</p>


Parameter (Alternate Specification)	Description
RetenPeriod_ExpDate (rp_ed)	<p>Defines the retention period or expiration date of the file in the remote system.</p> <p>The format of the entered value determines whether the parameter is used as a retention period or as an expiration date.</p> <p>The retention period is the number of days, after which the file expires. Expiration date is the date, in Julian format, when the file expires.</p> <p>This parameter is typically used on the z/OS platforms for tape processing to prevent a tape from being overwritten. This parameter must be carefully defined with a disk file. The default is no expiration date on the file.</p> <p>The valid values are: N, a number of days up to 9999, or <i>yyyy/ddd</i>.</p> <div>  <p>This parameter is only supported for send transfers to a z/OS system.</p> </div>
REMOVETRAIL (rmtrail)	<p>Defines whether to remove all trailing spaces and nulls before you transfer the file.</p> <p>The valid values are Y or N.</p> <div>  <p>This parameter is only valid when you receive a file using the cfrecv command.</p> </div>
STORCLASS (sc)	<p>Defines the z/OS storage class as specified in the Data Facility/System Managed Storage.</p> <p>You can use the Data Facility/System Managed Storage to indicate the media type of the host file, the backup, restore, and archive policies of the installation.</p> <p>The maximum length of the defined value is 8 characters, it can contain numeric, alphabetic, or national characters (\$, #, @).</p> <div>  <p>The first character must be an alphabetic or national character.</p> </div>
SUBMIT (sj)	<p>Defines the z/OS JCL you want to submit.</p> <p>The valid values are N or a file that contains the z/OS JCL.</p>
SysOutClass (s1)	<p>Defines the class to which the JES output is routed.</p> <p>On z/OS systems, the printer queues are organized around a printer class instead of a specific printer. This class has a one-character name that is either alphabetic or numeric.</p>
SysOutCopies (sp)	<p>Defines the number of copies to print of a particular report on a remote computer.</p> <div>  <p>This parameter is only valid when the remote platform is z/OS.</p> </div>

Parameter (Alternate Specification)	Description
SysOutDestination (sd)	<p>Defines the destination for the job that you submit to the z/OS internal reader.</p> <div>  <p>This parameter is only valid when the remote platform is z/OS.</p> </div>
SysOutFcb (sb)	<p>Defines the name of the form control buffer as specified in JES.</p> <div>  <p>This parameter is only valid when the remote platform is z/OS.</p> </div>
SysOutUserName (si)	<p>Defines the user name assigned to a job that you submit to the z/OS internal reader.</p> <div>  <p>This parameter is only valid when the remote platform is z/OS.</p> </div>
SysOutWriter (sw)	<p>Defines the external writer name that you use to process this printer file on z/OS.</p> <p>This is the name of a service program on z/OS, which controls the time to process this file from the printer queue. You can write the service program to decide how to process the print file.</p> <div>  <p>Do not specify a value for this parameter unless you are instructed by the system analyst of the z/OS system.</p> </div>
UNIT (du)	<p>Defines the remote file z/OS unit name.</p> <p>The maximum length of the defined value is 8 characters.</p>
VOLUME (dv vol volser)	<p>Defines the remote file z/OS volume name.</p> <p>The maximum length of the defined value is 6 characters, and it must be alpha-numeric.</p>
zOS	<p>Defines whether you can perform transfers to or from z/OS.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> Y: you can perform transfers to or from z/OS. N: you cannot perform transfers to or from z/OS, and all z/OS specific transfer parameters are ignored.

TIBCO Accelerator Parameters

You must supply the TIBCO Accelerator transfer parameters when you use the Accelerator (ACC) technology to do a transfer.

Parameter (Alternate Specification)	Description
Accelerate (ACC)	<p>Defines whether to conduct file transfers using Accelerator technology.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • N: does not use the Accelerator technology for a transfer. • Y: forces a transfer to be conducted using Accelerator technology, so you can speed up data transfer over IP networks with high latency.
ACCCompression (ACCC)	<p>Defines whether to compress the data transferred through Accelerator.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • N: no compression. • Y Best: best balance between speed and compression quality. • Default: highest compression with slightly lower speed. • Fast: lower compression but with fast speed. <div>  Accelerator uses a proprietary compression compatible with ZLIB compression. </div>
ACCEncryption (ACCE)	<p>Defines whether to encrypt the data transferred through Accelerator.</p> <p>The encryption type is 256-bit Blowfish encryption. The valid values are N or Y.</p>
ACCHost (ACCH)	<p>Defines the IP address or host name of Accelerator server.</p> <p>The valid values are N, <i>hostname</i>, or <i>IP _address</i>.</p> <div>  A host value defined on the command line or in a transfer template overrides the ACCHost value configured in <code>config.txt</code> file. If the value defined on the command line or in a transfer template is N, and you have Accelerate set to Y, the value configured for ACCHost in <code>config.txt</code> file is used. </div> <p>For more information, see Configuration Parameters.</p>
ACCMaxSpeed (ACCMAX)	<p>Defines the maximum speed, in kilobytes per second, for transfers through Accelerator.</p> <p>You can set this parameter on the command line or in a transfer template. The valid values are from 256 to 1000000.</p> <p>For more information, see Transfers Using Templates.</p>

Parameter (Alternate Specification)	Description
ACCPort	<p>Defines the port number on which the platform server listens for transfers using the Accelerator technology.</p> <p>The valid values are <i>N</i> or <i>port_number</i>. Default value is 9099.</p> <div data-bbox="699 453 740 495">  </div> <div data-bbox="808 386 1487 575"> <p>A port number defined on the command line or in a transfer template overrides the ACCPort value configured in <code>config.txt</code> file. If the value defined on the command line or in a transfer template is <i>N</i> and you have Accelerate set to <i>Y</i>, the value configured for ACCPort in <code>config.txt</code> file is used.</p> </div> <p>For more information, see Transfers Using Templates and Configuration Parameters.</p>
ACCProtocol (ACCP)	<p>Defines the transmission protocol for file transfers through Accelerator server.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • TCP: Transmission Control Protocol • UDP: Accelerator server enhanced version of User Datagram Protocol • PDP: Parallel Delivery Protocol

Configuration Parameters

To configure TIBCO MFT Platform Server, you can modify the parameters in the `config.txt` file according to your transfer needs and installation environments.

The `config.txt` file is by default located in the `$CFROOT/config` directory. You can use a text editor to modify the configuration file. After configuring, TIBCO MFT Platform Server can be used as a responder (server) or an initiator (client). The default port number is 46464.



TIBCO MFT Platform Server must be restarted for any changes to the `config.txt` file to take effect. For more instructions, see [Starting TIBCO MFT Platform Server](#).

The `config.txt` file is divided to 3 sections, [Server Configuration](#), [Client Configuration](#), and [Common Configuration](#). Each section provides detailed information about the available parameters in the configuration file.

Server Configuration

You can configure TIBCO MFT Platform Server to act as a responder.


See the following default server (responder) configuration for TIBCO MFT Platform Server.



```
# [ SERVER ]
ListenAdapterIP:           All                { All, IpName/Address }
ListenAdapterIPv6:        All                { All, IpName/AddressIPv6 }
Port:                      46464
PortIPv6:                 N                  { N, IPv6 Port }
TraceLevel:               N                  { N, Low|L, Medium|M, High|H }
TracePath:                /mftps/trace/Responder { N, Path }
TraceSizeServer:          N                  { N, # of Kb }
ConvTbl:                  N                  { N, FileName }
ExitPrgm:                 N                  { N, FileName }
RequiredNodeDefinition:   N                  { N, Y }
AcceptVerifiedUser:       N                  { N, Y }
ResponderProfile:         N                  { N, Y, D }
AllowRoot:                N                  { N, All, Password }
Umask_Default:            N                  { N, 3 digit number }
Uperm_Default:            N                  { N, 3 digit number }
Timeout:                  120                { Transfer timeout in min }
RunCyberRespAsNonRoot:    N                  { Y, N }
ClassOfService:           Default            { from cfcos.cfg }
PamAuth:                  N                  { N, service_name }
# When RunCyberRespAsNonRoot is set to Y ResponderProfile must be set to Y


# SSL Communication Additional Parameters.
SSLPort:                  56565
SSLPortIPv6:              N                  { N, IPv6 Port }
TunnelPort:               58585
TunnelPortIPv6:           N                  { N, IPv6 Port }
ClientVerification        Y                  { N, Y }
CertificateKeyFileName:
PrivateKeyFileName:
PrivateKeyPwdFileName:
TrustedAuthorityFileName:
AuthorizationFileName:    N                  { N, FileName }
SSLTraceLevel:            N                  { N, Y }
SSLTracePath:             /mftps/trace/SSLResponder { N, Path }
CheckCRL:                 N                  { N, Y }
CAPath:
SSLEnabledProtocols:      TLSV1,TLSV1.1,TLSV1.2 { TLSV1,TLSV1.1,TLSV1.2 }
Ciphers:                  HIGH                { openssl_cipher_list }
```





Server Configuration Parameters

The following table lists parameters used to configure TIBCO MFT Platform Server as a responder.

Parameter Name	Description
ListenAdapterIP	<p>Defines the IP address at which the responder listens for incoming connections.</p> <p>The default value for this parameter is <code>All</code>, which means connections can bind to any IP address.</p> <p>If a machine has more than one IP address, you can bind the connection to a particular one. It guarantees that all the transfers go only through this particular IP address.</p>
ListenAdapterIPv6	<p>Defines the IPv6 address at which the responder listens for incoming connections.</p> <p>The default value for this parameter is <code>All</code>, which means connections can bind to any IP address.</p> <p>If a machine has more than one IPv6 address, you can bind the connection to a particular one. It guarantees that all the transfers go only through this particular IPv6 address.</p>
Port	<p>Defines the IP port on which TIBCO MFT Platform Server listens for incoming requests.</p> <p>The valid values are from 1024 to 65535, because lower ports are usually reserved for standard applications. The default value is 46464.</p>
PortIPv6	<p>Defines the IPv6 port on which TIBCO MFT Platform Server listens for incoming requests.</p> <p>The valid values are <code>N</code> or a number ranging from 1024 to 65535, because lower ports are usually reserved for standard applications.</p>
TraceLevel	<p>Defines the level of tracing.</p> <p>The valid values are <code>N</code>, <code>Low</code> <code>L</code>, <code>Medium</code> <code>M</code>, or <code>High</code> <code>H</code>. The default value is <code>N</code>.</p> <p>Usually tracing has only to be turned on at the request of TIBCO Support for troubleshooting purpose.</p> <div>  <ul style="list-style-type: none"> This parameter cannot be used within a transfer template. When the TraceLevel is used, the TraceSizeServer must be defined. </div>

Parameter Name	Description
TracePath	<p>Defines the name of the path that holds the responder trace file.</p> <p>A unique trace file is created for each file transfer. The file name contains the local transaction number and Process ID (PID).</p> <p>A global server trace file is created each time the cfstart command is used to start the TIBCO MFT Platform Server daemon, CyberResp. The file name is <i>server name</i> with the current time as an extension. This file contains the PIDs of all child processes that are started by TIBCO MFT Platform Server.</p> <p> This parameter cannot be used within a transfer template.</p>
TraceSizeServer	<p>Defines the size limit of the trace file in kilobytes.</p> <p>The valid values are N or a number of kilobytes. The default value is N.</p> <p>The file name is the trace file name (local transaction number and PID) with a t1 extension. When the specified size limit has been reached, a second file with a t2 extension is created. When the second file is full, all data contained in the t1 file is deleted and it begins again from size 0. This process then repeats on the t2 file. This trace file swapping continues for the duration of the file transfer.</p> <p> <ul style="list-style-type: none"> This parameter cannot be used within a transfer template. When TraceSizeServer is used, TraceLevel or SSLTraceLevel in the server part must be defined. </p>
ConvTbl	<p>Defines the path to the standard conversion table.</p> <p>It provides ASCII to EBCDIC conversion for any transactions to or from z/OS and AS/400 platforms. The default name of this file is Comtblg.dat which is located in the \$CFROOT directory.</p> <p>For more information, see Conversion Tables.</p>
ExitPrgm	<p>Defines the path to the Exit program on the local machine.</p> <p>Using the Exit program, you can do customized post processing.</p> <p>For more information, see User Exits.</p>
RequiredNodeDefinition	<p>Defines whether a node definition is required for TIBCO MFT Platform Server to communicate with a remote system.</p> <p>Value Y means that the remote IP address requires a defined node. If the remote address is not defined in a node (in either the cfnode.cfg or cfprofile.cfg file), the responder rejects the initiator and sends back an error message.</p>

Parameter Name	Description
AcceptVerifiedUser	<p>Defines whether a remote verified user can log on to TIBCO MFT Platform Server using the remote user ID without a password.</p> <p>TIBCO MFT Platform Server recognizes the client as verified if the client sends an internal password inside the password field.</p> <p>To log on using only a user ID without a password, the initiator platform must provide the following for the remote user ID (this is case sensitive):</p> <ul style="list-style-type: none"> • Userid: *VER • Password: (Password field must be left blank.) <p>The initiator user ID can also be used as the responder user ID. This means that the same user ID has to exist on the responder as well as the initiator platforms.</p>
ResponderProfile	<p>Defines a local user name and password to substitute the incoming user name and password.</p> <p>By using responder profiles, a remote TIBCO MFT Platform Server does not have to get an actual user name and password from your local machine to initiate a transfer.</p> <div>  <p>ResponderProfile checking routine is always done before AcceptVerifiedUser checking. So if both are set up, AcceptVerifiedUser takes precedence.</p> </div> <p>ResponderProfile value D (Dual) means that the substitution of a real user ID occurs only if the <code>cfrprofile</code> file exists and a match is found. If there is no match found, then TIBCO MFT Platform Server tries to log on using the remote user ID and password, and no error message is generated.</p>
AllowRoot	<p>Defines whether the root account is considered as a valid user ID for transfers.</p> <p>If the responder profile defines the root account as the local user ID:</p> <ul style="list-style-type: none"> • When AllowRoot is set to <i>password</i> or <i>all</i>, the root account can be used as a valid user ID. • When AllowRoot is set to <i>no</i>, the root account cannot be used as a valid user ID.





Parameter Name	Description
Umask_Default	<p>Defines the UNIX umask applied to the newly created files on the server (responder).</p> <p>The valid values are N or a three digit number ranging from 000 to 777. You can use this parameter to modify the permissions for newly created files on the responder side, just as the UNIX umask sets the permissions on newly created files.</p> <p>If Umask_Default is set to N, then the file permission can be set according to the root user mask. On the initiator, the desired permission is modified according to the umask of the user that runs the cfsend or cfrecv command.</p> <div>  <p>There is no command line or template option for Umask. This parameter is only contained in the <code>config.txt</code> file.</p> </div>
Uperm_Default	<p>Defines the file permissions for newly created files on the server (responder).</p> <p>The valid values are N or a three digit number ranging from 000 to 777.</p> <p>The Uperm_Default parameter is used when a user sends a new file to the UNIX server and has not provided the uperm parameter. (This is not possible on a UNIX initiator, as the uperm parameter is set by default to the file permissions of the sending file).</p> <p>If the Uperm_Default is set to N, the file permissions for the newly created file are set according to the Umask_Default parameter.</p> <div>  <p>For transfers between UNIX systems, if a file is not executable on the initiator, it cannot be changed to executable on the responder. This is a property of UNIX, not of TIBCO MFT Platform Server.</p> </div>
Timeout	<p>Defines the amount of time in minutes that a connection stays open when waiting for a response from the remote side.</p> <div>  <p>Once the timeout value is reached, the connection ends.</p> </div>
RunCyberRespAsNonRoot	<p>Defines whether you can run TIBCO MFT Platform Server as a non-root user.</p> <div>  <p>When running CyberResp with a non-root user ID, you must use responder profiles. Therefore, when RunCyberRespAsNonRoot is set to Yes, ResponderProfile must be set to Yes too.</p> </div>
ClassOfService	<p>Defines the buffer size for the server in a transfer.</p> <p>The valid values are the different levels of SBUFSIZE and RBUFSIZE specified in the configuration file named <code>cfcos.cfg</code>. The transfer performance can be improved as server and client using different level of classes.</p>



Parameter Name	Description
PamAuth	<p>Defines the PAM service file name or the PAM service entry name.</p> <p>The valid values are N or the PAM service name. The PAM service defines the PAM parameters used for authentication. When this parameter is set to N, password or shadow password authentication is performed.</p>

Server SSL Communications Parameters

The following table lists parameters only used when performing SSL or tunnel transactions.

