



TIBCO[®] Managed File Transfer Platform Server for UNIX

User's Guide

Version 8.1.0

August 2021

Document Updated: August 2022



Contents

Contents	2
Product Overview	7
TIBCO® Managed File Transfer Suite	7
TIBCO Managed File Transfer (MFT) Platform Server	8
TIBCO MFT Platform Server Modes of Operation	8
TIBCO MFT Platform Server Groups	9
Root User	10
Cfadmin Member	10
Cfbrowse Member	14
Cftransfer Member	14
TLS/SSL Certificates Setup	16
Generating a Certificate Request	16
Parameters in the SSL Utility (sslutility.exe)	18
Viewing a Certificate	20
Encrypting a Private Key Password	21
Configuring a Certificate	22
Transfer Commands	25
File to File Transfers	26
Examples: Transfer Using cfsend or cfrecv Command	26
Wildcards	27
Directory Transfers	27
File Name Tokens	31
PPA: Preprocessing and Postprocessing Actions	39
File to Job Transfers	49
Examples: File to Job Transfers	49

Running Remote Commands	49
Remote Command Types	50
Examples: Running Remote Commands	50
Error Handling	50
File to Print Transfers	52
Transfer Using Nodes	53
Creating Nodes	54
Deleting Nodes	57
Listing Nodes	58
Node Parameters	58
Required Node Parameters to Create a Node	58
Optional Node Parameters to Create a Node	59
Examples: Transfer Using Nodes	65
Transfer Using Distribution Lists	67
Configuring Distribution Lists	67
Distribution List Parameters	68
Examples: Transfer Using Distribution Lists	68
Transfer Using Profiles	70
User Profile	70
Creating User Profiles	70
Listing User Profiles	71
Deleting User Profiles	72
User Profile Parameters	72
Responder Profile	74
Creating Responder Profiles	75
Listing Responder Profiles	76
Deleting Responder Profiles	76
Responder Profile Parameters	77
Transfer Using Templates	79

Sample Templates: TSEND and TRECVC	80
Transfer Parameters	84
Minimum Transfer Parameters	84
Optional Transfer Parameters	85
z/OS Specific Transfer Parameters	99
TIBCO Accelerator Parameters	106
Configuration Parameters	109
Server Configuration	110
Server Configuration Parameters	111
Server SSL Communications Parameters	116
Client Configuration	121
Client Configuration Parameters	121
Client SSL Communications Parameters	126
Common Configuration	128
Common Configuration Parameters	129
Extended Features	148
Checkpoint Restart	149
Checkpoint Restart Example	150
Conversion Tables/Custom Code Conversion	150
ASCII to EBCDIC Conversion Table Example	154
Directory Named Initiation (DNI)	155
fusing Utility	155
fusutil Utility	156
Configured Postprocessing	158
Argument Substitution	160
Configured Postprocessing Command Examples	161
CfAlias	161
CfAlias Parameters	162
Substitutable Parameters	163

Examples: Using CfAlias	164
Auditing (cfinq Utility)	165
cfinq Parameters	165
Changing CyberMgr Settings	175
Log Files	176
Access Control	180
Access Control Parameters	180
Examples: Access Control	184
Default Access Control Entries	185
Access Control Format	186
CRL Support	187
CRL Configuration	187
Configured SSL Authorization	188
Configured SSL Authorization Format	189
SSL Authorization Parameters	190
Examples: SSL Authorization	192
User Exits	193
CfXitData Structure	194
cfunix2dos.exe and cfdos2unix.exe Utilities	200
cfunix2dos.exe Utility	200
cfdos2unix.exe Utility	201
Appendix A: PAM Authentication	202
Configuring PAM Authentication	202
Configuring PAM Authentication Service	203
Appendix B: Activating CyberMgrBackup in Non-HA Environment	204
Uninstalling CyberMgrBackup	205
Appendix C: Active/Active High Availability Support	206
HA Parameters (config.txt)	208
HA Requirements	210

Installing Platform Server in HA Mode	210
Starting and stopping CyberMgr	211
Options for Configuring Primary and Secondary CyberMgr	211
TIBCO Documentation and Support Services	213
Legal and Third-Party Notices	215

Product Overview

TIBCO® Managed File Transfer Platform Server for UNIX is an integration product that provides 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.

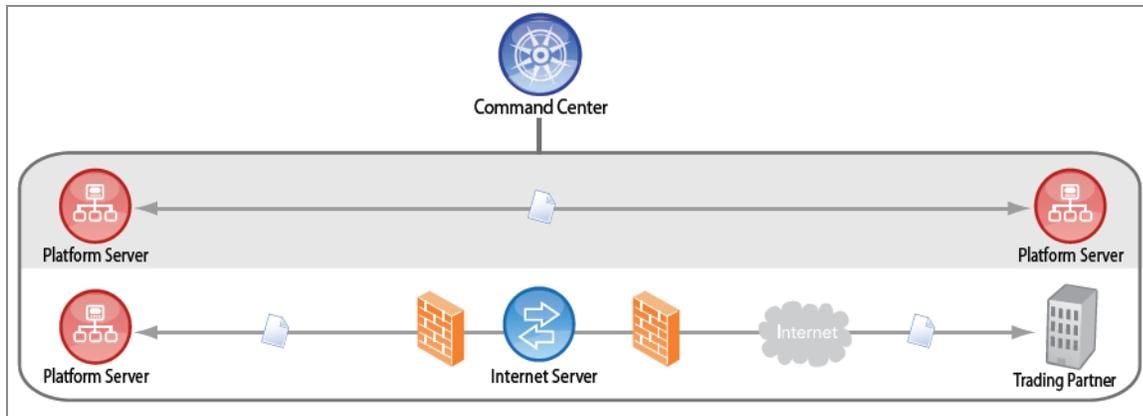
TIBCO® Managed File Transfer Suite

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, Solaris Sparc, Solaris Intel), Linux, z/Linux, iSeries, and IBM Mainframe.
- It offers a full range of encryption algorithms, 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.

TIBCO MFT Platform Server Modes of Operation

TIBCO 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>

TIBCO MFT Platform Server Groups

As a TIBCO MFT Platform Server user, you can be a member of any of the following groups:

- root
- cfadmin
- cfbrowse
- cftransfer

The order of group checking is always root, cfadmin, and then the other groups. Group names can be customized.

Root User

The difference between a root and a non-root user is shown by the following example.