Parameter Name	Description
SSLPort	<p>Defines an IP port on which TIBCO MFT Platform Server listens on for incoming SSL requests.</p> <p>The valid values are from 1024 to 65535, because lower ports are usually reserved for standard applications. The default value is 56565.</p> <p>If the parameter is not defined, TIBCO MFT Platform Server does not listen for incoming SSL requests.</p>
SSLPortIPv6	<p>Defines an IPv6 port on which TIBCO MFT Platform Server listens on for incoming SSL requests.</p> <p>The valid values are N or any number ranging from 1024 to 65535, because lower ports are usually reserved for standard applications. If this parameter is not defined, then responder IPv6 SSL processing is disabled.</p> <p>If non-SSL requests are received on this port, then an error message is sent to the initiator and the request is terminated.</p> <p>This field must be different than the PortIPv6 parameter.</p>
TunnelPort	<p>Defines an IP port on which TIBCO MFT Platform Server listens on for incoming tunnel requests.</p> <p>The valid values are from 1024 to 65535, because lower ports are usually reserved for standard applications. The default value is 58585.</p>
TunnelPortIPv6	<p>Defines an IPv6 port on which TIBCO MFT Platform Server listens on for incoming tunnel requests.</p> <p>The valid values are N or any number ranging from 1024 to 65535, because lower ports are usually reserved for standard applications. If this parameter is not defined, then responder IPv6 tunnel processing is disabled.</p> <p>This field must be different than the PortIPv6 parameter.</p>

Parameter Name	Description
ClientVerification	<p>Defines whether TIBCO MFT Platform Server performs SSL client authentication.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • N: the client certificate is not authenticated. • Y: the client certificate is authenticated. This is the default value. <p>For more information, see SSL Certificates Setup.</p>
CertificateFileName	<p>Defines the path to the certificate file used for an SSL transfer.</p> <div>  <p>It has no default value. There are separate parameters for server and client, but the same file name can be used for both.</p> </div> <p>For more information, see SSL Certificates Setup.</p>
PrivateKeyFileName	<p>Defines the path to the file with the private key that is associated with the SSL certificate.</p> <div>  <p>It has no default value. There are separate parameters for server and client, but the same file name can be used for both.</p> </div> <p>For more information, see SSL Certificates Setup.</p>
PrivateKeyPwdFileName	<p>Defines the path to the file with the private key password.</p> <p>It has no default value. To create this file, use the <code>createPwd.exe</code> file in the <code>\$CFROOT/util</code> directory.</p> <div>  <p>If the same certificate is used for a TIBCO MFT Platform Server server and a TIBCO MFT Platform Server client, the same private key password can be used for the server and client as well.</p> </div> <p>For more information, see SSL Certificates Setup.</p>
TrustedAuthorityFileName	<p>Defines the path to the file with trusted authority certificates.</p> <p>It has no default value. Use this parameter to define all the certificate authorities that are accepted by both a TIBCO MFT Platform Server and a TIBCO MFT Platform Server client.</p> <div>  <p>There are separate parameters for server and client, but the same file name can be used for both.</p> </div> <p>For more information, see SSL Certificates Setup.</p>

Parameter Name	Description
AuthorizationFileName	<p>Defines the path to the authorization file to be used with SSL transfers.</p> <p>If this parameter is not defined or is set to N, TIBCO MFT Platform Server does not perform additional authentication to the client certificate. This parameter is only valid when ClientVerification is set to Y. You can find a sample authorization file that can be used called <code>SSLAuth.cfg</code> located in the <code>\$CFROOT/config</code> directory.</p> <p>For more information on configuring this file, see Configured SSL Authorization.</p>
SSLTraceLevel	<p>Defines whether tracing is turned on for an SSL transfer.</p> <p>Usually tracing has only to be turned on at the request of TIBCO Support for troubleshooting purpose.</p> <div>  <ul style="list-style-type: none"> This parameter cannot be used within a transfer template. When the SSLTraceLevel is used, the TraceSizeServer parameter must be defined. </div>
SSLTracePath	<p>Defines the path to the SSL trace file.</p> <p>The path of the SSL trace file is <code>\$CFROOT/trace/ResponderSSL</code> under the SERVER section. Normally these files are only used when debugging SSL related problems with TIBCO Support.</p> <div>  <p>This parameter cannot be used within a transfer template.</p> </div>
CheckCRL	<p>Defines whether TIBCO MFT Platform Server checks the CAPath field for the hashed CRL files.</p> <p>For more information, see CRL Support.</p>
CAPath	<p>Defines the path where the CRL checking looks for the hashed file names.</p> <p>For more information, see CRL Support.</p>
SSLEnabledProtocols	<p>Defines which SSL protocols are supported when the platform server runs as a responder.</p> <p>The valid values are any combination of the following options:</p> <p>TLSV1,TLSV1.1,TLSV1.2. The default value is TLSV1,TLSV1.1,TLSV1.2. The comma means and.</p>

Parameter Name	Description
Ciphers	<p>Defines the cipher suites that can be used for TLS negotiation between the server and the client.</p> <p>The default value is HIGH, which means those ciphers suites with key lengths larger than 128 bits, and some cipher suites with 128-bit keys can be used.</p> <p>In addition, you can list all supported cipher suites separated by colons. You can list the OPENSSL supported ciphers by using the openssl command.</p> <p>Examples:</p> <p>List TLS Ciphers:</p> <pre>\$CFROOT/util/openssl ciphers -tls1</pre> <p>List "high" encryption TLS Cipher suites:</p> <pre>\$CFROOT/util/openssl ciphers -tls1 HIGH</pre>

Client Configuration

You can configure TIBCO MFT Platform Server to act as an initiator.


See the following default client (initiator) configuration for TIBCO MFT Platform Server.




```
# [ CLIENT ]
RequiredNodeDefinition: N { N, Y }
ConnectAdapterIP: All { All, IpName/Address }
ConnectAdapterIPv6: All { All, IpName/AddressIPv6 }
TraceLevelClient: N { N, Low|L, Medium|M, High|H }
TracePathClient: /mftps/trace/Initiator { N, Path }
TraceSizeClient: N { N, # of Kb }
Umask_User: N { N, Y }
Timeout: 120 { Transfer timeout in min }
DirTransfer0Files: Failure { Failure|F, Success|S }
RunPPAEndDirTx: N { N, Y }
ACCHost: N { N, Host }
ACCPort: 9099 { Port Number }
ClassOfService: Default { from cfcos.cfg }
StopOnFailure: Y { Y, N }


# SSL Communication. Additional Parameters.
CertificateFileName:
PrivateKeyFileName:
PrivateKeyPwdFileName:
TrustedAuthorityFileName:
SSLTraceLevel: N { N, Y }
SSLTracePath: /mftps/trace/SSLInitiator { N,
Path }
CheckCRL: N { N, Y }
CAPath:
SSLEnabledProtocols: TLSV1,TLSV1.1,TLSV1.2 {TLSV1,TLSV1.1,TLSV1.2}
Ciphers: HIGH { openssl_cipher_list }
```

Client Configuration Parameters

The following table lists parameters used to configure TIBCO MFT Platform Server as an initiator.



Parameter	Description
RequiredNodeDefinition	<p>Defines whether a node definition is required for TIBCO MFT Platform Server to communicate with a remote system.</p> <p>Value Y means that the remote IP address requires a defined node. If the remote address is not defined in a node, the initiator rejects the transfer and displays an error message.</p>
ConnectAdapterIP	<p>Defines the IP address through which the initiator sends or receives data for outgoing connections .</p> <p>The default value for this parameter is ALL which means connections can bind to any IP address.</p> <p>If a machine has more than one IP address, it is possible to bind the connection to a particular one. It guarantees that all the transfers only go through this particular IP address.</p>
ConnectAdapterIPv6	<p>Defines the IPv6 address through which the initiator sends or receives data for outgoing connections .</p> <p>The default value for this parameter is ALL which means connections can bind to any IPv6 address.</p> <p>If a machine has more than one IPv6 address, it is possible to bind the connection to a particular one. It guarantees that all the transfers go only through this particular IPv6 address.</p>
TraceLevelClient	<p>Defines the level of tracing that occurs.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • N No: no tracing takes place. This is the default value. • L Low: provides minimal information about the transfer, including local and remote transaction numbers, file names, the number of bytes transferred, fail or success status indicator, general message string, and the start and finish times of the transfer. • M Medium: includes all internal state messages. • H High : includes all networking data in addition to the information provided by the medium level trace. <p>Usually tracing has only to be turned on at the request of TIBCO Support for troubleshooting purpose.</p> <div>  <ul style="list-style-type: none"> • This parameter cannot be used within a transfer template. • When the TraceLevelClient is used, the TraceSizeClient must be defined. </div>

Parameter	Description
TracePathClient	<p>Defines the name of the path that holds the client trace file.</p> <p>A unique trace file is created for each file transfer. A file name contains the local transaction number and Process ID (PID).</p> <p>A global server trace file is created each time the cfstart command is used to start the daemon, CyberResp. The file name is <i>servername</i> with the current time as an extension. This file contains the PIDs of all child processes that are started by TIBCO MFT Platform Server.</p> <div>  <p>This parameter cannot be used within a transfer template.</p> </div>
TraceSizeClient	<p>Defines the size limit of the trace file in kilobytes.</p> <p>The file name is the trace file name (local transaction number and PID) with a t1 extension. When the specified size limit is reached, a second file with a t2 extension is created. When the second file is full, all data contained in the t1 file is deleted and it begins again from size 0. This process then repeats on the t2 file. This trace file swapping continues for the duration of the file transfer.</p> <div>  <ul style="list-style-type: none"> • This parameter cannot be used within a transfer template. • When TraceSizeClient is used, TraceLevelClient or SSLTraceLevel in the client part must be defined. </div>
Umask_User	<p>Defines whether to apply the user's umask to the incoming files.</p> <div>  <p>This parameter only applies to the initiator that is doing a receive transfer.</p> </div> <ul style="list-style-type: none"> • If Umask_User is set to N, the user's umask is ignored on incoming files. • If Umask_User is set to Y, the user's umask is applied to incoming files..
Timeout	<p>Defines the amount of time in minutes that a connection stays open when waiting for a response from the remote side.</p> <p>The default value is 120 minutes. Once the timeout value is reached, the connection ends.</p>

Parameter	Description
RunPPAEndDirTx	<p>Defines whether to have the post processing action run after each file in a directory or a distribution list is transferred.</p> <p>The valid values are No or Yes. The default value is No.</p> <p>When this parameter is set to Yes , the following rules apply:</p> <ul style="list-style-type: none"> • StopOnFailure is automatically set to Yes. Transfers stop on the first failed transfer. • Failure PPA runs on the first failed transfer. • Successful PPA runs only on the last transfer (assuming it is successful). <div>  <p>Since this is a global parameter, it affects all transfers.</p> </div>
ACCHost	Defines the IP address or host name of the Accelerator server.
ACCPort	<p>Defines the port number on which the Accelerator server listens for transfers using the Accelerator technology.</p> <p>The default number is 9099.</p>
ClassOfService	<p>Defines the buffer size for the client in a transfer.</p> <p>The valid values are the different levels of SBUFSIZE and RBUFSIZE specified in the configuration file named <code>cfcos.cfg</code> The transfer performance can be improved as server and client using different level of classes.</p>
StopOnFailure	Defines the default value for StopOnFailure when this parameter is not specified by the <code>cfsend</code> or <code>cfrecv</code> commands. StopOnFailure defines whether a directory transfer or distribution list transfer should stop on the first failure (Y) or continue if a transfer fails.
DirTransfer0Files	<p>Defines the status of the directory transfer when zero files are in the directory.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • F: When no files are transferred on a directory transfer, the transfer is a failed transfer. • S: When no files are transferred on a directory transfer, the transfer is a successful transfer. <p>The default value is F.</p>

Client SSL Communications Parameters

The following table lists parameters only used when performing SSL or tunnel transactions.

Parameter	Description
CertificateFileName	<p>Defines the path to the certificate file used for an SSL transfer.</p> <p>It has no default value. There are separate parameters for the server and client, but the same file name can be used for both.</p> <p>For more information, see SSL Certificates Setup.</p>
PrivateKeyFileName	<p>Defines the path to the file with the private key that is associated with the SSL certificate.</p> <p>It has no default value. There are separate parameters for the server and client, but the same file name can be used for both.</p> <p>For more information, see SSL Certificates Setup.</p>
PrivateKeyPwdFileName	<p>Defines the path to the file with the private key password.</p> <p>It has no default value. To create this file, use the <code>createPwd.exe</code> program in the <code>\$CFROOT/util</code> directory. If the same certificate is used for the server and client, then the same private key password can be used for the server and client as well.</p> <p>For more information, see SSL Certificates Setup.</p>
TrustedAuthorityFileName	<p>Defines the path to the file with the trusted authority certificates.</p> <p>It has no default value. It defines all of the certificate authorities that are accepted by the server and client. There are separate parameters for the server and client, but the same file name can be used for both.</p> <p>For more information, see SSL Certificates Setup.</p>
SSLTraceLevel	<p>Defines whether tracing is turned on for an SSL transfer.</p> <p>Usually tracing has only to be turned on at the request of TIBCO Support for troubleshooting purpose.</p> <div>  <ul style="list-style-type: none"> This parameter cannot be used within a transfer template. When SSLTraceLevel is used, the TraceSizeClient parameter must be defined. </div>
SSLTracePath	<p>Defines the path to the SSL trace file.</p> <p>The path of the SSL trace file is <code>\$CFROOT/trace/InitiatorSSL</code> under the CLIENT section respectively. Normally these files are only used when debugging an SSL related problems with TIBCO Support.</p> <div>  <p>This parameter cannot be used within a transfer template.</p> </div>

Parameter	Description
CheckCRL	Defines whether TIBCO MFT Platform Server checks the CAPath field for the hashed CRL files. For more information, see CRL Support .
CAPath	Defines the path where the CRL checking looks for the hashed file names. For more information, see CRL Support .
SSLEnabledProtocols	Defines which SSL protocols are supported when the platform server runs as an initiator. The valid values are any combination of the following options: TLSV1,TLSV1.1,TLSV1.2. The default value is TLSV1,TLSV1.1,TLSV1.2. The comma means and.
Ciphers	Defines the cipher suites that can be used for TLS negotiation between the server and the client. The default value is HIGH, which means those cipher suites with key lengths larger than 128 bits, and some cipher suites with 128-bit keys can be used. In addition, you can list all supported cipher suites separated by colons.

Common Configuration


You can use certain common configuration on all incoming and outgoing transfer requests.