Installed under	Remote incoming	Resolved local	Request runs under	Any file access done under	Groups membership checked for	Local uid in Log.txt
root	test	mary	mary	mary	mary	mary
tom	test	mary	tom	tom	mary	mary

Cfadmin Member

The following programs and CC requests check for membership in the cfadmin group:

Program/CC Request	Description
cfnode	Command-line program to manage nodes.
cfprofile	Command-line program to manage profiles.
cfping	Command-line program to "ping" target Platform Servers.
cfinq	Command-line program to inquire on completed transfers, active transfers and to update the CyberMgr configuration.
cfrprofile	Platform Server command to list, add, or delete responder profiles.
cc_node	Command Center-initiated request to manage nodes.
cc_profile	Command Center-initiated request to manage profiles.
cc_ping	Command Center-initiated ping requests.

Program/CC Request	Description
cc_get_active	Command Center-initiated request to inquire on active transfers.
cc_rprofile	Command Center-initiated request to list, add, or delete responder profiles.
cc_collector	Command Center request to collect transfers history.

The `cfadmin` group must exist. After installation if this group is removed, only file transfers work. Features other than file transfers, such as management, fail if the group is removed.

If you are a member of the `cfadmin` group, you have all Platform Server rights/access just like a member of the root group.

Special consideration should be given to the `cfadmin` group for High Availability (HA) setup.

- All machines participating in HA should resolve group names the same way.
- `cfadmin`, `cfbrowse`, and `cftransfer`, if they exist, should be identical on all machines in the same HA cluster. In other words, if you run an `ls` command on `HaDir`, the `cfadmin` group name must be the same, whether you run the `ls` command on machine A or machine B.

Any request made by executable files or Command Center which modifies the existing setup require a strict `cfadmin` only right. These requests are:
`node`, `profile`, `rprofile`, `cfinq mgr=u`

Apps/CC Requests	Octal	Ownership	Permissions
<code>cfnode.cfg</code>	664	Owner, <code>cfadmin</code> , others	Owner/ <code>cfadmin</code> - read and write Others - Only read
<code>cfnode.exe</code>	510	Owner, <code>cfadmin</code> , others	Owner/ <code>cfadmin</code> - execute
<code>CC_manage_nodes</code>			User must be a member of <code>cfadmin</code> to run any CC node request.

Apps/CC Requests	Octal	Ownership	Permissions
cfprofile.cfg	664	Owner, cfadmin, others	Owner/cfadmin - read and write Others - Only read
cfprofile.exe	510	Owner, cfadmin, others	Owner/cfadmin - execute
CC_manage_ rprofiles			User must be a member of cfadmin to run any CC rprofile request.
cfprofile.cfg	664	Owner, cfadmin, others	Owner/cfadmin - read and write Others - Only read
cfprofile.exe	510	Owner, cfadmin, others	Only owner/cfadmin - execute
CC_manage_profiles			User must be a member of cfadmin to run any CC profile request.

If you want to allow non-cfadmin users to create/update/delete their own profile or rprofile, change permissions to the following files:

Files	Octal	Ownership	Permissions
cfprofile.cfg	666	Owner, cfadmin, others	Anybody can read and write.
cfprofile.exe	511	Owner, cfadmin, others	Anybody can execute, other programmatically restricted to only see/create/update/delete their own rprofile.
cfprofile.cfg	666	Owner, cfadmin, others	Anybody can read and write.

Files	Octal	Ownership	Permissions
cfprofile.exe	511	Owner, cfadmin, others	Anybody can execute, other programmatically restricted to only see/create/update/delete their own rprofile.

Also, for the following files, the ownership, and permissions are shown below:

Files	Octal	Ownership	Permissions
Log.txt	664	Owner, cfadmin, others	File is owned by CyberMgr account.
visibility.txt	664	Owner, cfadmin, others	Shared by CyberResp account and CyberMgr account; or cfinq account and CyberMgr account. This is a temporary file.
cfinq.exe	511	Owner, cfadmin, others	Anybody can execute.

- To run `cfinq mgr=u` and update active CyberMgr daemon setting, you must be in the `cfadmin` group.
- To run `cfinq mgr=a` and see active transfers, you must be a member of `cfadmin` group.
- To run `cfinq` and see the history, you don't need to be in the `cfadmin` group. Depending on the `cfbrowse` group and `StrictGroupChecking`, you can see all history or only your own history.
- To run `fusping` and ping another machine, you don't need to be in the `cfadmin` group.

Cfbrowse Member

Some requests check for membership in the `cfbrowse` group if you are not in the `root` or `cfadmin` group.

The following files and CC requests check for membership in the `cfbrowse` group:
`cc collector log history`, `cfinq log history`

Group does not exist:

- If `StrictGroupChecking=N`, then you can see all log records.
- If `StrictGroupChecking=Y`, then your request is rejected.

Group exists, user is a member:

- If `StrictGroupChecking=N|Y`, then you can see all log records.

Group exists, user is not a member:

- If `StrictGroupChecking=N|Y`, then you can see only your own log records.

i Note: `cfinq mgr=active` only allows you to see active transfers, there is no option to cancel active transfers. Cancellation can be done only from the Command Center side.
 In HA setups, the `haDir/visibility` folder is used by `cfinq mgr=active` requests to create a temporary file, where `CyberMgr` writes active transfer information. This folder has permission `775`, restricting ordinary users from having write access to it. Therefore, `cfinq mgr=active` can be successful only when it is run by a `cfroot` or a `cfadmin` member.

Cftransfer Member

Only `cc_xfer` requests check for membership in the `cftransfer` group.

Group does not exist:

- If `StrictGroupChecking=N`, then you are allowed to run `cc_xfer` transfers.
- If `StrictGroupChecking=Y`, then you are not allowed to run `cc_xfer` transfers.

Group exists, user is a member:

- If `StrictGroupChecking=N|Y`, then you are allowed to run `cc_xfer` transfers.

Group exists, user is not a member:

- If `StrictGroupChecking=N|Y`, then you are not allowed to run `cc_xfer` transfers.

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 SSL Utility (`sslutility.exe`) to generate certificate request and private key files.

Before you begin

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

 **Note:** 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` file generates a certificate request file and a private key file. Then,

the certificate request file is sent to a certificate authority. The certificate authority returns a certificate. This certificate should be saved in a file.

The following example shows how to use the `sslutility.exe`. For more information regarding the parameters of the utility, see [Parameters in SSL 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
2

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

2

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 SSL Utility (sslutility.exe)

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

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

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.

Parameter	Description
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. Note: The value of a country code must be 2 characters.
Email address	Defines the email address that is associated with the certificate holder. Note: 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. Note: Ensure that there are no spaces in the file name.
Private Key File Name	Defines the full name of the private key file. Note: Ensure that there are no spaces in the file name.
Private Key Password	Defines the password used to access a private key.

Parameter	Description
	<p>Note: The maximum length of a private key password is 20 characters.</p>
Directory to Place the Generated Files	<p>Defines the directory where the certificate request file and private key file are.</p> <p>Press Enter to use the default /tmp directory.</p>

Viewing a Certificate

You can use the SSL Utility (`sslutility.exe`) 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 (0x3)
    Serial Number: 90 (0x5a)
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=US, ST=NY, L=GC, O=OrgA, OU=DevIA390, CN=a390.orga.com
    Validity
      Not Before: Jul  1 04:00:00 2019 GMT
      Not After : Jan  1 03:59:59 2021 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.

Before you begin

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` file, 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.

Before you begin

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:     Y                { 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:
SSLEnabledProtocols:   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:
SSLEnabledProtocols:   TLSV1,TLSV1.1,TLSV1.2 {TLSV1,TLSV1.1,TLSV1.2
}

```

```
}  
Ciphers:                HIGH                { openssl_cipher_list  
}
```

5. After SSL parameters are configured, use `cfstart -ssl` and/or `cfstart -tunnel` to start `CyberResp` daemon in SSL or Tunnel mode and enable transfers in SSL or Tunnel mode.

Transfer Commands

You can use the `cfsend` and `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. For more information, see [Transfer Using Templates](#).

 **Note:** In a `cfsend` or `cfrecv` command, if you use a node or a template and 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 ignore log off before the command is completed, add `nohup` before a command.
- To send the screen output to a specified file, add `> logFilePath >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` or `IPName/Address`
- `LocalFileName`
- `RemoteFileName`
- `RemoteUserId` (If you define a profile, this parameter is not needed.)
- `RemotePassword` (If you define a profile, this parameter is not needed.)

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

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.

i Note: Use the equal sign (=) or the colon (:) 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 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"
```