See the following default common configuration for all transfer requests, regardless of whether the server acts as a responder or an initiator.



```
# [ COMMON ]
SecurityPolicy:      N                      { None, HIPAA, FIPS140 }
LogEventFileName:    /mftps/log/Log.txt      { N, FileName }
TransnumFileName:    /mftps/transnum         { FileName }
PQFDirectory:        /mftps/PQF             { DirName }
AuditTempErrors:     N                      { N, Y }
SemaphoreKey:        0x07e9368b
SMTPServer:          N                      { IpName/Address:port, N }
FromAddress:          N                      { Email Address, N }
Subject:              N                      { Subject String, N }
CfgPostProc:         N                      { N, FileName }
AccessControlConfig: N                      { N, FileName }
AliasConfig:          N                      { N, FileName }
Encode:              N                      { non utf8 code page }
AdminGroup:           cfadmin                { group name }
BrowseGroup:          cfbrowse                { group name }
TransferGroup:        cftransfer              { group name }
LogDirectoryTransfers: Y                    { Y, N, Errors }
CRC:                  N                      { Y, N }
# Parse Commands
protect_cctransfer:   exec                   { none|reject|exec }
protect_cfdir:        reject                  { none|reject|exec }
protect_fusutil:      reject                  { none|reject|exec }
protect_receivedir:   exec                   { none|reject|exec }
protect_rcmd:         reject                  { none|reject|exec }
protect_ppa:          token                   { none|token|exec }
protect_cfgpostproc:  token                   { none|token|exec }
rejectcmdcharacters:  ;&|                    { up to 10 characters }
```

Common Configuration Parameters

The following table lists parameters used to configure for all transfer requests.

Parameter	Description
SecurityPolicy	<p>Defines whether TIBCO MFT Platform Server complies with any security policy on send and receive transfers.</p> <ul style="list-style-type: none"> • HIPAA: this setting requires TIBCO MFT Platform Server to comply with HIPAA (Health Insurance Portability and Accountability Act) standards. The standards require all file transfers to use encryption key length that is 128 bits or greater. • FIPS140: this setting requires TIBCO MFT Platform Server to comply with FIPS (Federal Information Processing Standard). This requires that all file transfers to use SSL with an encryption type of Rijndael (AES) which uses a key length of 256 bits. This is a Government standard that certifies cryptographic modules used for the protection of sensitive but unclassified information and communications in electronic commerce within a security system. • None: no security policy is enforced on TIBCO MFT Platform Server. <p> If you initiate a transfer using DES encryption, which is not allowed for either HIPAA or FIPS-140, the encryption is overridden with a certified encryption method. If you are using HIPAA, a prompted message is displayed informing you the encryption is changed to Blowfish Long. If you are using FIPS-140, you receive a prompted message informing you the encryption is changed to Rijndael (AES).</p>
LogEventFileName	<p>Defines the name of the file that holds the initiator log file.</p> <p>By default, the log file is located in the <code>\$CFROOT/log</code> directory. Ensure that the directory exists before you change the name of the log directory.</p>
TransnumFileName	<p>Defines the file name where the current transaction number is stored. Platform Server uses this file to generate the transaction ID when a transfer is started. This parameter should only be set when running in a container because this file must be saved in persistent storage. Otherwise, you should use the default value.</p>
PQFDirectory	<p>Defines the directory where the PQF files are stored. PQF files store transfer restart information when a transfer fails and can be restarted. This parameter should only be set when running in a container because PQF files should be saved in persistent storage. If this parameter is not set, restart still works when executing in a container, but the restart information is lost if the container restarts.</p>
AuditTempErrors	<p>Defines whether all transfer attempts or only the final attempt is logged.</p>

Parameter	Description
SemaphoreKey	<p>Defines the key used to create a semaphore.</p> <p>If there are several transfers going on simultaneously, the output statements from different transactions can overwrite each other. This situation can be prevented by using a semaphore that synchronizes access to the log file.</p> <p>The valid values are decimal numbers between 1 and 2147483647 or hexadecimal numbers between 0x00000001 and 0x7fffffff. Hexadecimal numbers must be prefixed with 0x.</p> <div>  <p>The default value 0x07e9368b cannot be changed unless instructed by TIBCO Support.</p> </div>
SMTPServer	<p>Defines the name of the email server and the port that is used to send out email notifications.</p> <p>The format to define the port is:</p> <p>your.smtp.server:port</p> <p>If the port is not defined, it defaults to port 25.</p> <p>Example:</p> <p>your.smtp.server:25</p>
FromAddress	Defines the value of the From Name field in the email notification.
Subject	<p>Defines the Subject line of the email notification.</p> <p>The maximum length of the defined value is 256 characters.</p>
CfgPostProc	<p>Defines the name of the file that holds the post processing configuration.</p> <p>For more information, see Configured Post Processing.</p>
AccessControlConfig	<p>Defines the path to the AccessControl.cfg in the \$CFROOT/config directory.</p> <p>You can change the default directory for a file based on the USERID, NODE or IPADDR parameters on responder transfer requests only.</p> <p>For more information, see Access Control.</p>
AliasConfig	<p>Defines the path to the CfAlias.cfg file in the \$CFROOT/config directory.</p> <p>You can use an alias file name based on the USERID, NODE or IPADDR parameters for responder transfer requests only.</p> <p>For more information, see CfAlias.</p>


Parameter	Description
Encode	<p>Defines how file names are translated when sent to the remote Platform Server or Internet Server. Valid values are:</p> <ul style="list-style-type: none"> • N: File names contain standard Latin characters and will not be converted to UTF-8. • A valid Coded Character Set: Tells Platform Server to convert the file names from this character set to UTF-8. <p>When this parameter is set to a value other than N, the data is converted from this character set to UTF-8. TIBCO MFT Platform Server then sends the UTF-8 file name to the target system where it is converted back to the characters set defined on that Platform Server or Internet Server.</p> <p>For example, if you have file names with Korean or Chinese characters, then you should set this parameter to the character set of the local UNIX machine.</p>
AdminGroup	Defines the group name that holds users who can configure nodes, profiles, and responder profiles, as well as view audit records from all users.
BrowseGroup	<p>Defines the group name that holds users who can view audit records from all users.</p> <div>  <p>Users who are not in the specified browse group can only view transactions that they conducted.</p> </div>
TransferGroup	<p>Defines the group name that holds users who can conduct platform to platform file transfers initiated from Command Center.</p> <div>  <p>If this group does not exist and a transfer request comes in from Command Center, the transfer can succeed based on the node configurations for the Command Center. If the group does exist and the end user account being used for a file transfer initiated from Command Center is not a member, the transfer fails.</p> </div>
LogDirectoryTransfers	<p>Defines whether to log cfdir requests when doing directory transfers.</p> <p>The valid values are Y, N or Errors. The default value is Y. Errors means the cfdir request is logged only when an error occurs.</p>
CRC	<p>Defines whether to perform a CRC check.</p> <p>The valid values are N or Y.</p>

Parse Commands

Parameter	Description
protect_cctransfer	<p>Defines the processing performed when MFT Command Center initiates a Platform Server Transfer to Platform Server for UNIX.</p> <p>The valid values are:</p> <p>none: No additional parsing is performed when Platform Server executes a system command.</p> <p>reject: If a command to be executed includes any of the characters defined by the rejectcmdcharacters parameter, the command terminates with an error.</p> <p>exec: The default value is exec: The command to be executed calls the exec function. This call does not allow multiple commands to be executed in a single command string.</p> <p>The exec option supports up to 100 command line parameters.</p>
protect_cfdir	<p>Defines the processing performed when a cfdir request is received. A cfdir request prompts Platform Server to return a directory list or the status of a file or directory.</p> <p>The valid values are:</p> <p>none: No additional parsing is performed when Platform Server executes a system command.</p> <p>reject: If a command to be executed includes any of the characters defined by the rejectcmdcharacters parameter, the command terminates with an error.</p> <p>exec: The command to be executed calls the exec function. This call does not allow multiple commands to be executed in a single command string. The exec option supports up to 100 command line parameters.</p> <p>The default value is reject.</p>

Parameter	Description
protect_cfgpostproc	<p>Defines the processing performed when a configured postprocessing command is executed.</p> <p>The valid values are:</p> <p>none: No additional parsing is performed when Platform Server executes a system command. You can add the ampersand sign (&) as the last character to prompt Platform Server to execute the command in the background. You can add the pound sign (#) as the last character to prompt Platform Server to wait for the system call to complete.</p> <p>token: If a PPA or substitutable postprocessing token includes any of the characters defined by the rejectcmdcharacters parameter, the command terminates with an error. The following PPA or configured postprocessing tokens lists some of the tokens:</p> <ul style="list-style-type: none"> • File Name related (i.e. any token computed from a file name) • User Data • Process Name <p>exec: The command to be executed calls the exec function. This call does not allow multiple commands to be executed in a single command string. You can add the ampersand sign (&) as the last character to prompt Platform Server to execute the command in the background. You can add the pound sign (#) as the last character to prompt Platform Server to wait for the exec call to complete.</p> <p>The exec option supports up to 100 command line parameters.</p> <p>The default value is token.</p>

Parameter	Description
protect_fusutil	<p>Defines the processing performed when a fusutil request is received. A fusutil request prompts Platform Server to perform one of the following functions:</p> <ul style="list-style-type: none"> • Deletes a file or directory • Renames file or directory • Returns whether a file exists • Creates a directory <p>The valid values are:</p> <p>none: No additional parsing is performed when Platform Server executes a system command.</p> <p>reject: If a command to be executed includes any of the characters defined by the rejectcmdcharacters parameter, the command terminates with an error.</p> <p>exec: The command to be executed calls the exec function. This call does not allow multiple commands to be executed in a single command string. The exec option supports up to 100 command line parameters.</p> <p>The default value is reject.</p>
protect_ppa	<p>Defines the processing performed when a PPA command is executed.</p> <p>The valid values are:</p> <p>none: No additional parsing is performed when Platform Server executes a system command. You can add the ampersand sign (&) as the last character to prompt Platform Server to execute the command in the background. You can add the pound sign (#) as the last character to prompt Platform Server to wait for the system call to complete.</p> <p>token: If a PPA or configured postprocessing token includes any of the characters defined by the rejectcmdcharacters parameter, the command terminates with an error. The following PPA or configured postprocessing tokens lists some of the tokens:</p> <ul style="list-style-type: none"> • File Name related (i.e. any token computed from a file name) • User Data • Process Name <p>exec: The command to be executed calls the exec function. This call does not allow multiple commands to be executed in a single command string. You can add the ampersand sign (&) as the last character to prompt Platform Server to execute the command in the background. You can add the pound sign (#) as the last character to prompt Platform Server to wait for the system call to complete. The exec option supports up to 100 command line parameters.</p> <p>The default value is token.</p>

Parameter	Description
protect_rcmd	<p>Defines the processing performed when receiving a command other than cfdir or fusutil.</p> <p>The valid values are:</p> <p>none: No additional parsing is performed when Platform Server executes a system command. You can add the ampersand sign (&) as the last character to prompt Platform Server to execute the command in the background. You can add the pound sign (#) as the last character to prompt Platform Server to wait for the system call to complete.</p> <p>reject: If a command to be executed includes any of the characters defined by the rejectcmdcharacters parameter, the command terminates with an error.</p> <div>  <p>You can the ampersand sign (&) to tell Platform Server to execute the command in background.</p> </div> <p>exec: The command to be executed calls the exec function. This call does not allow multiple commands to be executed in a single command string. You can add the ampersand sign (&) as the last character to prompt Platform Server to execute the command in the background. You can add the pound sign (#) as the last character to prompt Platform Server to wait for the exec call to complete. The exec option supports up to 100 command line parameters.</p> <p>The default value is reject.</p>
protect_receivedir	<p>Defines the processing performed when executing a receive directory and Platform Server issues a "cfsend trtype:c rcmd: '_cfdir '" command to request a directory list.</p> <p>The valid values are:</p> <p>none: No additional parsing is performed when Platform Server executes a system command.</p> <p>reject: If a command to be executed includes any of the characters defined by the rejectcmdcharacters parameter, the command terminates with an error.</p> <p>exec: The command to be executed calls the exec function. This call does not allow multiple commands to be executed in a single command string.</p> <p>The default value is exec.</p>
rejectcmdcharacters	<p>Defines the characters that are validated when reject or token is defined for a parameter. When one of these parameters is in a command or token, the command terminates with an error.</p> <p>The valid values are up to 10 characters.</p> <p>The default value is ; & .</p>

Extended Features

TIBCO MFT Platform Server provides various features to meet different transfer requirements. See the following list for the extended features of TIBCO MFT Platform Server:

- [Checkpoint Restart](#)
You can use this feature to perform a checkpoint transfer at a specified time interval.
- [Conversion Tables/Custom Code Conversion](#)
You can use this feature to convert text files between various character-set specifications.
- [Directory Named Initiation \(DNI\)](#)
You can use this feature to have the files in a directory transferred automatically to one or more targeted TIBCO MFT Platform Server.
- [fusing Utility](#)
You can use this feature to determine if a remote platform server is running.
- [fusutil Utility](#)
You can use this feature to rename, move or delete a transferred file on a remote platform.
- [Configured Post Processing](#)
You can use this feature to configure post processing actions for all transfers.
- [CfAlias](#)
You can use this feature to change the name or location of a transferred file.
- [Auditing \(cfinq Utility\)](#)
You can use this feature to view the status of all completed transfers along with a detailed list of all transfer parameters.
- [Access Control](#)
You can use this feature to transfer files directly to the predefined directory.
- [CRL Support](#)
You can use this feature to ensure the certificates have not been revoked when performing SSL transfers.
- [Configured SSL Authorization](#)
You can use this feature to personalize the SSL authorization to determine if the certificates can be accepted or rejected when performing SSL transfers.
- [User Exits](#)
You can use this feature to build additional processes on top of advanced file transfer capabilities to exit the programs.
- [cfunix2dos.exe Utility](#)
You can use this feature to convert a file from UNIX format to DOS format.
- [TIBCO Accelerator](#)
You can use the TIBCO Accelerator technology to improve data transfer speed over IP network connections (high bandwidth, high latency).



TIBCO MFT Platform Server provides support for Docker container deployment. For details, see *TIBCO Managed File Transfer Platform Server for UNIX Docker Container Deployment*.

Checkpoint Restart

With this feature, TIBCO MFT Platform Server can take a checkpoint at a user-specified interval for a transfer.

A checkpoint contains information about the transfer such as file pointers, byte count and compressed byte count. In case of a failure, the transfer does not restart at the beginning of the file but starts at the last checkpoint stored. This feature is especially useful in the case where the network connections are slow or the file you want to transfer is large.

To use checkpoint restart, you have to define the following three parameters:

- **TryNumber**: specifies a try count for each initiated transfer before it will fail with a permanent error .
- **RetryInterval**: determines how long the initiator waits before the next retry of the transfer. By default, this interval is set to 1 minute.
- **CheckPointInterval**: defines how often a checkpoint is taken. By default, this interval is set to 1 minute.

For more information, see [TryNumber](#), [RetryInterval](#) and [CheckPointInterval](#).

Details of every checkpoint transfer and every initiated transfer that has the **TryNumber** parameter defined are stored in a Persistent Queue File (PQF) file. This file is stored in the `$CFROOT/PQF` directory and is deleted once a transfer is completed successfully or has exceeded the **TryNumber** parameter.

Note the following points when doing a checkpoint transfer:

- If the initiator `cfsend` or `cfrecv` is shut down during a checkpoint restart transfer, `CyberResp` has to be stopped and restarted so that it can pick up the PQF file to restart that transfer.
- If a checkpoint transfer fails when UNIX is a responder, the PQF file is not deleted.
- When `CyberResp` is running by a non-root user, the transfer can only be restarted by the same non-root user. Otherwise, the following message is displayed:

Transfer cannot be restarted because CyberResp user is different than Transfer.

Checkpoint Restart Example

On the initiator, you can perform a send transfer with the following configurations:

```
TryNumber: 3
RetryInterval: 2
CheckPointInterval: 1
```

After this transfer is in process for over a minute, if the connection to the responder is down for some reason, the initiator tries three times to restart the transfer from this checkpoint. Because the **RetryInterval** parameter is set to 2 minutes, TIBCO MFT Platform Server waits for 2 minutes between each retry attempt.