- 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 > /tmp/file 2>&1
```

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 the Platform Server commands 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 alphanumeric 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`.

i Note: You must put wildcards after the last forward slash (/) to interpret them. Wildcards before the last forward slash (/) are ignored and causes errors.

Directory Transfers

By specifying certain transfer parameters and options, you can 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
ScanSubDir (ssd)	<p>Defines whether all subdirectories from the file path are scanned.</p> <p>Note: This parameter only applies to directory transfers.</p> <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> <p>Note: This parameter applies to directory transfers and distribution list transfers.</p> <p>The valid values are Y or N.</p>
Test	<p>Defines whether to display the local and remote file names that are transferred without doing the actual transfers, so that you can verify whether the file names are correct before the actual transfers.</p> <p>Note: This parameter applies to directory transfers and distribution list transfers.</p> <p>The valid values are Y or N.</p>
ExecPostProc	<p>Defines whether to have the postprocessing action execute after each file in a directory or after the last file in the directory or distribution list is transferred.</p>

Parameter (Alternate Specification)	Description
	The valid values are <code>Default</code> , <code>Parent</code> , and <code>Child</code> .
<code>ExecPreProc</code>	Defines whether to have the preprocessing action execute before each file in a directory or before the entire directory or distribution list is transferred. The valid values are <code>Default</code> , <code>Parent</code> , and <code>Child</code> .

Directory Tokens

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

i **Note:** Tokens are case-sensitive.

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

Token	Description
<code>\$(SDIR)</code>	<p>This parameter substitutes the subdirectory of the file transferred. You can use this case-sensitive token with a receive transfer as part of the <code>LocalFileName</code> path and with a send transfer as part of the <code>RemoteFileName</code> path.</p> <ul style="list-style-type: none"> In the <code>LocalFileName</code> parameter when using a <code>cfrecv</code> command. In the <code>RemoteFileName</code> parameter when using a <code>cfsend</code> command. <p>For example,</p> <pre>cfrecv ScanSubDir:Y LocalFileName: '/home/johndoe/data/\$(SDIR) /</pre>

Token	Description
	<pre data-bbox="505 289 979 384">\$ (RemoteFileName) ' RemoteFileName: '/mftps/data/*' n:MontrealNode</pre> <p data-bbox="505 417 1187 527">As a result of this example, all files from /mftps/data directory and subdirectories are received with the same structure.</p>
\$ (MEMBER)	<p data-bbox="505 636 1203 747">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 data-bbox="505 772 1203 884" style="background-color: #f0f0f0; padding: 5px;"> <p data-bbox="524 793 1117 863">Note: Use this token only when you perform a receive transfer from a remote z/OS system.</p> </div> <p data-bbox="505 909 1162 978">If there is no \$(MEMBER) token in the file name from the z/OS side, this token is ignored.</p> <p data-bbox="505 1010 1203 1161">For example, if you use /mftps/\$(MEMBER)/filename as a file path, then the file path can be resolved to /mftps/DATA/filename WHERE \$(MEMBER)= DATA from PROD.FILE.PDS(DATA).</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 from remote '/mftps/data' directory are received into local '/home/johndoe/data' directory.

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 from '/home/johndoe/data' folder and sub-folders 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 from '/mftps/data' directory and subdirectories 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.

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.

i Note: When you use the dollar sign (\$) or the backward 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 name	Description	Generated Value
<code>\$(LocalFile)</code>	Local file name tokens are used by the <code>cfsend</code> command in the <code>remotefilename</code> parameter.	For example: <ul style="list-style-type: none"> Local file: <code>/home/usr/temp/files/testfile1.txt</code> Remote file: <code>c:\target\\$(LocalFile)</code> Resolution: <code>testfile1.txt</code>
<code>\$(LocalFileBase)</code>	Local file name only.	For example: <ul style="list-style-type: none"> Local file: <code>/home/usr/temp/files/testfile1.txt</code> Remote file: <code>c:\target\\$(LocalFileBase)</code> Resolution: <code>testfile1.txt</code> .
<code>\$(LocalFileExt)</code>	Extension of the local file.	For example: <ul style="list-style-type: none"> Local file: <code>/home/usr/temp/files/testfile1.txt</code> Remote file: <code>c:\target\\$(LocalFileExt)</code>

Token name	Description	Generated Value
		Resolution: testfile1.txt.
<code>\$(LocalFileName)</code>	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) Resolution: testfile1.txt. <div style="border: 1px solid #ccc; background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: You must use the full path of the remote file, otherwise, the file might be transferred to the <code>\$CFROOT</code> directory.</p> </div>
<code>\$(LocalFL##)</code>	File qualifier computed from left-justified position specified by ##.	<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) Resolution: testabc.
<code>\$(LocalFR##)</code>	File qualifier computed from right-justified position specified by ##.	<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) Resolution: testabc.aaa.bb.c.txt.
<code>\$(NoLocalFileBase)</code>	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

Token name	Description	Generated Value
		<ul style="list-style-type: none"> Remote file: c:\target\\$(NoLocalFileBase) Resolution: a.b.c.txt.
\$(NoLocalFileExt)	Local file name excluding the extension.	For example: <ul style="list-style-type: none"> Local file: /home/usr/temp/files/a.b.c.txt <ul style="list-style-type: none"> Remote file: c:\target\\$(NoLocalFileExt) Resolution: a.b.c.txt.
\$(LocalUserId)	Local user ID being used for the file transfer.	For example: <ul style="list-style-type: none"> Local user ID: TESTLAB\cfuser1 Local file: /home/usr/temp/files/testfile1.txt <ul style="list-style-type: none"> Remote file: c:\target\file1\$(LocalUserId).txt Resolution: d:\target\file1cfuser1.txt.
\$(RemoteFile)	Remote file name tokens used by the cfrecv command in the localfilename parameter.	For example: <ul style="list-style-type: none"> Local file: /home/usr/temp/\$(RemoteFile) <ul style="list-style-type: none"> Remote file: c:\source\directory\testfile1.txt Resolution: testfile1.
\$(RemoteFileBase)	The base name of the remote file.	For example: <ul style="list-style-type: none"> Local file: /home/usr/temp/\$(RemoteFileBase)

Token name	Description	Generated Value
		<ul style="list-style-type: none"> Remote file: <p>c:\source\directory\testfile1.txt Resolution: testfile1.</p>
\$(RemoteFileExt)	Extension of the remote file.	<p>For example:</p> <ul style="list-style-type: none"> Local file: <p>/home/usr/files/\$(RemoteFileExt)</p> <ul style="list-style-type: none"> Remote file: <p>c:\source\directory\testfile1.txt Resolution: testfile1.txt</p>
\$(RemoteFileName)	Remote file name including the extension.	<p>For example:</p> <ul style="list-style-type: none"> Local file: <p>/home/usr/files/\$(RemoteFileName)</p> <ul style="list-style-type: none"> Remote file: <p>c:\source\directory\testfile1.txt Resolution: testfile1.txt</p> <div style="border: 1px solid #ccc; background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: You must use the full path of the remote file, otherwise, the file might be transferred to the \$CFROOT directory.</p> </div>
\$(RemoteFL##)	The file qualifier computed from left-justified position specified by ##.	<p>For example:</p> <ul style="list-style-type: none"> Local file: <p>/home/usr/files/\$(RemoteFL01)</p> <ul style="list-style-type: none"> Remote file: <p>c:\source\directory\testabc.aaa.bb.c.txt Resolution: testabc.</p>
\$(RemoteFR##)	The file qualifier	For example:

Token name	Description	Generated Value
	computed from right-justified position specified by ##.	<ul style="list-style-type: none"> Local file: /home/usr/files/\${RemoteFR01} Remote file: c:\source\directory\testabc.aaa.bb.c.txt Resolution: testabc.aaa.bb.c.txt.
\$(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 Resolution: /home/usr/files/file1.cfuser1.txt
\$(Remote TransactionNumber)	Remote transaction number.	<p>For example:</p> <p>Local file: /home/usr/temp/files/testfile1.txt</p> <p>Remote file: c:\target\\$(RemoteTransactionNumber).testfile1.txt Resolution like: RA18100052.testfile1</p>
\$(NoRemote FileBase)	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 Resolution: a.b.c.txt

Token name	Description	Generated Value
<code>\$(NoRemoteFileExt)</code>	Remote file name excluding the extension.	For example: <ul style="list-style-type: none"> Local file: /home/usr/temp/files<code>\$(NoRemoteFileBase)</code> Remote file: c:\source\directory\a.b.c.txt Resolution: a.b.c.txt
<code>\$(Date)</code>	Local date.	YYYYMMDD
<code>\$(Date1)</code>	Local date.	YYMMDD
<code>\$(Date2)</code>	Local date.	MMDDYY
<code>\$(Date3)</code>	Local date.	DDMMYY
<code>\$(DateUS)</code>	Local date.	MMDDYYYY
<code>\$(YDate)</code>	Yesterday's date.	Local Date YYYYMMDD - 1 day
<code>\$(YDateUS)</code>	Yesterday's date in the US format.	Local Date MMDDYYYY - 1 day
<code>\$(SDD)</code>	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
<code>\$(SJ)</code>	The Julian day (ddd).	For example: 320
<code>\$(SMON)</code>	The month of the year to be specified using the first three	For example: JAN for the month of JANUARY

Token name	Description	Generated Value
	letters and upper case.	
\$(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
\$(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
\$(Time)	Local time.	HHMMSSMSS
\$(Time1)	Local time.	HHMMSS
\$(Time2)	Local time.	HHMMSST

Token name	Description	Generated Value
<code>\$(TransactionNumber)</code>	Local transaction number.	For example: <ul style="list-style-type: none"> Local file: <code>/home/usr/temp/files/testfile1.txt</code> Remote file: <code>c:\target\\$(TransactionNumber).testfile1.txt</code> Resolution like: <code>IA18100117.testfile1.txt</code>
<code>\$(UserData)</code>	The user data defined by the <code>cfsend</code> or <code>cfrecv</code> commands.	For example: <code>cfsend ud:AcctFile</code> The resolution to this token is <code>AcctFile</code> .

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/revenue.txt rf:'c:\reports\$(LocalFileBase).\$(Date1)-\$(Time1).\$(LocalFileExt)' node:TorontoNode
```

In the Windows system, the result is expected to be the `revenue.210929-095201.txt` file in the `c:\reports` directory.

PPA: Preprocessing and Postprocessing Actions

PPA is a set of commands implemented before and after a specified action. There are two types of PPA commands:

- Postprocessing actions
- Preprocessing actions

With PPA, you can use up to 4 preprocessing/postprocessing commands. Preprocessing commands execute before a transfer starts. Postprocessing commands execute after a file transfer is completed either successfully or unsuccessfully.

See the following PPA parameters you can use: PPA1, PPA2, PPA3, and PPA4.

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

PPA Actions For Platform Server Transfer Clients

Postprocessing Actions For Platform Server Transfer Clients

Postprocessing actions take place at the end of a transfer. Here are some uses for postprocessing actions:

- Process a file after it has been uploaded or downloaded.
- Notify a user when a file has been uploaded.
- Delete, or move a file after it has been downloaded.
- Notify an admin when a transfer fails.

Postprocessing actions can execute on the initiator (local) or the responder (remote). Postprocessing actions can execute based on a successful or failed transfer.