If the initiator is down, then you can restart the process by restarting the `CyberResp` daemon, so it can scan the PQF file and attempt to restart transferring any files that are in the PQF directory.

Conversion Tables/Custom Code Conversion

With this feature, you can convert text files between various character-set specifications.

The character conversion tables are for single byte conversion. If you need to do double byte conversions, there are two options:



1. Convert the file using the iconv utility, then send the file as a binary file.
2. Send a file to Platform Server for z/OS, then use Platform Server for z/OS to perform double byte conversion. For more information on double byte conversion, see LCT and RCT parameters in the *TIBCO Managed File Transfer Platform Server for z/OS User's Guide*.

The platform server provides the following four conversion tables:

Conversion Table	Description
Comtblg.classic	The old comtblg.dat shipped with previous versions (before version 7.1).
Comtblg.cp037	Extended ASCII table that is based on <i>IBM Code</i> page 037.
Comtblg.cp1047	Extended ASCII table that is based on <i>IBM Code</i> page 1047.
Comtblg.dat	ASCII/EBCDIC table used by the platform server at run time. (By default a copy of Comtblg.cp037.)

MFTPS_install/Comtblg.dat contains the following table which converts data from ASCII to EBCDIC and vice versa:

00010203372D2E2F16050A0B0C0D0E0F	ASCII-EBCDIC portion of the translation table
101112133C3D322618193F27221D351F	
405A7F7B5B6C507D4D5D5C4E6B604B61	
FOF1F2F3F4F5F6F7F8F97A5E4C7E6E6F	
7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6	
D7D8D9E2E3E4E5E6E7E8E9BAE0BBB06D	
79818283848586878889919293949596	
979899A2A3A4A5A6A7A8A9C04FDOA107	
9F000000000000000000000000000000	
00000000000000000000000000000000	
41AA4AB100B26AB5BDB49A8A5FCAAFBC	
908FEAFABEA0B6B39DDA9B8BB7B8B9AB	
6465626663679E687471727378757677	
AC69EDEEEBEFECBF80FDFEFBFCADAE59	
4445424643479C485451525358555657	
8C49CDCECBCFCCE170DDDED8D8EDF	
002E2E2E2E2E2E2E2E2E2E2E2E2E2E	EBCDIC-ASCII portion of the translation table
2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E	
2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E	
2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E	
2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E	
20A0E2E4E0E1E3E5E7F1A22E3C282B7C	
26E9EAE8E8E8E8E8E8E8E8E8E8E8E8	
2D2FC2C4C0C1C3C5C7D1A62C255F3E3F	
F8C9CACBC8CDCECFCC603A2340273D22	
D8616263646566676869ABBBF0FDFEB1	
B06A6B6C6D6E6F707172AABAE6B8C680	
B57E737475767778797AA1BFD0DDDEAE	
5EA3A5B7A9A7B6BCBDBE5B5DAFA8B4D7	
7B414243444546474849ADF4F6F2F3F5	
7D4A4B4C4D4E4F505152B9FBFCF9FAFF	
5CF7535455565758595AB2D4D6D2D3D5	
30313233343536373839B3DBDCD9DA2E	

To activate conversion tables, you have to turn the `ASCII_to_EBCDIC` parameter on. For more information, see [ASCII_to_EBCDIC](#) parameter in z/OS Specific Transfer Parameters.

When this parameter is on, it uses the file names that are specified in the `ConvTbl`, `LocalCTFile`, and `RemoteCTFile` parameters. For more information, see [CONVTBL](#), [LocalCTFile](#) and [RemoteCTFile](#) in Optional Transfer Parameters.

You can define the `ConvTbl` parameter in the `config.txt` file to specify the default conversion table for all transfers. If this parameter is not set, the `$CFROOT/Comtblg.dat` file is used.

You can specify two conversion tables: one on the local side, and one on the remote side. In this way, you can have a standard character set for all types of transfer, without having a conversion table between every two possible character sets. To identify which table translates for send and which translates for receive during editing, it is good practice that you place a few lines between the two tables.

You can define the local conversion table using the `LocalCTFile` parameter in a transfer template or the `lct` parameter on the command line. Similarly, you can define the remote conversion table using the `RemoteCTFile` parameter in a transfer template or the `rct` parameter on the command line. The maximum lengths of the `lct` and `rct` parameters are both 16 characters. However, they support file names relative to the current working directory and the `$CFROOT` directory on the local side.

- For UNIX, the directories are searched in the following order: the directory where CyberResp is running on the remote side, the `$CFROOT` environment variable if it is defined in the environment of the CyberResp process, and then in the `/mftps` directory.
- For z/OS, the platform server searches an in-core table that has to be enabled at startup or through an operator command.

- For Windows, the platform server searches in the working directory.



If both the **ConvTbl** and the **LocalCTFile** parameters are specified, the **LocalCTFile** parameter overrides the **ConvTbl** parameter. The platform server does not convert the file twice. The **ConvTbl** parameter is not affected by **RemoteCTFile**.

For a File Send, the top half of the conversion table is used. For a File Receive, the bottom half of the conversion table is used. For example, in a send transfer, if both the **LocalCTFile** and **RemoteCTFile** parameters are used, then the top half of local conversion table file is used on the local side, and the bottom half of remote conversion table file is used on the remote side. The reverse is true for a receive transfer.

Nodes can also support both local and remote conversion tables. Unless the parameters are overridden on the command line, conversion tables are used whenever that node is specified.

ASCII to EBCDIC Conversion Table Example

See the following ASCII to EBCDIC table. The EBCDIC to ASCII table works the same way.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	37	2D	2E	2F	16	05	0A	0B	0C	0D	0E	0F
1	10	11	12	13	3C	3D	32	26	18	19	3F	27	22	1D	35	1F
2	40	5A	7F	7B	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	AD	E0	BD	5F	6D
6	79	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	C0	6A	D0	A1	07
8	9F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A	41	AA	4A	B1	00	B2	6A	B5	BD	B4	9A	8A	5F	CA	AF	BC
B	90	8F	EA	FA	BE	A0	B6	B3	9D	DA	9B	8B	B7	B8	B9	AB
C	64	65	62	66	63	67	9E	68	74	71	72	73	78	75	76	77
D	AC	69	ED	EE	EB	EF	EC	BF	80	FD	FE	FB	FC	AD	AE	59
E	44	45	42	46	43	47	9C	48	54	51	52	53	58	55	56	57
F	8C	49	CD	CE	CB	CF	CC	E1	70	DD	DE	DB	DC	8D	8E	DF

Each ASCII or EBCDIC character is represented by 2 hexadecimal digits. For example, ASCII character E is hexadecimal 45 or X'45'. So to find the location of the ASCII character E within the table, you can go down to row 4. The second hexadecimal digit is 5, so you can move across to column 5. The point at which they meet is the hexadecimal value X'C5'. This means the hexadecimal EBCDIC value for E is X'C5'. If you want the E to be represented by a different EBCDIC hexadecimal value you can edit this value in this table. When a transfer is completed and the data is converted to EBCDIC, the new value is used.



The ASCII character set in the default table supports the extended ASCII range which covers special characters outside the English alphabet. For standard ASCII support, you can use the `comtblg.classic` file. To replace the default table, you can rename the existing `comtblg.dat` file, and then rename the existing `comtblg.classic` file to become the new `comtblg.dat` file. The conversion tables currently available do not support wide characters or multibyte character sets.

For other conversions besides standard ASCII to EBCDIC conversion, you can copy the default `comtblg.dat` file to create new customized tables of your own. You can assign conversion tables to the nodes. For more information, see [Transfers Using Nodes](#).



You must always replace a 2-digit hexadecimal number with a 2-digit hexadecimal number. If the table is invalid, conversion cannot be performed. The table consists of two sections with 16 lines each, therefore the entire file must have 32 lines across and 32 lines down. If it contains anything else, it does not work.

Directory Named Initiation (DNI)

With this feature, the platform server can detect the existence of files that are placed within a directory or sub directories, and automatically transfer those files to one or more targeted remote systems.

For more information, see *TIBCO Perl Directory Named Initiation (DNI) Installation and Operations Guide* contained within the `dni.tar` file, which is located in the `$CFROOT/dni` directory.

To extract the `dni.tar` file, on the command line, navigate to the `$CFROOT/dni` directory and then use the following `tar` command:

```
tar xvf dni.tar
```

fusping Utility

You can use the **fusping** utility in the `$CFROOT/bin` directory to determine if a platform server is running remotely.

See the following usage of the **fusping** utility:

```
usage: fusping parameters:[values]
[parameters]:
h: or Host and Port: - h:[IpAddress]:[PortNumber] or h:[IpName]:[PortNumber]
?: - Help
```

For example, you can determine the status of a remote z/OS server by using the following command:

```
fusping h:[11.22.33.55]:[46464]
```

See the following example of the output:

```
Host:          11.22.33.55
Port:          46464
System Name:   Name=A390,STC=CFUSN65,CPUType=1234,CPUID=5555
Key Expiration: 20351231
Version:       MFT Platform Server z/OS,Version=720,PTFLevel=CZ01977:720
```

For example, you can determine the status of a remote Windows server by using the following command:

```
fusping h:[11.22.33.44]:[46464]
```

See the following example of the output:

```
Host:          11.22.33.44
Port:          46464
System Name:   WIN44
Key Expiration: 20351231
Version:       Ftms32.DLL, Version 7.1.1 (Build 1042. PTF CW01972 UNICODE)
               FtmsDni.DLL, Version 7.1.1 (Build 1042. PTF CW01972 UNICODE)
               FtmsTcpS.DLL, Version 7.1.1 (Build 1042. PTF CW01972 UNICODE)
               FtmsVer.DLL, Version 7.1.1 (Build 1042. PTF CW01972 UNICODE)
               FusionMs.DLL, Version 7.1.1 (Build 1042. PTF CW01972 UNICODE)
               HoLib.DLL, Version 7.1.1 (Build 1042. PTF CW01972 UNICODE)
               HOTrace.DLL, Version 7.1.1 (Build 1042. PTF CW01972 UNICODE)
               SMTPDll.DLL, Version 7.1.1 (Build 1042. PTF CW01972)
               FtmsMgr.EXE, Version 7.1.1 (Build 1042. PTF CW01972 UNICODE)
               FtmsCmd.EXE, Version 7.1.1 (Build 1042. PTF CW01972 UNICODE)
               FtmsMon.EXE, Version 7.1.1 (Build 1042. PTF CW01972 UNICODE)
               FtmsSvr.EXE, Version 7.1.1 (Build 1042. PTF CW01972 UNICODE)
               FusionVer.EXE, Version 7.1.1 (Build 1042. PTF CW01972 UNICODE)
```

For example, you can determine the status of a remote UNIX server with IPv6 address by issuing the following command:

```
fusping h:[2001:0DB8:0000:0000:0000:0000:1428:0000]:[46464]
```

See the following example of the output:

```
[root@localhost mftps720]# fusping h:[2001:0DB8:0000:0000:0000:0000:1428:0000]:[46464]
Host:          2001:0DB8:0000:0000:0000:0000:1428:0000
Port:          46464
```



```

System Name:      localhost.localdomain
Key Expiration:   N\A.
Version:
Build 2           cfsend, MFT Platform Server Initiator Version 7.2 build 2. Maint
Build 2           cfrecv, MFT Platform Server Initiator Version 7.2 build 2. Maint
Build 2           cfdir, TIBCO Software Inc.
Copyright (c) 1995-2015 TIBCO Software Inc. ALL RIGHTS RESERVED. TIBCO Software Inc.
Confidential Information.
MFT Platform Server - cfdir utility
Version 7.2 build 2. Maint Build 2
Build 2           cfnode, TIBCO Software Inc.
Copyright (c) 1995-2015 TIBCO Software Inc. ALL RIGHTS RESERVED. TIBCO Software Inc.
Confidential Information.
MFT Platform Server - Node Definition Maintenance Utility
Version 7.2 build 2. Maint Build 2
Build 2           cfprofile, TIBCO Software Inc.
Copyright (c) 1995-2015 TIBCO Software Inc. ALL RIGHTS RESERVED. TIBCO Software Inc.
Confidential Information.
MFT Platform Server - Profile Maintenance Utility
Version 7.2 build 2. Maint Build 2
Build 2           cfrprofile, TIBCO Software Inc.
Copyright (c) 1995-2015 TIBCO Software Inc. ALL RIGHTS RESERVED. TIBCO Software Inc.
Confidential Information.
MFT Platform Server - Responder Profile Maintenance Utility
Version 7.2 build 2. Maint Build 2

```

fusutil Utility

When a file transfer is completed, you can use the **fusutil** utility to perform post processing actions, such as renaming, moving, or deleting a file.

Different operating systems support different commands. The **fusutil** utility provides a common interface to rename, move, or delete a file or directory, and to verify if a file or directory exists in a remote system. You can use the **fusutil** command as a post processing action running command.

The **rdir** | **renamedir**, **ddir** | **deletedir**, **rmdir** | **removedir**, **mvd** | **movedir** options are used to rename, delete, and move directories, while the **r** | **rename**, **d** | **delete**, **m** | **move** and **e** | **exist** options are used to rename, delete, move, and verify the existence of files.



- The maximum length of the **fusutil** command is 1024 characters.

See the following post processing command examples using each of the **fusutil** utility options:

Post_Action1: S,R,COMMAND,fusutil E *filename*

Post_Action2: S,L,COMMAND,fusutil D *filename*

Post_Action3: S,R,COMMAND,fusutil M *old_filename new_filename*

Post_Action4: F,R,COMMAND,fusutil R *old_filename new_filename*

Post_Action2: S,L,COMMAND,fusutil DDIR *directoryname*

Post_Action3: S,L,COMMAND,fusutil RMDIR *directoryname*

Post_Action4: S,R,COMMAND,fusutil MVDIR *directoryname new_directoryname*

Post_Action5: F,R,COMMAND,fusutil RDIR *old_directoryname new_directoryname*



- When processing the `EXIST` option, the code also checks if the file is available for use. This can be completed on all platforms except UNIX, because there is no standard call to accomplish this on UNIX.
- When using the `RENAME` or `DELETE` option on UNIX in a directory to which you have write access, it is possible to remove or rename a file that does not belong to you. This is a function of how UNIX security works.
- The `ddir | deletedir` option deletes non-empty directory recursively, while the `rmdir | removedir` option removes an empty directory only.

The results of the used commands are returned in codes. See the following table for the meanings of return codes:

Return Code	Description
0	Success.
4	General network errors and the command will be retried.
8	Severe error. The command will not be retried.
Any other return code	Check the return code message for more information.

Configured Post Processing

With this feature, you can configure post processing actions for all transfers.



For information on defining different post processing actions for different transfers, see [Post Processing Actions \(PPA\)](#).

After a transfer is completed, TIBCO MFT Platform Server searches a configuration file containing the commands and the associated parameters. If the properties of the transfer match the parameters, then the command is triggered. This offers greater flexibility than user exits through the use of parameters and argument substitution.

If any redirecting is done, on the initiator side the redirecting goes to the directory from which the `cfsend` command is used. On the responder side, it goes to the directory from which CyberResp is running.


TIBCO MFT Platform Server installs a sample configured post processing file named `CfgPostProc.cfg` in the `$CFROOT/config` directory.

See the following table for the required parameters:

Parameter	Description
SUBMIT	Identifies the start of the parameters.
COMMAND	Defines the command you want to execute.

See the following table for parameters which set up the criteria the transfer has to meet before the configured post processing action is run:

Parameter	Description
TYPE	Defines the type of the file transfer request. The valid values are <code>SEND</code> , <code>RECEIVE</code> , or <code>BOTH</code> .

Parameter	Description
SOURCE	Defines the source of the file transfer request. The valid values are INITIATOR , RESPONDER , or BOTH .
STATUS	Defines whether a transfer request is successful or unsuccessful. The valid values are SUCCESS , FAILURE , or BOTH .
FILENAME or DSN	Defines the absolute path of the local file name. It is compared against the local file name in the file transfer request.
PROCESS	Defines the process name associated with the transfer request. The maximum length of the defined value is 8 characters.  This only applies to z/OS transfers.
IPADDR	Defines the IP address of the machine that communicates with TIBCO MFT Platform Server.
NODE	Defines the node name in the transfer request. For initiator requests, this parameter is used when the NODE parameter is used in a transfer request. For responder requests, TIBCO MFT Platform Server scans the list of nodes for matches on the IP address. These entries are then matched against the value specified in this NODE parameter.



If a parameter is not defined, it is considered as a match. If all parameters match, the file in the **command** is transferred.

Argument Substitution

You can pass transfer properties to the executable command as substitutable command line arguments.

You can enter any of the following listed argument names after the **COMMAND** entry in the configuration file.

Argument Name	Data Substituted
&TYPE	Send or Receive
&SOURCE	Initiator or Responder
&STATUS	Success or Failure
&RC	Numeric return code (0 if successful)
&FILENAME or &DSN	Local file name
&PROCESS	Process name
&NODE	Node name (or NODE if no node can be found)

Argument Name	Data Substituted
&IPADDR	IP address (or IPADDR if no IP address can be specified)
&TRN	Local transaction number

In the following example, the file name and type of the transfer request are substituted for the **&FILENAME** and **&TYPE** arguments and passed to the executable as command-line arguments.

COMMAND=cmdfile.com &FILENAME &TYPE,

Configured Post Processing Command Examples

You can configure the parameters in the `CfgPostProc.cfg` file according to your end goal.

See the following two configured post processing examples:

```
SUBMIT,COMMAND=loaddb --filename &FILENAME -source &IPADDR,
TYPE=RECEIVE,
STATUS=SUCCESS, SOURCE=RESPONDER,
FILENAME=jan.sales,
NODE=ACCOUNTING,
PROCESS=cfusion
SUBMIT,COMMAND=cmdfile,TYPE=SEND,
STATUS=BOTH, SOURCE=INITIATOR,
FILENAME=infile.txt,
IPADDR=111.222.33.44
```

CfAlias

If you are an administrator, with this feature, you can associate an alias with an actual fully qualified file name, so the user does not know the actual file name used in the system.

Some architectures do not want users to know the file names or locations of the files they send to the server. Or the administrator wants to handle file naming and location automatically for users. TIBCO MFT Platform Server supports substitutable parameters that can be used to assign values to file names on the responder side.




This feature is only supported on the responder side.


CfAlias Parameters

TIBCO MFT Platform Server provides a sample alias file called `CfAlias.cfg` in the `$CFROOT/config` directory.



In `CfAlias.cfg` file, one parameter is defined in each line, and continuations are defined by a comma followed by a space. You can set the path for the `CfAlias.cfg` file using the **AliasConfig** parameter in the `config.txt` file. For more information, see [AliasConfig](#) in Common Configuration Parameters.

See the following table for the required `CfAlias` parameters. At least, you must define either the **USERID** parameter or the **NODE/IPADDR** parameters.

Parameter	Description
USERID	<p>Defines the user ID of the user who initiates the transfer request.</p> <p>The valid values are <i>userId</i> or DEFAULT.</p> <div>  <p>DEFAULT indicates a match with any user.</p> </div>

Parameter	Description
NODE	<p>Defines the name of the node on which the transfer request is initiated.</p> <p>The valid values are <i>nodename</i> or DEFAULT.</p> <div>  <div>DEFAULT indicates a match with any node.</div> </div>
IPADDR	<p>Defines the IP address of the server on which the transfer request is initiated.</p>

See the following table for parameters which set up the criteria the transfer has to meet before CfAlias is applied:

Parameter	Description
TYPE	<p>Defines the type of transfer request.</p> <p>The valid values are SEND, RECEIVE, or BOTH.</p> <div>  <div>A send transfer to the initiator is considered as a receive transfer to the responder.</div> </div>
FILE	<p>Defines the fully qualified name you want to use instead of the alias file.</p>
ALIAS	<p>Defines the actual name of the file requested by the initiator.</p>
ALLOW	<p>Defines whether the user who initiates the transfer can define the actual file name if no match is found.</p> <div>  <div>ALLOW and FILE/ALIAS are mutually exclusive.</div> </div> <p>The valid values are:</p> <ul style="list-style-type: none"> • NO: the FILE and ALIAS parameters must be defined. • YES: the FILE and ALIAS parameters must not be defined.

When creating a CfAlias, you must define either **NODE/IPADDR** or **USERID**. Then you can define what type of transfer request you want to monitor. Then finally set **ALLOW** to decide whether the initiator user can define file name on the responder. If the parameters of a send transfer do not match any entry in the CfAlias.cfg file, the transfer is rejected.

Substitutable Parameters

The administrator can define substitutable parameters in the **FILE** parameter of the CfAlias file.

Substitutable parameters are defined by a percent sign (%) followed by the parameter name. See the following table for the supported substitutable parameters:

Substitutable Parameters	Description
%JDATE	Julian date (YYDDD)

Substitutable Parameters	Description
%JDATEC	Julian date (CCYYDDD)
%GDATE	Gregorian date (YYMMDD)
%GDATEC	Gregorian date (CCYYMMDD)
%TIMET	Time (HHMMSST)
%TIME	Time (HHMMSS)
%NODE	Node name (If no node is defined, use the value NODE.)
%USER	User name
%TRN	Transaction number
%SYSID	System name
%ACB	VTAM ACB name (z/OS only)

For example, `FILE=/u/prtom/abc123.20090718.1601029` can be substituted as `FILE=/u/%USER/abc123.%GDATEC.%TIMET`.

Examples: Using CfAlias

You can configure the parameters in the `CfAlias.cfg` file according to your end goal.

A daily report named `report.doc` is received by TIBCO MFT Platform Server running on a UNIX server everyday from a remote TIBCO MFT Platform Server user named JohnDoe. The user sends a new report each day and the `report.doc` sent on the previous day is replaced with the new report file. The UNIX administrator wants to prevent the existing `report.doc` from being replaced without involving JohnDoe.

To solve this, the administrator can simply set up two CfAlias groupings in the `CfAlias.cfg` file as follows:

```

USERID=JohnDoe,
NODE=DEFAULT,
TYPE=RECEIVE,
FILE=/home/JohnDoe/DailyReports/report.%GDATE.doc,
ALIAS=report.doc
*
USERID=JohnDoe,
NODE=DEFAULT,
ALLOW=NO

```

With the settings configured in the first CfAlias grouping set, when JohnDoe sends in his daily report, it is put in the following directory with a new file name each day based on the current date:

```
/home/JohnDoe/DailyReports/report.%GDATE.doc
```

For example, if the date is July 18, 2009, the file can be created as `report.090718.doc`. JohnDoe has no knowledge of where or how his report is stored. Also, note the `TYPE=RECEIVE` setting, this is because a receive transfer on the responder is a send transfer from the initiator. Finally, the second CfAlias grouping restricts JohnDoe from having any other access to any file that is not `report.doc`.

Auditing (cfinq Utility)

TIBCO MFT Platform Server writes log files to store transfer parameters and the values for all transfer requests for auditing purposes. The **cfinq** utility provides a way to view this information. The audit records can also be viewed through the Command Center Search Audits and Audit Polling capability.

Log Files

TIBCO MFT Platform Server has comprehensive logging located in the `$CFROOT/log` directory. You can find records of all transfer requests performed on the server.

The **cfinq** utility provides two ways of viewing the audit information: the summary view and the detailed view. The summary view only consists of the following columns: Index, Transaction, Status, IP Address and Local File.



- To obtain all transactions in the specified query, you must either use the root account, or a member of the `cfbrowse` group or `cfadmin` group as defined in the `$CFROOT/config.txt` file. Without this access, you can only view your own transactions.
- The `cfbrowse` group is used only for auditing purposes and does not have other rights that a user in the `cfadmin` group has.

The daily audit log is called `Log.txt.yyyymmdd`. Each day a new log is generated with the date appended to the end of the file name. You can change the path and log prefix by editing the **LogEventFileName** parameter in `config.txt` file. This log file is a standard ASCII text file which contains one record on each line.

The following sample `Log.txt` shows one transfer request log information:



```
VersionNumber=8.0. Maint build 4, Priority=N/A, LocalTranNumber=R829800336,
RemoteTranNumber=1829800335, TransferStartTime=163525, TransferStartDate=20180829,
TransferCompletionTime=163525, TransferCompletionDate=20180829,
TransferEndTime=163525 , TransferEndDate=20180829, TransferDirection=Receive,
TransferWork=File, TransferCommand=N/A,
TransferProcessName=N/A, TransferScheduleDate=N/A, TransferScheduleTime=N/A,
TransferExpirationDate=N/A, TransferExpirationTime=N/A, CompressionType=Non e,
CompressedBytes=N/A,
ConvertCRLF=no, EBCDICTranslate=no, TLS=no, EncryptionType=N/A, RecordFormat=N/A,
FileCreateOptions=Create, FileAttributes=N/A, UNIXFile Permissions=644,
EmailSuccessAddr=N/A,
EmailFailureAddr=N/A, Allocation Type=N/A,AllocationDirectory=N/A,
AllocationPrimary=N/A, AllocationSecondary=N/A, Volume=N/A, Unit=N/A, Nodeclass=N/A,
Storclass=N/A, Mgtclass=N/A,
Dataclass=N/A, BlockSize=0, RecordLength=0, UserData=N/A, LogonDomain=N/A,
LocalFileName=/root/vani/bulk_recv/sslkeysand certs, LocalUserid=root, RemoteFileName=/
root/ravindra/sslkeysandcerts,
RemoteUserid=root, RemoteNodeName=10.108.80.84, RemoteNodeType=IpName, Remote
PortNumber=N/A, TryCount=0, TryMaxCount=0, ByteCount=32, RecordCount=N/A,
MemberCount=N/A, CheckPointCount=0,
CheckpointRestart=no, CheckPointInterval=0, StatusMsg=Transfer Completed, CrlMsg=N/A,
StatusDiagCode=00, StatusSeverity=00, StatusReturnCode=N/A, TransferStatus=Success,
LocalFile=N/A,
RemoteCTFile=N/A, TempError=No, PPA1Action=N/A, PPA1Source=N/A, PPA1Status=N/A,
PPA1Data=N/A, PPA1ReturnCode=N/A, PPA2Action=N/A, PPA2Source=N/A, PPA2Status=N/A,
PPA2Data=N/A,
PPA2ReturnCode=N/A, PPA3Action=N/A, PPA3Source=N/A, PPA3Status=N/A, PPA3Data=N/A,
PPA3ReturnCode=N/A, PPA4Action=N/A, PPA4Source=N/A, PPA4Status=N/A, PPA4Data=N/A,
PPA4ReturnCode=N/A,
Accelerator=N/A, ACCProtocol=N/A, ACCEncryption=N/A, ACCCompression=N/A,
ACCMaxSpeed=N/A, ACCHost=N/A, ACCPort=N/A, SecurityPolicy=None,
RemoveTrailingSpaces=N, ScanSubDir=N/A,
ClassOfService=Default, CRC=N/A, TLSProtocol=N/A, TLSCipher=N/A,
```




cfinq Parameters

See the following table for parameters supported by the **cfinq** utility.



- The **cfinq** utility does not accept any negative values.
- Navigation commands are case sensitive.

Parameter (Alternate Specification)	Description
DAYS	<p>Defines the number of days to search.</p> <p> DAYS must not exceed 1826 (5 years).</p> <ul style="list-style-type: none"> • If SDATE and EDATE are both defined, DAYS is ignored. • If SDATE is not defined, Start Date = Current Date - number of Days. • If SDATE is not defined and EDATE is defined, Start Date = EDATE - number of Days.
DESCRIPTION (DESCR)	<p>Defines the user data.</p> <p>Based on the DESCRIPTION parameter, the cfinq utility can search the log files and present detailed information for any transfers matching that description.</p> <p>A message is displayed on the screen if there are no transactions specified for the DESCRIPTION.</p>
ENDDATE (EDATE)	<p>Defines the end date in the format of yyyymmdd.</p> <ul style="list-style-type: none"> • EDATE=TOD or EDATE=TODAY means today. • EDATE=YES or EDATE=YESTERDAY means yesterday. <p>The default is TODAY.</p> <p> To use ENDDATE, you must define STARTDATE or DAYS.</p>
ENDTIME (ETIME)	<p>Defines the end time in the 24 hour format of hhmmss.</p> <p>The default value is 240000.</p> <p>If STIME is not defined, the cfinq utility searches for the transaction only within the 000000 - ETIME period.</p>
EXCEPTIONS (EXC)	<p>Defines the type of transfers to select.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • U: unsuccessful • S: successful • Default: successful or unsuccessful
LOCALFILE (LF)	<p>Defines the local file name.</p>

Parameter (Alternate Specification)	Description
LOCALUSER (LUSER)	<p>Defines the local user name (user ID).</p> <p> If you specify a user name other than your own, you must have the appropriate security authorization.</p>
LOCTRANSNUM (LTRN)	<p>Defines the unique local transaction number of the transfer.</p> <p>Based on the LOCTRANSNUM parameter, the cfinq utility can search the log files and present the detailed information for that transaction number. A message is displayed on the screen if no transaction for the LOCTRANSNUM is specified.</p>
LOGDIR (LOGD)	Defines the log files directory.
MAXXFER (MAX)	<p>Defines the maximum number of requests that can be returned.</p> <p>The default number is 500. The valid values are from 1 to 100,000.</p> <p> The oldest transfer information is gathered first. If the total number of records exceeds 10000, the information close to the SDATE is not included in the transaction list.</p>
PROCESS (PRO)	Defines the process name.
REMHST (RHOST)	<p>Defines the remote system name.</p> <p>This can be a node name, SNA LU Name, host name or IP address in dotted decimal format.</p> <p> Generic selection cannot be used for IP addresses.</p>
REMTRANSNUM (RTRN)	Defines the remote transaction number.
STARTDATE (SDATE)	<p>Defines the start date of the search in the format <code>yyyymmdd</code>.</p> <ul style="list-style-type: none"> • SDATE = TOD or SDATE = TODAY means today. • SDATE = YES or SDATE = YESTERDAY means yesterday. <p>The default value is TODAY.</p>
STARTTIME (STIME)	<p>Defines the start time in 24 hour format of <code>hhmmss</code>.</p> <p>The default is 000000. If ETIME is not defined, the cfinq utility searches for the transaction only within the STIME - 240000 period.</p>
TEMPERROR (TMPERR)	<p>Defines whether to print the request for temporary errors that are in the audit file.</p> <p>This parameter applies regardless of whether the Print parameter is defined or not. The valid values are Yes or No. The default value is No.</p>

Examples: Using cfinq Utility

You can run the **cfinq** utility from the `$CFROOT/bin` directory on the command line to obtain transfer information.

The **cfinq** utility accepts parameters on the command line. You can specify the criteria to be met to provide a detailed query of the TIBCO MFT Platform Server records. The following example queries records of successful transfers only for the local user named JohnDoe over a 20 day time span starting from September 9, 2012 at 9:01 am and ending at 3 pm, with a maximum of 1000 records listed.

```
cfinq sdate:20120909 days:20 stime:090100 etime:150000 luser:JohnDoe lf:/unix/SERVER/
log EXC:S max:1000
```



Use either an equal sign or a colon to separate the parameter from the value. You must not insert any space between the parameter name, the equal or colon sign, and the parameter value.