By default, postprocessing actions are synchronous; transfers will wait for the postprocessing actions to complete before the transfer completes. This applies to postprocessing actions executed locally or on a remote Platform Server for UNIX. If you want a postprocessing action to execute asynchronously (in a background), you can add an ampersand at the end of the command.

Example: `ppa1:"S,L,COMMAND,sleep 10&"`



Note: When a PPA is executed asynchronously, the PPA return code is not saved in the audit Log.txt record.

Preprocessing Actions for Platform Server Transfer Clients

Preprocessing actions are performed before Platform Server checks for the existence of a file. You can use preprocessing actions to perform action files such as to zip files prior to a

transfer and also to override the initiator or responder file name. Up to four preprocessing and postprocessing commands can be defined.

Example:

i **Note:** Zip a file before sending it, and unzip the file after successfully receiving the file.

```
LF: /tmp/mydir.zip
PPA:"P,L,COMMAND,myzip %DIR %LFILE"
PPA:"S,R,COMMAND,myunzip %DIR %LFILE"
```

P,L: Indicates that a preprocessing action is to be performed on the local system.

S,R: Indicates that a postprocessing action is to be performed on the remote system for a successful transfer.

Preprocessing Actions for Directory Transfers

Preprocessing actions for directory transfers can be executed in two places:

- Before the directory transfer starts
- Before each file transfer occurs

For a config.txt parameter, use the following command:

```
ExecPostProc: Child { Parent, Child } #used to be RunPPAEndDirTx { N, Y }
ExecPreProc:  Child { Parent, Child }
```

For CLI, use the following shortcuts:

```
epop      or      ExecPostProc
epep      or      ExecPreProc
```

i **Note:** If ExecPreProc or ExecPostProc is set to Parent, then %TRN for Dir Transfer is resolved as the transaction ID of the parent transfer. If ExecPreProc or ExecPostProc is set to Child, then %TRN for Dir Transfer is resolved as the transaction ID of the individual transfer.

On the responder side transfers, preprocessing action always runs on each transfer. Preprocessing action with override is not supported on the responder side. Parent level is useful only on the client side, only when you initiate a directory send or receive.

Transfer Retry Considerations for Preprocessing

When a transfer is retried after a recoverable error, or after the preprocessing action terminates the transfer with a destroyable error, preprocessing actions may be executed or may be bypassed. These criteria define when preprocessing actions are executed for a retried file transfer.

Initiator Transfer retry

- When a transfer retries because a preprocessing action sets the return code to a recoverable error (2, 3 or 4), all preprocessing actions will execute on a transfer retry.
- When all of the preprocessing actions have executed with return code 0 or 1, the processing actions are not executed on the transfer retry.

Responder Transfer retry

- When a transfer performs a checkpoint restart, preprocessing actions are not executed.
- When a transfer is retried and check restart is not performed, all preprocessing actions are executed.

Preprocessing and Postprocessing Use the Same PPA Parameters

Preprocessing and postprocessing actions share the same 4 PPA parameters.

Example:

```
PPA="S,L,COMMAND,PostCommand1 %lfile %lfile.%gdate.%time"  
PPA="P,L,COMMAND,pre-command %lfile"  
PPA="S,R,COMMAND,PostCommand2 %lfile"  
PPA="F,R,COMMAND,PostCommandFailure %lfile"
```

The order of preprocessing action is important because the return code determines the next step.

- 0: Continue with the transfer.
- 1: Override transfer parameters and continue with the transfer. Override will take place only if `override:FN` option was used. If no `override:FN` was used, then the transfer will continue as with `rc = 0`.
- 2-4: Stop the transfer and retry at the next interval. If all tries have been exhausted, terminate the transfer.
- >4: Stop the transfer with a permanent error case.

PPA1 is processed first, then PPA2, and so on.

Examples:

Two local PPAs are defined:

```
cfsend .... lf:/tmp/yz/test.tar rf:'/tmp/yz/${LocalFileName}.2'
PPA1:"P,L,COMMAND,touch /tmp/yz/test.small.3"
PPA2:"P,L,COMMAND,tar -cvf /tmp/yz/test.tar /tmp/yz/test.small*"
```

One local preprocessing action and one remote preprocessing action:

```
cfsend .... lf:/tmp/yz/test.tar rf:/tmp/yzp/'${LocalFileName}.2'
PPA1:"P,L,COMMAND,tar -cvf /tmp/yz/test.tar /tmp/yz/test.small*"
PPA2:"P,R,COMMAND,mkdir /tmp/yzp"
```

Two preprocessing actions and two postprocessing actions. Preprocessing action is requested at the parent level, postprocessing action is requested at the child level:

```
cfrecv .... lf:/tmp/yz/* rf:"/tmp/yzp/*"
PPA1:"S,L,COMMAND,touch /tmp/testI.transfer.%TRN"
PPA2:"P,L,COMMAND,mkdir /tmp/testI.prep"
PPA3:"S,R,COMMAND,touch /tmp/testR.transfer.%TRN"
PPA4:"P,R,COMMAND,mkdir /tmp/testR.prep" epep:parent epop:child"
```

Preprocessing Action with Override Option

Sometimes, you need to override certain transfer parameters (like the file name), based on a pre-PPA execution result. The `override:` options allow you to handle such a scenario by passing the `override` parameter to the PPA action app or script.

Syntax: "P,L,Command,Script.bat `override:OverrideFileName CLI_for_Script.bat`"

where *Script.bat* and *CLI_for_Script.bat* should be provided by a user.

where *OverrideFileName* should be provided and filled in by the user. *OverrideFileName* is used by PPA to read a list of parameters which should be overridden.

Example: Take `c:\zip*`, create a .zip file by using the custom `zipfiles.bat` script, assign the zip file name as `c:\temp\I41200001.zip`, and then, send the zip file using `cfsend LF:` parameter.

```
PPA="P,L,COMMAND,zipfiles.bat override:%trn.txt inzip:c:\zip*
outzip:c:\temp\%trn.zip"
```

If the program (`zipfiles.bat`) sets return code to 1, it must create a file defined by the `override:` parameter. In this case, it is something like `I41200001.txt` and the content should always be `field:value`.