You can also use the **cfinq** menu to select which transfer information is displayed. The following menu is displayed after entering the **cfinq** command on the command line.

```
*****
YOU HAVE ENTERED THE FOLLOWING VALUES FOR YOUR INQUIRY:

LOCTRANSNUM.....[]
REMTRANSNUM.....[]
LOGDIR.....[]
STARTDATE.....[]
ENDDATE.....[]
DAYS.....[]
STARTTIME.....[]
ENDTIME.....[]
MAXXFER.....[]
LOCALFILE.....[]
LOCALUSER.....[]
REMHOST.....[]
DESCRIPTION.....[]
PROCESS.....[]
EXCEPTIONS.....[]
TEMPERROR.....[]
INITRESPFLAG.....[]
*****
***   PRESS [q] [enter] TO QUIT THE PROGRAM   ***
***   PRESS [a] [enter] TO OBTAIN WHOLE RECORD LIST   ***
***   PRESS [c] [enter] TO OBTAIN CURRENT RECORD LIST   ***
***   PRESS [p] [enter] TO OBTAIN PREVIOUSLY VIEWED RECORD LIST   ***
***   PRESS [m] [enter] TO OBTAIN MENU SCREEN   ***
***   PRESS [n] [enter] or [enter] TO OBTAIN NEXT RECORD LIST   ***
***   PRESS [h] or [?] [enter] TO OBTAIN HELP SCREEN   ***
***   PRESS [index #] [enter] TO OBTAIN DETAILED RECORD INFORMATION   ***
*****
==>
```

You can enter a single letter to obtain record listings. For example, if you enter "a" and hit Enter. The following results are displayed:

```
*****
INDEX    TRANSACTION    STATUS    IPADDRESS    LOCALFILE
*****
1 I113500000 Success 127.127.127.0:46464 /home/a.txt
2 I113500001 Success 127.127.127.0:46464 /home/remotefile
3 I113500002 Success 127.127.127.0:46464 /home/localfile
4 I113500003 Success 127.127.127.0:46464 /home/a.txt
5 I113500004 Success 127.127.127.0:46464 /home/tmp/CG.DAT
6 I113500005 Success 127.127.127.0:46464 /home/tmp/CLEAN.EXE
7 I113500006 Success 127.127.127.0:46464 /home/tmp/RUN.BAK
8 I113500007 Success 127.127.127.0:46464 /home/tmp/HOOK.REG
9 I113500008 Success 127.127.127.0:46464 /home/tmp/RUN.BAK
10 R113500009 Success 127.127.127.0:46464 /home/tmp/WAKE.EXE
11 R113500002 Success 127.127.127.0:46464 /home/tmp/EXPRESS.INI
==>
```

If you want to view transaction number I113500002, you can enter "3" (the index number for transaction 3). The following detailed report is displayed:

RECORD:3

```

Version Number..... 7.2 build 4. Maint Build 4
Priority..... N/A
Local Transaction Number... I113500002
Remote Transaction Number... R113500003
Transfer Start Time..... 160701
Transfer Start Date..... 20150113
Transfer End Time..... 160705
Transfer End Date..... 20150113
Transfer Direction..... Send
Transfer Work..... File
Transfer Command..... N/A
Transfer Process Name..... N/A
Transfer Schedule Date..... N/A
Transfer Schedule Time..... N/A
Transfer Expiration Date... N/A
Transfer Expiration Time... N/A
Compression Type..... None
Compressed Bytes..... 0
Convert CRLF..... no
EBCDIC Translate..... no
SSL..... no
SSL Port Number..... N/A
Encryption Type..... N/A
Record Format..... FixedBlock
File Create Options..... CreateReplace
File Attributes..... N/A
UNIX File Permissions..... 644
Allocation Type..... N/A
Allocation Directory..... N/A
Allocation Primary..... N/A
Allocation Secondary..... N/A
Volume..... N/A
Unit..... N/A
Stor Class..... N/A
Mgt Class..... N/A
Data Class..... N/A
Block Size..... 0
Record Length..... 80
User Data..... N/A
Logon Domain..... N/A
Local File Name..... /home/localfile
Local User ID..... root
Remote File Name..... /home/remotefile
Remote User ID..... root
Remote Node Name..... 127.127.127.0
Remote Port Number..... 46464
Try Count..... 1
Try Max Count..... 1
Byte Count..... 17
Record Count..... N/A
Member Count..... N/A
Check Point Count..... N/A
Check Point Restart..... no
Check Point Interval..... 0
Status Msg..... File Transfer Complete
Crl Msg..... N/A
Status Diag Code..... 00
Status Severity..... 00
Status Return Code..... N/A
Transfer Status..... Success
Node Class..... N/A
Remote Node Type..... N/A
LocalCTFile..... N/A
RemoteCTFile..... N/A
PPA1 Action..... N/A
PPA1 Source..... N/A

```

```
PPA1 Status..... N/A
PPA1 Data..... N/A
PPA1 Return Code..... N/A
PPA2 Action..... N/A
PPA2 Source..... N/A
PPA2 Status..... N/A
PPA2 Data..... N/A
PPA2 Return Code..... N/A
PPA3 Action..... N/A
PPA3 Source..... N/A
PPA3 Status..... N/A
PPA3 Data..... N/A
PPA3 Return Code..... N/A
PPA4 Action..... N/A
PPA4 Source..... N/A
PPA4 Status..... N/A
PPA4 Data..... N/A
PPA4 Return Code..... N/A
Temporary Error..... No
Email Success Address..... N/A
Email Failure Address..... N/A
Accelerator..... N/A
Accelerator Protocol..... N/A
Accelerator Encryption..... N/A
Accelerator Compression..... N/A
Accelerator MaxSpeed..... N/A
Accelerator Host..... N/A
Accelerator Port..... N/A
Security Policy..... None
Remove Trailing Spaces..... N
Scan Subdirectories..... N
ClassOfService..... Default
*****
===>
```

Access Control

Using the access control feature, you can send a file directly to a predefined directory.

You can change the default directory for a file transfer based on access control parameters, such as **USERID**, **NODE** and **IPADDR**. For more information, see [Access Control Parameters](#).




You can only use this feature for responder transfers.



Access Control Parameters


You can change the parameters in an access control file to define a default directory to transfer files.


TIBCO MFT Platform Server provides a sample access control file called `AccessControl.cfg` in the `MFTPS_install/config` directory.


See the following table for supported parameters in an access control file:

Parameter Name	Description
USERID	Defines the local user ID. Use DEFAULT to indicate that this is the default value for a system.  You must specify either the USERID or NODE/IPADDR parameter. And you can specify both USERID and NODE/IPADDR .

Parameter Name	Description
NODE	<p>Defines the node name.</p> <p>Use DEFAULT to indicate that this is the default value for a system.</p> <div>  <p>You must specify either the USERID or NODE/IPADDR parameter. And you can specify both USERID and NODE/IPADDR. This parameter is mutually exclusive with the IPADDR parameter.</p> </div>
IPADDR	<p>Defines the IP address.</p> <div>  <p>You must specify either the USERID or NODE/IPADDR parameter. And you can specify both USERID and NODE/IPADDR. This parameter is mutually exclusive with the NODE parameter.</p> </div>
DESCRIPTION	<p>Defines a description as a user comment for this access control, at a maximum, the length of the description is 32 bytes.</p>
SEND_DIR	<p>Defines the default directory to locate the files you want to send to another system.</p> <p>This parameter has no default value.</p>
RECEIVE_DIR	<p>Defines the default directory to locate the files you want to receive from another system.</p> <p>This parameter has no default value.</p>
COMMAND_DIR	<p>Defines the default directory to locate where you want to use the commands in this system.</p> <p>This parameter has no default value.</p>

Parameter Name	Description
SEND_OPTION	<p>Defines the options for sending files.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • ROOT If you have specified a directory in a transfer command, then the directory is appended to the directory defined by the SEND_DIR parameter. • FORCE If you have specified a directory in a transfer command, then the directory is changed to the directory defined by the SEND_DIR parameter. The directory name defined in the request is ignored. The file name is appended directly to the directory defined by the SEND_DIR parameter. • ALLOW If you have specified a directory in a transfer command, the directory is used. If you have not specified a directory, then it is changed to the directory defined by the SEND_DIR parameter. • REJECT If you have specified a directory in a transfer command, then the file transfer terminates with errors. Otherwise, data is processed from the directory defined by the SEND_DIR parameter. • NEVER The user defined with the NODE or USERID parameter can never send files. • USE If you have specified a directory in a transfer command, then the directory is used. If you have not specified a directory, then the directory where the user exists when starting the previous CyberResp (the \$PWD environment variable for CyberResp) is used. <p> If SEND_OPTION is not specified, USE is the default setting.</p>

Parameter Name	Description
RECEIVE_OPTION	<p>Defines the options for receiving files.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • ROOT If you have specified a directory in a transfer command, then the directory is appended to the directory defined by the RECEIVE_DIR parameter. • FORCE If you have specified a directory in a transfer command, then the directory is changed to the directory defined by the RECEIVE_DIR parameter. The directory name defined in the request is ignored. The file name is appended directly to the directory defined by the RECEIVE_DIR parameter. • ALLOW If you have specified a directory in a transfer command, the directory is used. If you have not specified a directory, then it is changed to the directory defined by the RECEIVE_DIR parameter. • REJECT If you have specified a directory in a transfer command, then the file transfer terminates with errors. Otherwise, data is processed from the directory defined by the RECEIVE_DIR parameter. • NEVER The user defined with the NODE or USERID parameter can never receive files. • USE If you have specified a directory in a transfer command, then the directory is used. If you have not specified a directory, then the directory where the user exists when starting the previous CyberResp (the \$PWD environment variable for CyberResp) is used. <p> If RECEIVE_OPTION is not specified, USE is the default setting.</p>

Parameter Name	Description
COMMAND_OPTION	<p>Defines the options for using commands.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • ROOT If you have specified a directory in a transfer command, then the directory is appended to the directory defined by the COMMAND_DIR parameter. • NEVER The user defined with the NODE or USERID parameter can never use commands. • USE If you have specified a directory in a transfer command, then the directory is used. If you have not specified a directory, then the directory where the user exists when starting the previous CyberResp (the \$PWD environment variable for CyberResp) is used. <div>  <p>If COMMAND_OPTION is not specified, USE is the default setting.</p> </div>
SUBMIT_OPTION	<p>Defines the options for submitting jobs.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • ALLOW The user defined in a transfer command can submit a job. • NEVER The user defined with the NODE or USERID parameter can never submit a job.

Examples: Access Control

You can define access control parameters to use the access control feature of TIBCO MFT Platform Server. For example, you can define **RECEIVE_OPTION** parameter to determine whether to append both the file name and sender directory name to the receiver directory name.

- **RECEIVE_DIR=/a/b**
RECEIVE_OPTION=FORCE

In this example, the directory name is defined in the **RECEIVE_DIR** parameter and the **RECEIVE_OPTION** parameter is set to **FORCE**. As a result, the defined file name in the **LocalFileName** parameter is appended to the directory defined by the **RECEIVE_DIR** parameter.

If the **LocalFileName** parameter in the transfer command is set as:

```
/test/2008/accounting/tax.data
```

The actual file name is expected to be:

```
/a/b/tax.data
```

- **RECEIVE_DIR=/a/b**
RECEIVE_OPTION=ROOT

In this example, the directory name is defined in the **RECEIVE_DIR** and the **RECEIVE_OPTION** parameter is set to **ROOT**. As a result, both the defined directory name and file name in the **LocalFileName** parameter are appended to the directory defined by the **RECEIVE_DIR** parameter.

If the **LocalFileName** parameter in the request is set as:

```
/test/2008/accounting/tax.data
```

The actual file name is expected to be:

```
/a/b/test/2008/accounting/tax.data
```

- **RECEIVE_DIR=/a/b**
RECEIVE_OPTION=ROOT

In this example, the directory name is defined in the **RECEIVE_DIR** parameter and the **RECEIVE_OPTION** parameter is set to **ROOT**. As a result, both the defined directory name and file name in the **LocalFileName** parameter are appended to the directory defined by the **RECEIVE_DIR** parameter.

If the **LocalFileName** parameter in the request is set as:

```
/test/2008/accounting/tax.data
```

The actual file name is expected to be:

```
/a/b/test/2008/accounting/tax.data
```

Default Access Control Entries

To provide a default entry in case no matches are made, specify default entries for the **USERID** and **NODE** parameters by using the **DEFAULT** value.

See the following example:

```
USERID=DEFAULT,
NODE=NODEA,
SEND_DIR=/mftps/data,
SEND_OPTION=ROOT,
RECEIVE_OPTION=NEVER
*
USERID=DEFAULT
NODE=DEFAULT
SEND_OPTION=NEVER
RECEIVE_OPTION=NEVER
```

Using the **DEFAULT** value, all received files from the **NODEA** node can be placed in the **/mftps/data/RemoteFileName** directory, where *RemoteFileName* is defined in the **RemoteFileName** parameter passed in a transfer command. And the other users cannot send or receive from this server.

Access Control Format

To use access control features, you must follow the formatting rules.

The formatting rules are listed as follows:

- Enter parameters on a single line or on multiple lines.
- Delimit parameters by inserting a comma. If you use a space after the comma, you can continue the parameters on the next line.

For example:

```
USERID=DEFAULT,
NODE=NODEA,
SEND_DIR=/mftps/data,
SEND_OPTION=ROOT,
RECEIVE_OPTION=NEVER
```

is equivalent to:

```
USERID=DEFAULT, NODE=NODEA, SEND_DIR="/mftps/
data", SEND_OPTION=ROOT, RECEIVE_OPTION=NEVER
```

- Enclose special characters in double quotation marks (").
- Define comments by placing an asterisk (*) at the beginning of each line. If you are using UNIX systems, you can define comments by placing // and /* */.

On Windows and UNIX platforms, the sample access control file, which is the `AccessControl.cfg` file, is read each time a transfer is received. Parameter validation is only performed when there is a match for the value defined in the **NODE/USER** parameter and the **TransferType** parameter.

The access control processing is completed at the end of the first match, so if you want to control multiple transfer types for the same pattern, modify the access control entries in the same grouping. Therefore, place more specific conflicting entries in the access control file, so that they are not matched by an earlier, less specific grouping.

CRL Support

Certificate Revocation List (CRL) is used with SSL transfers to ensure the SSL certificates have not been revoked.

CRL support provides an additional way to verify that certificates submitted to TIBCO MFT Platform Server are from a trusted source.

A CRL list is a list of digital certificates, more specifically of serial numbers for certificates that have been revoked. Therefore, the SSL transfers based on revoked certificates are no longer performed. For more information on CRLs, see <http://www.ietf.org/rfc/rfc3280.txt>.

CRL Configuration

You can define a CRL list in the `$CFROOT/config/config.txt` file.

To use CRL, set the **CheckCRL** parameter to Y in the `config.txt` file.

TIBCO MFT Platform Server accesses CRL certificate authority files in a directory, with hashed file names based on OpenSSL naming convention. You can specify the directory with the **CAPath** parameter in the `config.txt` file.

For more information, see [CheckCRL](#) and [CAPath](#) in Server SSL Communications Parameters.

You must rename certificate authority files to correct file names. To get correct hash values, locate and reuse the copy of OpenSSL file in the `$CFROOT/util` directory.

Configuring a Sample CRL

In the following example, a CRL list with certificate authorities is located in a directory with a hashed file name.

Prerequisites

Ensure that you already have a CRL list configured with the file name.