When the return code from `zipfiles.bat` is 1, the following CLI parameters can be overridden by the values from the `override I41200001.txt` file:

```
LF: c:\temp\%trn.zip
UDATA: ziptest
```

LOCALFILE:	LFILE:	LF: Override Value
REMOTEFILE:	RFILE:	RF: Override Value
USERDATA:	UDATA:	Override Value
PROCESS:	PR:	Override Value

Below is a sample file:

```
LF:/tmp/test.zip
RF:c:/temp/test.zip
UDATA:PreProcessZip
PR:ProcTest
```

The following criteria are checked when using the `override:` option:

- If it is a preprocessing command
- If the PPA action is `COMMAND`
- If the PPA data contains parameter `Override:xxxx`

Example: This example fits the criteria.

```
PPA="P,L,COMMAND,zipfiles.bat override:myfile.txt inzip:c:\zip* outzip
c:\temp\%trn.zip"
```

Example: This example does not fit the criteria.

```
PPA="P,L,COMMAND,checkifile.bat %LFILE"
```

Each time the preprocessing action is called, it is recorded in the PCI log/message file.

i Note: PCI log files contain information about security events, configuration changes, server stop/start, preprocessing overrides, and other user actions.

PPA Command Format

You must follow the following command format to perform preprocessing and postprocessing actions after file transfers.

The preprocessing action command format is:

```
PPA<number>="P,L|R,COMMAND|CALLJCL|CALLPGM|SUBMIT,command data"
```

The postprocessing action command format is:

```
PPA<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:

- P|S||F
 - P: Preprocessing
 - S: Postprocessing on successful transfer
 - F: Postprocessing on failed transfer
- 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.

i Note: 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.

i Note: Spaces are allowed in PPA command data. Ensure that the entire PPA parameter is enclosed within quotation marks.

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.

i Note: PPA commands do not support standard transfer 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.prod.taxes.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.prod.taxes.txt
%HDIR	High level directory	a
%HLQ	High level qualifier of file	config
%LFILE	File name with directory	C:\a\b\c\d\config.prod.taxes.txt
%LLQ	Low level qualifier of file (data after the last period mark)	txt
%NODRIVE	File name without drive name	a\b\c\d\config.prod.taxes.txt
%NOHLQ	No high level qualifier of file	prod.taxes.txt
%NOLLQ	No low level qualifier of file (data before last period)	config.prod.taxes
%NOHDIR	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)	080929
%GDATEC	Gregorian date with century (ccyymmdd)	20210929

Parameter	Description	Resolved Name Examples (Based on C:\a\b\c\d\config.prod.taxes.txt)
%JDATE	Julian date (YYDDD)	08273
%JDATEC	Julian date with century (CCYYDDD)	2021273
%LUSER	Local user ID	
%PROC	Process name	ABC123
%RUSER	Remote user ID	
%TIME	Time (hhmmss)	165030
%TRN	Transaction number	I929800001
%UDATA	User data	UDABC123



Note: You can use multiple PPA substitutable parameters within a single PPA command. Substitutable parameters are processed one at a time.

PPA Error Codes

For PPA executed on the local machine, the PPA return codes are saved in the `Log.txt` file. PPA error messages are recorded if you set the trace level to medium (M) or high (H). To view the PPA return code, run it with `Wait For Completion` option which shows the # sign at the end. 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 remote PPA commands, the remote system transfer audit record contains the return code of the PPA commands.

For local PPA commands, the local transfer audit record contains the return code of the PPA commands.

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.

i **Note:** 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 `cfscmd` command:

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

Error Handling

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

The types of return 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 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
- 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 predefine the connectivity and authentication parameters. This makes the transfer 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` or the shared HA configuration 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 for 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 (for information purposes only)
- IP address or host name
- Port number
- Optional: security compliance level (used for HIPAA and FIPS mode transfers)
- Optional: netmask for incoming remote IP address
- Optional: netmask6 for incoming remote IPv6 address
- Optional: use of TLS/SSL and Tunnel mode 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 list, add, update, and delete node definitions.

i Note: 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 `Yes` at the `cfnode` prompt.

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

i Note: 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. At the prompt, set the values for both the required and optional parameters.

i Note: 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 comma-separated numbers for the MFT Command Center support:
1: support All, but Alter
```

```
8: support Alter
: 1
Do CRC checking for this node?
1: Yes
2: No
3: Default
: 2
A Node definition was created for:
  SystemType      = Windows
  Protocol        = tcpip
  HostName        = 192.168.0.44
  Server          = 46464
  TLS             = N
  Compression     = RLE
  Encryption      = BF
  SecurityPolicy  = Default
  LocalCTFile     = MyComtblg.dat
  AcceptVerifiedUser = Y
  ResponderProfile = D
  CommandSupport  = ALL
  CRC             = No
```

Node definitions are stored in the `cfnode.cfg` file located in the `$CFROOT/config` or the `HA config` directory. After a node definition is created, you can specify the name of the node you want to use for the transfer.

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 are listed.
3. Enter `cfnode a:list n:Node1`.
Node1 is listed.

Node Parameters

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

Required Node Parameters to Create a Node

i Note: If any of the required parameters is not defined and the `prompt` parameter is set to `No`, the `cfnode` command fails to create a node.

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</p>

Parameter (Alternate Specification)	Description
	<p>apply:</p> <ul style="list-style-type: none"> For initiator requests (that is, <code>cfsend/cfrecv</code>), 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> <p>Note: The node name is not case-sensitive.</p>
Port (p)	<p>Defines the port number on which the remote node is listening for incoming transfer requests.</p>
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 to Create a Node

i Note: 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 <code>fusping</code> 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. • ALTER: When <code>Alter</code> is set in the CC Node definition, then any incoming Command Center request to cancel an active transfer by its TRN is allowed. If this option is not set, then such a request is rejected. <code>ComandnCenter=ALL</code> does not include <code>Alter</code>. <code>Alter</code> must be added explicitly. For example: <div style="background-color: #e6f2ff; padding: 5px; margin-top: 10px;"> <pre>ComandnCenter=ALL,Alter</pre> </div>

Parameter (Alternate Specification)	Description
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 compression. This is not recommended due to its high CPU usage. • RLE: Run Length Encoding (RLE) compression. • ZLIB: ZLIB1 through ZLIB9. This is recommended as it causes an optimum balance between efficient compression and low CPU usage. ZLIB2 is recommended. • NO: no default compression. • NEVER: never use compression. <div style="border: 1px solid #ccc; background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: NEVER is the only value that cannot be overridden by <code>cfsend</code> or <code>cfrecv</code> on the command line.</p> </div>
CRC	<p>- (Yes NO Default)</p> <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. • Default: uses CRC value from <code>config.txt</code>
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>
Encrypt (e)	<p>Defines the encryption type for all transfers with this node.</p>

Parameter (Alternate Specification)	Description
	<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. • AES: AES/Rijndael, 256-bit encryption. • AES128: 128-bit encryption. • NO: no encryption. • NEVER: never uses encryption. <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Note: NEVER is the only value that cannot be overridden by <code>cfSEND</code> or <code>cfRECV</code> on the command line.</p> <p>AES128 is supported for transferring files to z/OS only. Since AES provides superior 256-bit encryption, it is a good practice to use AES.</p> <p>When using encryption, AES encryption is recommended due to its high level of security and efficient encryption.</p> </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 <code>No</code> or <code>None</code> or Translation File Name.</p> <p>When LCT file is given, custom A<->E translation is used.</p> <p>When LCT is <code>No</code>, the default A<->E translation is used.</p> <p>When LCT is <code>None</code>, then no ASCII < - > EBCDIC translation is performed on this system.</p> <p>The maximum length of the defined value is 16 characters.</p>

Parameter (Alternate Specification)	Description
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. Netmask is used to validate incoming requests against an IP address subnet, rather than against a specific IP address.</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. Netmask6 is used to validate incoming requests against an IPv6 Address subnet, rather than against a specific IP address.</p>
Prompt	<p>Defines whether to activate the <code>cfnode</code> command to an interactive mode.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • Yes: <code>cfnode</code> prompts the user for all information required to create a node. This is the default value. • No: <code>cfnode</code> 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 on the remote side. Valid values are <i>filename</i> and No. The maximum length of the defined value is 16 characters.</p>
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

Parameter (Alternate Specification)	Description
	<p>with the transfer request.</p> <ul style="list-style-type: none"> • No: Do not check the responder profiles. Validate the credentials against the operating system credentials. • Dual D: First, validate the credentials against the responder profiles. If no match is made, validate the credentials against the operating system credentials. • Do not define: ResponderProfile value from config.txt will be used.
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.
TLS/SSL	<p>Either T (Tunnel), No, or Yes depending on whether remote node requires a TLS/SSL connection.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • No: SSL mode is not used for initiator requests • Yes: SSL Mode is used for initiator requests • T: TLS Tunnel mode is used for initiator requests <p>The TLS/SSL parameter is ignored for responder/incoming requests.</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</p>

Parameter (Alternate Specification)	Description
	<p>the client sends an internal password in 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
cfrecv lf:/home/usr/file rf:"c:\temp\test.txt" n:windows
```

i Note: The UID and PWD parameters that define the credentials were left out of the `cfsend` and `cfrecv` examples. While these parameters are supported, we suggest creating a user profile to define the credentials.

i Note: 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 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 continue to execute the command if the user logs off before the `cfrecv` command is completed:

```
nohup cfrecv lf:/home/usr/file  
rf:"c:\temp\test.txt" n:windows
```

- Add the redirect 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 > /tmp/file 2>&1
```

Transfer Using Distribution Lists

With distribution lists, you can use a single command to send a single file 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 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.

 **Note:** 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, windowsVM
Directory=c:/tmp/xyz
```

Distribution List Parameters

The following table lists required parameters for the distribution lists.

Parameter	Description
distribution_list_name	<p>Defines the name of the distribution list.</p> <p>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.</p> <p>Note: The pound sign (#) cannot be used within a distribution list name and it has special meaning in the UNIX environment.</p>
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.

i Note: If the node uses different credentials, a user profile must be configured for each target node. User 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 servers can configure responder profiles to map the incoming user ID.

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

where the local file '/home/user/file' will be sent to the '/tmp/prod/data' directory for 'Store1' and 'Store2' nodes. The local file '/home/user/file' will be sent to 'c:\temp\test.txt' for 'Store5' node.

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

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.

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

User Profile

A User profile defines a remote user name and remote password that can be used by a local user. User profile definitions are saved in the `cfprofile.cfg` file.

A user profile contains the following information:

- **Node:** the remote system with which the user profile is associated
- **Local user name:** the name of the local user who can use this user 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 user profiles through the `cfprofile` command. Before you update any information in `cfprofile.cfg`, a backup file called `cfprofile.bak` is created. You can activate user profiles by simply specifying the node for a transfer.

Creating User Profiles

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

i Note: 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 user profile.

i Note: 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 user 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 User Profiles

You can list user 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 user profiles defined for TIBCO MFT Platform Server.

Deleting User Profiles

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

Procedure

1. Optional: On the command line, navigate to the `$CFROOT/bin` directory.
2. Enter `cfprofile a:delete node:nodename user:username`.

This deletes a user profile previously defined.

User Profile Parameters

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

Required User Profile Parameters

i Note: 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)	Defines the name of the node with which the user profile is associated.

Parameter (Alternate Specification)	Description
	<p>Note: The node name is not case-sensitive. The node must already exist before you can successfully add or update a user profile.</p>
Password (p)	Defines the password used to log on to the node (remote system).
User (u)	Defines the user name used to log on to the node (remote system).
	<p>Note: 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>

Optional User Profile Parameters

i Note: 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> <p>The default is: 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>

Parameter (Alternate Specification)	Description
Prompt	<p>Defines whether to execute the <code>cfprofile</code> command in an interactive mode.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • YES: <code>cfprofile</code> prompts you for all required information to create or update a user profile. This is the default value.
-? (/?)	<p>Displays online help for the <code>cfprofile</code> command.</p>

Note: Only the root account or members of the `cfadmin` group can use this option.

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.

If this parameter is set to `*ALL`, this user profile can be used by all local users who want to perform transfers with the defined node.

Note: You are prompted as to whether you want to create `cfprofile.cfg` when it is not found.

- No: if you do not want to be prompted, you can set this parameter to No.

Responder Profile

A responder profile defines the credentials used to validate the incoming user name and password. When a match is made, the Local User ID is used for the active file transfer. Each responder profile definition is defined in a clear text file named `cfprofile.cfg`.

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 (encrypted). 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 when a match is made on the incoming Node and UserId/Password credentials.

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.

When `PasswordRuleChecking` is turned on, the `ValidatePwdRules()` function for remote password is used to check the password given when the responder profile was created. *VER password is an exception from this check, it is not validated. During password validation, the password characters are compared with the rules provided in the `config.txt` file. For more information, see *Common Configuration Parameters*.

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.

i Note: You can create profiles for other users only if you are the root account or a member of the `cfadmin` group.

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.

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