Procedure

1. On the command line, navigate to the `$CFROOT/config` directory.
2. Type the following command to replace the `my.crl` file with the absolute file path.

```
./openssl crl -hash -noout -in my.crl
```

The output screen is expected to be:

```
> ./openssl crl -hash -noout -in my.crl
592b5bc9
```

In this case, the hashed value is generated as 592b5bc9.

3. Type the following commands with the generated hash value in Step 2.

```
cp <your certificate authority file> /usr/CAfiles/592b5bc9.0
cp my.crl /usr/CAfiles/592b5bc9.r0
```

4. Open the `$CFROOT/config/config.txt` file, modify the **CAPath** parameter to the directory where you have placed the hashed files in Step 3.

For this example, the **CAPath** parameter is set to `/usr/CAfiles`.

Configured SSL Authorization

TIBCO MFT Platform Server supports a proprietary extension to standard SSL or tunnel processing so that a system administrator can determine that certificates are either accepted or rejected.

You can configure SSL authorization using the sample authorization configuration file called `SSLAuth.cfg`. The `SSLAuth.cfg` file is by default located in the `$CFROOT/config/` directory.



The authorization configuration file checking is in addition to authorization SSL checking. This checking is performed only if a certificate is accepted by SSL.



The `SSLAuth.cfg` file is compared against certificates received by TIBCO MFT Platform Server. The `SSLAuth.cfg` file is not used on an TIBCO MFT Platform Server client.

The components of a distinguished name (DN) of a certificate are compared to the parameters in the `SSLAuth.cfg` file to determine if a certificate is accepted or rejected.

If no `SSLAuth.cfg` file is defined, or a match is not found in the `SSLAuth.cfg` file, the request is then accepted. All requests contain a variety of parameters. If a parameter is not defined, then it is assumed that the parameter is a match.

The authorization file checking is performed in sequence. For example, if a certificate matches an early entry in the `SSLAuth.cfg` file, the authorization file checking stops matching any later entries.

Because the authorization file checking is processed with a "first-in, first-out" (FIFO) method, if you want to reject all checking requests unless all the certificates are defined by the `SSLAuth.cfg` file, insert the following statements as the last entry in the `SSLAuth.cfg` file:

ACCEPT

Accept an SSL request

REVOKE | REJECT

Do not accept an SSL request

Configured SSL Authorization Format

You have to follow the formatting rules when configuring the `SSLAuth.cfg` file.

On many of the parameters, a generic terminating character is supported.

- TIBCO MFT Platform Server uses asterisk (*) character to represent that a entry matches all values that begin with the text before the * character.



This feature does not support * in the middle of a word as a wildcard character.

- If the entry ends with a comma (,), then the entry is continued on the next entry.
- Parameters can be defined on a single line or continued over multiple lines.
- All parameter data except the **Accept** and **Revoke | Reject** statements are case sensitive.

SSL Authorization Parameters

You can specify the SSL authorization parameters in the `SSLAUTH.cfg` file, which is located in the `$CFROOT/config/` directory.



You must define the parameters in uppercase.

The following table lists SSL authorization parameters to configure the `SSLAUTH.cfg` file:

Parameter	Description
/CN	Defines the common name defined in a certificate. This is usually the name of the person who requests the certificate. Generic entries are supported.
/OU	Defines the organization unit defined in a certificate. This is also known as the department. Generic entries are supported.
/O	Defines the organization defined in a certificate. This is also known as the company. Generic entries are supported.
/L	Defines the locality defined in a certificate. This is also known as the city. Generic entries are supported.
/ST	Defines the state/province defined in a certificate. Generic entries are supported.
/C	Defines the country defined in a certificate. Generic entries are supported.
/SN	Defines the serial number defined in a certificate. Generic entries are not supported.
/SDATE	Defines the start date for a certificate in the format: <i>ccyyymmdd</i> . The start date is compared against the date that the transfer request is received. <ul style="list-style-type: none"> • If the start date is before the current date, then authorization file checking turns to the next parameter. • If the start date is after the current date, then the transfer request is terminated, and an error is sent to the remote system. Generic entries are not supported.

Parameter	Description
/STIME	<p>Defines the start time for a certificate in the format: <i>hhmm</i>.</p> <p>The start time is compared against the time that the transfer request is received. This parameter is used in conjunction with the SDATE parameter.</p> <ul style="list-style-type: none"> • If the start time is before the current date, then authorization file checking turns to the next parameter. • If the start time is after the current date, then the transfer request is terminated, and an error is sent to the remote system. <p>Generic entries are not supported.</p>
/EDATE	<p>Defines the end date for a certificate in the format: <i>ccyymmdd</i>.</p> <p>The end date is compared against the date that the transfer request is received.</p> <ul style="list-style-type: none"> • If the end date is after the current date, then authorization file checking turns to the next parameter. • If the end date is before the current date, then the transfer request is terminated, and an error is sent to the remote system. <p>Generic entries are not supported.</p>
/ETIME	<p>Defines the end time for a certificate in the format: <i>hhmm</i>.</p> <p>The end time is compared against the time that the transfer request is received. This parameter is used in conjunction with the EDATE parameter.</p> <ul style="list-style-type: none"> • If the end time is after the current date, then authorization file checking turns to the next parameter. • If the end time is before the current date, then the transfer request is terminated, and an error is sent to the remote system. <p>Generic entries are not supported.</p>

Examples: SSL Authorization

A system administrator can determine whether to accept or reject certificates by personalizing SSL authorization. For example, you can set the SSL authorization parameters and the **ACCEPT** and **REVOKE** | **REJECT** statements in the SSLAuth.cfg file to use this feature.

1. To accept all certificates defined with the organization (/O) of OrgA, and the organization unit (/OU) of Marketing, and reject all other certificates, set the following in the SSLAuth.cfg file:

```
Accept /OU=Marketing/O=OrgA
revoke
```

2. To reject any certificates with the serial number (/SN) of 987654 or 123456, but accept all other certificates, set the following in the SSLAuth.cfg file:

```
revoke /SN=987654
revoke /SN=123456
Accept
```

3. To accept all certificates defined with the organization (/O) of ACME, and the organization unit (/OU) started with ACCT, but reject all other certificates, set the following in the SSLAuth.cfg file:

```
Accept /OU=ACCT*/O=ACME
revoke
```

4. To accept all certificates matching the specification of the `/CN`, `/L`, `/ST`, `/C`, `/OU` and `/O` parameters, and the validation from December 1, 2008 to November 30, 2009, and to reject all other certificates, set the following in the `SSLAuth.cfg` file:

```
Accept      /CN=Joe*,
           /L=New York,
           /ST=NY,
           /C=US,
           /OU=Dept1,
           /O=ACME,
           /SDATE=20081201,
           /EDATE=200911300
revoke
```

User Exits

Using user exits, you can build additional processes for advanced file transfer capabilities.

You can have direct access to transfer parameters in a C/C++ program, so that command-line arguments are not required.

The **ExitPrgm** parameter in the `$CFROOT/config/config.txt` file is defined to the path to the exit program on the local machine. For more information, see [ExitPrgm](#) in Server Configuration Parameters.

In the `$CFROOT/samples` directory, you can find an example exit program named `exitprg.cpp`, and a compiled `exitprg.exe` file based on the `exitprg.cpp` file.

All exit programs are compiled with the `CfXitData.h` file, which is located in the `$CFROOT/samples` directory. This file contains the data structure including all the information to be passed to your exit program. For more information, see [CfXitData Structure](#).

All user exit programs are called when a transfer attempt is completed.

- **CF_INIT_POST_TRANSFER** is called by an initiator.
- **CF_RESP_POST_TRANSFER** is called by a responder.

Both functions take a pointer to a **CfXitData** structure as an argument, which contains all parameters. The function prototype is:

```
int CF_INIT_POST_TRANSFER(CfXitData* control);
int CF_RESP_POST_TRANSFER(CfXitData* control);
```

To compile your code, navigate to the directory where your source code is placed, and then use the following command:

```
gcc -c output_file source_code
```

The output file is the file that is expected to be pointed to with the **ExitPrgm** parameter.

CfXitData Structure

The **CfXitData** structure is contained in the `CfXitData.h` file, which is by default located in the `$CFROOT/samples` directory.

See the following example of the `CfXitData.h` file:

```
typedef struct __CfXitData {
    int StatusDiagCode;
    BYTE StatusSeverity;
    BYTE Compression;
    char RecordFormat[2];
    short int BlockSize;
    BYTE PermittedActions;
    char StatusMsg[255];
    char LocalUserId[255];
    DWORD Key;
    DWORD AllocationPrimary;
    DWORD AllocationSecondary;
    DWORD RecordLength;
    DWORD EncryptionType;
```


```

char VolSer[7];
int IsVolSer;
char UNIT[9];
int IsUnit;
char AllocationType[2];
char NewFileAvail[2];
char LocalPassword[255];
char IpAddress[255];
char RemoteUserId[255];
char RemotePassword[255];
char RemoteDomain[255];
char LocalFileName[255];
char RemoteFileName[255];
char TraceFileName[255];
char WriteMode[2];
char TransferFunction[2];
char TransactionNumber[11];
char TransferWork[2];
char CheckPointRestart[2];
char CR_LF[2];
int DataType;
int Port;
DWORD ByteCount;
BOOL FirstTime;
int GoingToRetry;
int TryCount;
int TriedCount;
}CfXitData;

```

The following table lists the parameters that define the **CfXitData** structure:

Field Names	Data Type	Field Values
StatusDiagCode	int	Indicates the return code on reply: <ul style="list-style-type: none"> • x00: Success • x01: Failure • x09: Abort
StatusSeverity	BYTE	Indicates the severity of the transfer status: <ul style="list-style-type: none"> • x00: Success • x01: Informational • x02: Warning • x03: Error • x10: No Checkpoint • x20: No Restart • x80: Retry network error • x81: Retry file error
Compression	BYTE	Indicates the type of compressions: <ul style="list-style-type: none"> • x00: No compression • x11: LZ compression • x12: RLE compression

Field Names	Data Type	Field Values
RecordFormat	char[2]	<p>Indicates the remote file record format:</p> <ul style="list-style-type: none"> • F: Fixed blocked • V: Variable blocked • U: Undefined • X: Fixed • Y: Variable
BlockSize	Short int	Indicates the z/OS dataset block size.
PermittedActions	BYTE	<p>Indicates the type of permitted actions:</p> <ul style="list-style-type: none"> • x00: None • x02: EOF • x04: CRLFEOF • x08: System • x10: Hidden • x20: Archive • x40: Read Only • x80: NTFS Compressed
StatusMsg	char[255]	Indicates the text message with transfer status.
LocalUserId	char[255]	Indicates the local user ID that initiates the transfer.
AllocationPrimary	DWORD	Indicates the initial number of units for physical storage to allocate when creating datasets.
AllocationSecondary	DWORD	Indicates the next number of units for physical storage to allocate as soon as the initial allocation in the datasets is used up.
RecordLength	DWORD	<p>Indicates the maximum logical record length, that is, the string length used to encode the data records of the file.</p> <p>The maximum logical record length in a z/OS system is 32,760.</p> <p>For best results, you might omit this parameter if you want to send or receive a file into a file that already exists, because the file can be determined with an appropriate length .</p> <div>  <p>This parameter is ignored when you perform a send transfer to TIBCO MFT Platform Server for Windows, because it is a z/OS-specific parameter.</p> </div>

Field Names	Data Type	Field Values
EncryptionType	DWORD	Indicates the type of encryption: <ul style="list-style-type: none"> • x80: Data is not encrypted (check x00 too) • x40: DES Encryption • x20: 3DE Encryption • x10: Blowfish Encryption • x08: Blowfish Long • x04: Rijndael • x00: No encryption (check x80 too)
VolSer	char[7]	Indicates the remote file volume name.
IsVolSer	int	N/A
UNIT	char[9]	Indicates the remote file unit name.
IsUnit	int	N/A
AllocationType	char[2]	Indicates the type of allocation: <ul style="list-style-type: none"> • T: Tracks • C: Cylinders • K: Kilobytes • M: Megabytes
NewFileAvail	char[2]	Indicates whether the new file is available: <ul style="list-style-type: none"> • I: Immediate • D: Deferred (Tape)
LocalPassword	char[255]	Indicates the password for local system (occasionally NULL).
IpAddress	char[255]	Indicates the IP address of the remote system.
RemoteUserId	char[255]	Indicates the user ID on the remote system.
RemotePassword	char[255]	Indicates the password to log in to the remote system.
RemoteDomain	char[255]	Indicates the Windows NT Domain for login (not required in all transfers).
LocalFileName	char[255]	Indicates the name of the local file for the transfer.
RemoteFileName	char[255]	Indicates the name of the remote file for the transfer.

Field Names	Data Type	Field Values
TraceFileName	char[255]	Indicates the name of the file which tracing is logged to.
WriteMode	char[2]	Indicates the options for file creation: <ul style="list-style-type: none"> • R: Replace • A: Append • C: Create • X: Create / Replace • Y: Create / Append • Z: Create / Replace / New
TransferFunction	char[2]	Indicates the transfer types: <ul style="list-style-type: none"> • S: Send • R: Receive
TransactionNumber	char[11]	Indicates the transaction number for the transfer.
TransferWork	char[2]	Indicates the type of file transfers: <ul style="list-style-type: none"> • F: File to File • P: File to Print • C: Remote Command • J: File to Job
CheckpointRestart	char[2]	Indicates whether to use checkpoint: <ul style="list-style-type: none"> • Y: Checkpoint used • N: Checkpoint not used
CR_LF	char[2]	Indicates whether to use the CR/LF delimitation: <ul style="list-style-type: none"> • Y: Yes • N: No • L: Line Feed only
DataType	int	Indicates the type of data: <ul style="list-style-type: none"> • 0: Binary • 1: ASCII • 2: EBCDIC
Port	int	Indicates the IP port number used for transfer.

Field Names	Data Type	Field Values
ByteCount	DWORD	Indicates the number of bytes transmitted.
GoingToRetry	int	Indicates whether the transfer is successful and to retry: <ul style="list-style-type: none"> • 0: transfer is successful and never retried. • 4: transfer is unsuccessful due to a network error and to retry later. • 8: transfer is unsuccessful due to something other than a network error and to retry later.
TryCount	int	Indicates the value assigned to the TryNumber parameter.
TriedCount	int	Indicates the number of attempts for transfer.

cfunix2dos.exe Utility

You can use the **cfunix2dos.exe** utility to convert a file from UNIX format to DOS format.

Use the **cfunix2dos.exe** utility to add line a feed character (^M) to the end of each line of a UNIX format file. As a result, you can send the file to Windows in binary format.

When using the **./cfunix2dos.exe** utility, the format is:

```
./cfunix2dos.exe filename
```

Example of cfunix2dos.exe Utility

See the following example of the utility in use and the output:

```
./cfunix2dos.exe /usr/tmp/file.txt
cfunix2dos complete for file ==> /usr/tmp/file.txt
Input bytes=3074
Output bytes=3131
```

TIBCO Accelerator

You can use TIBCO Accelerator technology to improve data transfer speed over IP network connections (high bandwidth, high latency).

Completed transfers with TIBCO Accelerator technology have been proved to be up to 10 to 100 times faster than FTP, overcoming slowness due to latency and high error rate. TIBCO Accelerator technology is added to TIBCO MFT Platform Server to provide a faster way to send files to remote destinations, where high latency is a problem with long distance connections.

TIBCO Accelerator technology uses its own version of User Datagram Protocol (UDP), and it offers a parallel implementation of Transmission Control Protocol (TCP), called Parallel Delivery Protocol (PDP).

TIBCO Accelerator Ports

TIBCO Accelerator listens on different ports based on coming requests.

By default, the TIBCO Accelerator listens on the following ports:

- Port 9000: for requests using TCP or UDP protocol.
- Port 9002: for requests using PDP protocol.

- Port 9099: for general requests through TIBCO Accelerator.

When the request has been received, the TIBCO Accelerator client sets a port number for the transfer between the TIBCO Accelerator server and the responder, the port number is in the range 9100 - 9199.



To perform transfers from a TIBCO Accelerator client to a TIBCO Accelerator server, you must ensure that firewall ports 9000, 9002 and 9100 - 9199 are opened. If requests are initiated from an external computer, these firewall ports must be opened for incoming traffic. If requests are initiated from an internal computer, these firewall ports must be opened for outgoing traffic. Port 9099 is typically used locally and therefore does not require any firewall charges.

Using TIBCO Accelerator within TIBCO MFT Platform Server for UNIX

You can speed up file transfers by passing files through a platform server where TIBCO Accelerator daemon is running.

TIBCO Accelerator technology is available in TIBCO MFT Platform Server for Linux and TIBCO MFT Platform Server for Windows. You can send and receive files from z/OS, UNIX and Windows platforms, but only when the files are transferred through the TIBCO MFT Platform Server running the TIBCO Accelerator daemon (RSTunnel.exe).



The TIBCO Accelerator within TIBCO MFT Platform Server can act as a client, a server, or both. For more information about the implementation of TIBCO Accelerator within TIBCO MFT Platform Server for Windows, see *TIBCO Managed File Transfer Platform Server for Windows User's Guide*.

On the platform server, you can perform the following tasks to use TIBCO Accelerator:

- [Managing TIBCO Accelerator](#)
- [Configuring the TIBCO Accelerator Server](#)
- [Configuring the TIBCO Accelerator Client](#)

Define the following parameters in the `config.txt` file to set up the TIBCO Accelerator features:

- **Accelerate** (ACC)
- **ACCCompression** (ACCC)
- **ACCEncryption** (ACCE)
- **ACCHost** (ACCH)
- **ACCMAXSpeed** (ACCMAX)
- **ACCPort**
- **ACCProtocol** (ACCP)

For more information, see [TIBCO Accelerator Transfer Parameters](#).

Managing TIBCO Accelerator

You can use the command line to start and stop TIBCO Accelerator and verify the status of a running daemon.

Prerequisites

Ensure that you have set the **\$CFROOT** variable before using commands to manage TIBCO Accelerator.

Procedure

1. On the command line, navigate to the `$CFROOT/Accelerator` directory.
2. Use the following commands to manage TIBCO Accelerator:

- Enter `./acctstart` to start TIBCO Accelerator.
- Enter `./accstop` to stop TIBCO Accelerator.
- Enter `./accstatus` to check the status of the ACC Tunnel processes.

Configuring the TIBCO Accelerator Server

You can modify the `RocketStreamConfig.xml` file to configure the TIBCO Accelerator server.



Generally you do not have to make configuration changes to the base settings, unless you want to bind to a specific IP address or change the default ports.

Procedure

1. On the command line, navigate to the `$CFROOT/Accelerator` directory.
2. Open the `RocketStreamConfig.xml` file using any text editor.
3. Navigate to the `<inbound>` element and modify the values as you want. The following are the default sets of IP addresses and ports:

```
<inbound enabled="true">
  <!-- Rocket Stream listeners that remote agents connect to -->
  <listener type="tcp" enabled="true">
    <address>0.0.0.0</address>
    <port>9000</port>
  </listener>
  <listener type="pdp" enabled="true">
    <address>0.0.0.0</address>
    <port>9002</port>
  </listener>
  <listener type="udp" enabled="true">
    <address>0.0.0.0</address>
    <port>9000</port>
  </listener>
</inbound>
```

4. Stop and start the TIBCO Accelerator daemon to make the changed `RocketStreamConfig.xml` file effective. For more information, see [Managing TIBCO Accelerator](#).

Configuring TIBCO Accelerator Client

You can modify the `RocketStreamConfig.xml` file to configure the TIBCO Accelerator client.

You have to change the routing table to instruct the TIBCO Accelerator client which TIBCO Accelerator server it has to send the files to.

Procedure

1. On the command line, navigate to the `$CFROOT/Accelerator` directory.
2. Open the `RocketStreamConfig.xml` file using any text editor.
3. Navigate to `<routing-table>` element and modify the values. You can set up multiple `<route>` elements for multiple servers. In each `<route>` elements, there are two `<destination>` elements:
 - The first `<destination>` element defines the destination host, which is the IP address of the platform server responder.
 - The second `<destination>` element defines the gateway host, which is the IP address of the TIBCO Accelerator server. The gateway host receives the transferred files and pass them to the destination host.



You can also modify the default port numbers used for various protocols. The port numbers must match both the source and destination computers.

The following is an example of the routing table:

```
<routing-table>
  <route>
    <!-- final destination host name or IP address -->
    <destination>10.10.10.10</destination>
    <!-- remote accelerator host name or IP address and ports for each
supported protocol -->
    <remote-accelerator>
      <destination>192.168.1.10</destination>
      <ports>
        <pdp>9002</pdp>
        <udp>9000</udp>
        <tcp>9000</tcp>
      </ports>
    </remote-accelerator>
  </route>
</routing-table>
```

- Optional: Navigate to the `<listener>` element at the top of the `RocketStreamConfig.xml` file if you have to change the default port 9099 and IP address that the TIBCO Accelerator client binds to.

The following is the default set of port number and IP address:

```
<listener type="tcp" enabled="true">
  <address>0.0.0.0</address>
  <port>9099</port>
</listener>
```

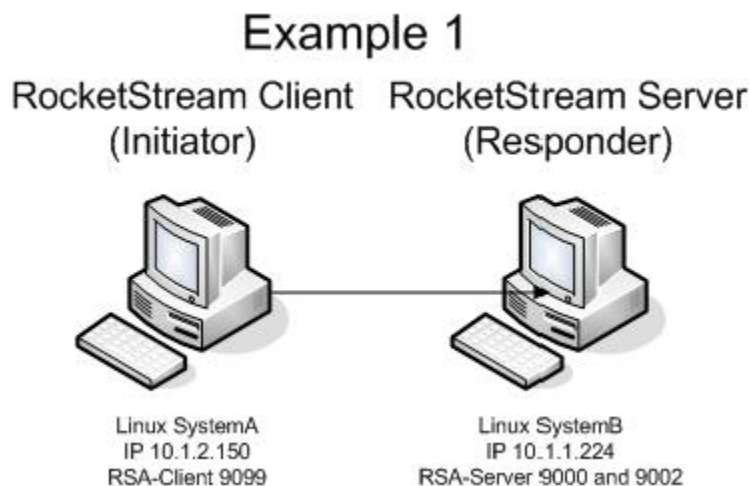
- Stop and start the TIBCO Accelerator daemon to make the changed `RocketStreamConfig.xml` file effective.

For more information, see [Managing TIBCO Accelerator](#).

Example 1: Transfer from Linux to Linux through TIBCO Accelerator

This example describes a file sent between two platform servers installed on Linux system through TIBCO Accelerator.

The following figure illustrates the example:



The transfer takes two important steps:

- First, you have to verify if the platform server responder (SystemB) has a TIBCO Accelerator daemon running and is listening on the port 9000 and 9002. To verify this, run the `./accstatus` command. For more information, see [Managing TIBCO Accelerator](#).

- Next, you can initiate a file transfer on SystemA to send a file to SystemB. To have the file be sent through TIBCO Accelerator, use the transfer command is as follows:

```
cfsend ip:10.1.1.224 port:46464 lf:/home/usr/file
rf:/opt/temp/${LocalFileName} uid:remote_user pwd:remote_password
ACC=Y ACCH=10.1.2.150 ACCPort:9099 ACCP:PDP
```

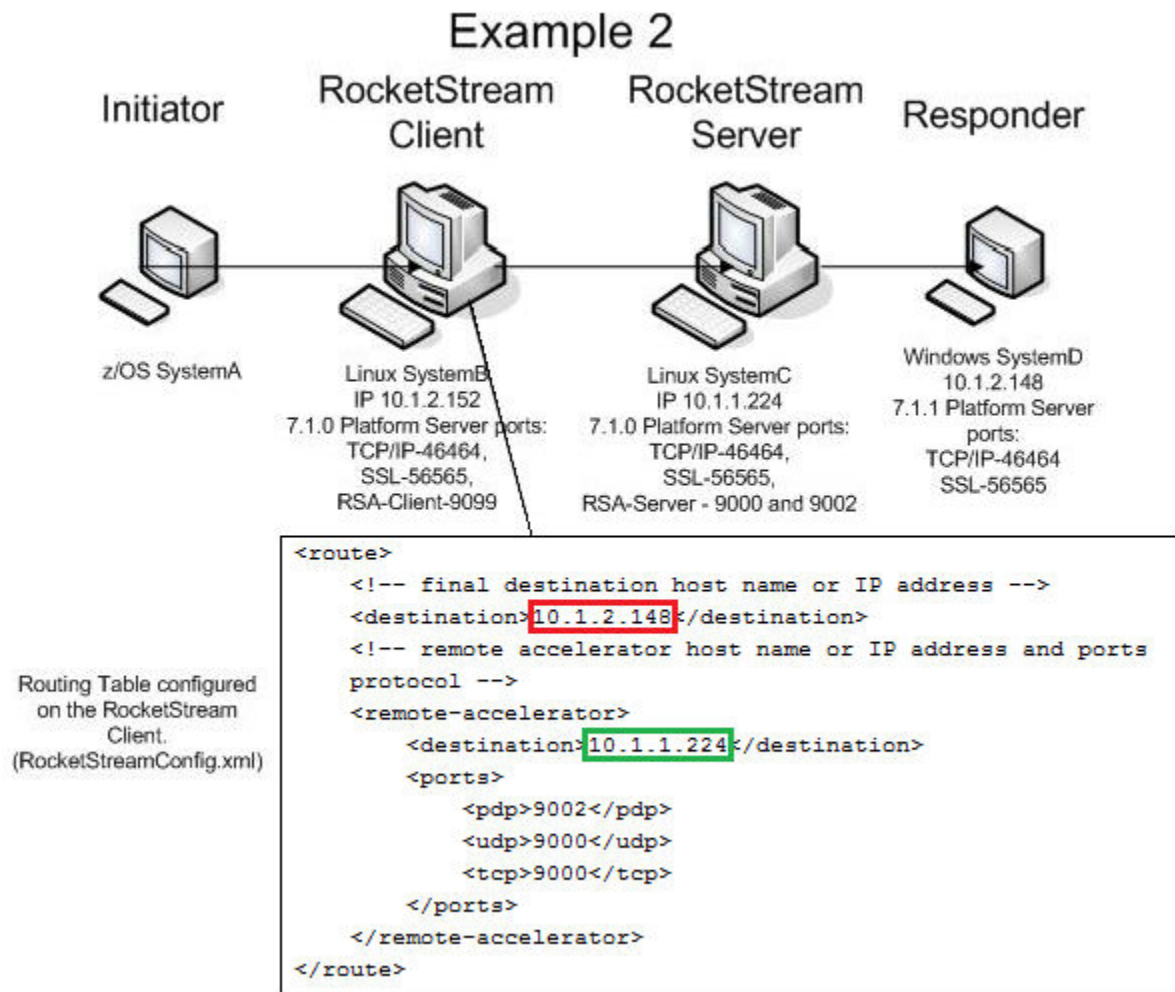
In this example, the last four parameters are added to send a file through TIBCO Accelerator:

- The **ACC** parameter is set to Y to send this file using TIBCO Accelerator.
- The **ACCH** parameter is set to the address to the TIBCO Accelerator client that receives the transfer request.
- The **ACCPort** parameter is set to the port number which the TIBCO Accelerator client listens on.
- The **ACCP** parameter is set to the transfer protocol which is used in the transfer.

Example 2: Transfer from z/OS to Windows through TIBCO Accelerator on Linux

This example describes a file sent from a platform server on z/OS to a platform server on Windows, through TIBCO Accelerator installed on two platform servers on Linux.

The following figure illustrates the example:



The transfer takes two important steps:

- First, you have to configure the routing table on the TIBCO Accelerator client (SystemB) to enter the addresses of the TIBCO Accelerator server (SystemC) and the final destination, which is the platform server responder (SystemD). For more information, see [Configuring the TIBCO Accelerator Client](#).



If the final destination is the TIBCO Accelerator server itself, no routing table entry is required, as shown in [Example 1](#). It is only required when the platform server responder is a different machine than the TIBCO Accelerator server.

- Next, you have to configure the platform server initiator (SystemA), to enter the information of the TIBCO Accelerator client (SystemB). This configuration is done by referencing the routing table that has been configured earlier. For more information on how to initiate file transfers on a platform server from a z/OS machine, see *TIBCO Managed File Transfer Platform Server for z/OS documentation*.

A TIBCO Accelerator server can send a file to any TIBCO MFT Platform Server responder with version 7.0 or lower. This includes TIBCO MFT Platform Server on Windows, UNIX, z/OS, and IBM i machines.

If you want further assistance for this example, contact TIBCO Support.

Appendix: PAM Authentication

To enable Pluggable Authentication Module (PAM) authentication, you must configure PAM on your system.

TIBCO MFT Platform Server for UNIX supports two methods of authentication:

- Authentication using the password or shadow password file
This is the default authentication mechanism. Authentication is performed against the password or shadow password files. To enable password authentication, set the **PamAuth** parameter to **N** in the `config.txt` file. By default, this parameter is set to **N**, and no other configuration is required.
- Authentication using PAM
Authentication is performed using PAM. The authentication method can be any authentication methods supported by PAM, including the password or shadow password files and LDAP. To enable PAM authentication, set the **PamAuth** parameter to the name of the PAM service in the `config.txt` file.

Configuring PAM Authentication

You can configure PAM by creating a configuration file or configuration entry for the PAM service.



It is critical that you work with your system administrator to configure the PAM service settings. Each system might configure PAM slightly differently. The system administrator has to review the PAM requirements for your system before making any PAM changes.

The following PAM module types are used.

PAM Module Type	Description
auth	Used to authenticate the user id and password credentials.
account	Used to validate that the authenticated user is able to access the system.

Configuring PAM Authentication Service

When using PAM, you must configure the authentication and account services that you want to use. You can configure PAM in one of the following ways:

- Generally for Linux systems: create a file in the `/etc/pam.d` directory with a file name that matches the service name set by the **PamAuth** parameter in the `config.txt` file.
- Generally for UNIX systems(including AIX): create entries in the `/etc/pam.conf` file with a service name that matches the service name set by the **PamAuth** parameter in the `config.txt` file.

See the following examples for how to configure LDAP authentication. The examples assume that the **PamConfig** parameter is set to `mftserver`.

For Linux: create a file called `/etc/pam.d/mftserver`

Sample parameters to configure LDAP support:

```
auth    sufficient    pam_ldap.so use_first_pass
account sufficient    pam_ldap.so
```

For UNIX: create entries in the `/etc/pam.conf` file

Sample parameters to configure LDAP support:

mftserver	auth	sufficient	pam_ldap.so use_first_pass
mftserver	account	sufficient	pam_ldap.so

TIBCO Documentation and Support Services

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the [TIBCO Product Documentation](#) website, mainly in HTML and PDF formats.

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

- *TIBCO® Managed File Transfer Platform Server for UNIX Installation*
- *TIBCO® Managed File Transfer Platform Server for UNIX User's Guide*
- *TIBCO® Managed File Transfer Platform Server for UNIX Docker Container Deployment*
- *TIBCO® Managed File Transfer Platform Server for UNIX 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 TIBCO Software Inc.

TIBCO, the TIBCO logo, the TIBCO O logo, and Slingshot are either registered trademarks or trademarks of TIBCO Software 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.

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.txt 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. TIBCO SOFTWARE 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 TIBCO Software Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 2003-2022. TIBCO Software Inc. All Rights Reserved.