```
cfprofile 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:

```
cfprofile prompt:YES

Enter a valid Node Name: dataServerA
Enter a valid Remote User: abc
Enter a valid Remote Password:
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 initiating transfer will set the User ID to and password to the values defined in the responder profile definition. The transfer is processed on the local machine using the local user ID `johndoe`.

Listing Responder Profiles

You can list responder 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 in the console the responder profiles defined for TIBCO MFT Platform Server.

Deleting Responder Profiles

You can delete responder profiles using the `cfprofile 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 table lists parameters supported for the `cfrprofile` command.

Required Responder Profile Parameters

i **Note:** 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> <p>Note: The node name is not case-sensitive. The node must already exist before you can successfully add or update a responder profile.</p>
Rpass (rp)	<p>Defines the password supplied by the initiator.</p> <p>If this responder profile is associated with a verified user, Rpass must be set to <code>*VER</code>.</p>
Ruser (r)	<p>Defines the user name supplied by the initiator.</p> <p>Note: 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>

Optional Responder Profile Parameters

i Note: The `cfrprofile` 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 `cfrprofile` command.

Parameter (Alternate Specification)	Description
<code>LocalUser (l luser)</code>	<p>Defines the local user used to process the transfer request initiated by the defined remote user.</p> <p>Note: Only members of the <code>cfadmin</code> group or the root user can define this parameter. Otherwise, this parameter is automatically set to the current user.</p>
<code>Action (a)</code>	<p>Defines the action you want to take.</p> <p>The valid values are: <code>Delete</code>, <code>List</code>, and <code>Add</code>.</p>
<code>Prompt</code>	<p>Defines whether to activate the <code>cfrprofile</code> command to an interactive mode.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • YES: <code>cfrprofile</code> prompts you for all required information to create or update a user profile. This is the default value. <p>Note: You are prompted as to whether you want to create <code>cfrprofile.cfg</code> 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 <code>No</code>.
<code>-? (/?)</code>	Displays online help for the <code>cfrprofile</code> command.

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

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`.

i Note: 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`.

i Note: If the remote system is Windows or UNIX, the `LocalFileName` parameter in the template is to define the name of the file to save 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`.

i Note: 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` is sent to the `dataServerB` node, even if another node has been defined in `TSEND` template.

i Note: 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 TRECVCV

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
# 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:           10                             { Transfer timeout in min }
ClassOfService:    Default                         { from cfcos.cfg }
CRC:               N                               { N, Y }
MaintainFileTimestamp: N                         { N, Y }
TraceLevelTransfer: D                             { D|Default, N, Y }

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

```

```

ExecPostProc:      D                { D|Default, Parent, Child }
ExecPreProc:       D                { D|Default, Parent, Child }

# 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:       N                 { Y, N, W }
UTF8BOM:        N                 { A, R, B, N }

```

See the following example of the TRECVC template:

```

# Sample template file for cfrecv command

LocalFileName:    /home/user/file
RemoteFileName:   c:\tmp\unix.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 }
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:         10                       { Transfer timeout in min }
ClassOfService:  Default                   { from cfcos.cfg }
CRC:             N                        { N, Y }
MaintainFileTimestamp: N                  { N, Y }
TraceLevelTransfer: D                     { D|Default, N, Y }

# Additional Directory Transfer Parameters
ScanSubDir:      N                        { N, Y }
StopOnFailure:   Y                        { N, Y }
Test:            N                        { N, Y }
ExecPostProc:    D                        { D|Default, Parent, Child }
ExecPreProc:     D                        { D|Default, Parent, Child }

# 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.

You can use the following types of parameters:

- [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 or IPName/IPAddress	Defines the destination host.
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. We suggest defining UserID and password credentials in a user profile and not on the <code>cfsend/cfrecv</code> command line.</p>

Note: Do not define this field when using User Profiles.

Parameter (Alternate Specification)	Description
RemoteFileName (rf)	<p>Defines the name of the file or data set at the remote location.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p>Note: When specifying the remote file name on a command line:</p> <ul style="list-style-type: none"> Platform Server for Windows will automatically convert slash (/) to a backslash (\). So, you can define file names using slashes and the slashes will be converted to backslashes as needed. 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>
UserId (uid)	<p>Defines the remote user ID for the file transfer. We suggest defining User ID and password credentials in a user profile and not on the cfsend/cfrecv command line.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p>Note: Do not define this field when using User Profiles.</p> </div> <p>For Windows security systems, you can use the format <i>domain\username</i> or <i>domain/username</i>.</p>

Optional Transfer Parameters

You can supply the optional transfer parameters (provided below in alphabetical order) as required when you run the cfsend or cfrecv command.

Parameter (Alternate Specification)	Description
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>

Parameter (Alternate Specification)	Description
	<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 style="border: 1px solid #ccc; background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: This parameter is used when sending files from UNIX to z/OS.</p> </div>
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 class of service for transfer.</p> <p>The valid values are class of service names defined in the <code>cfcos.cfg</code> file. These entries define the Send and Receive TCP Buffer sizes. Increasing these values can improve performance in a high latency environment.</p>
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>

Parameter (Alternate Specification)	Description
	<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.
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>
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> <div style="background-color: #f0f0f0; padding: 5px; margin: 10px 0;"> <p>Note: When you perform transfers to or from z/OS, you can use this parameter or the DELIM parameter.</p> </div> <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

Parameter (Alternate Specification)	Description
	<p>performed by TIBCO MFT Platform Server for UNIX .</p> <ul style="list-style-type: none"> • 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 replaces an existing file. This is the default option. • CA Y: creates a new file or appends to an existing file. • CRN Z: creates a new file, and if necessary, creates the directory path to this file or replaces an existing file.
EmailFailure (emf)	<p>Defines the email address or addresses to send an email when a transfer fails. You can specify multiple email addresses by delimiting the email addresses with a comma.</p>

Parameter (Alternate Specification)	Description
EmailSuccess (ems)	<p>Defines the email address or addresses to send an email when a transfer is successful. You can specify multiple email addresses by delimiting the email addresses with a comma.</p> <p>Note: 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 the SMTP email settings are configured in the config.txt file.</p>
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

Parameter (Alternate Specification)	Description
<p>Note: When encryption is required, it is a good practice to use AES (256-bit) encryption.</p>	
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>
epop OR ExecPostProc	<p>Defines whether the postprocessing action executes after each file in a directory or a distribution list which is transferred. The valid values are Child, Parent, and Default. The default value is Child.</p> <p>Child: postprocessing is performed after each transfer.</p> <p>Parent: postprocessing is performed after all files have been transferred.</p> <p>When this parameter is set to Parent, 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). <p>For more information, see Client Configuration Parameters.</p>
eep OR ExecPreProc	<p>Defines whether the preprocessing action executes before the start of the first transfer. The valid values are Child, Parent, and Default. The default value is Child.</p> <p>Child: Preprocessing is performed before each transfer.</p> <p>Parent: Preprocessing is performed once per directory transfer, before any individual file transfer is started.</p> <p>For more information, see Client Configuration Parameters.</p>

Parameter (Alternate Specification)	Description
IpName/Address (ip)	<p>Defines the host name or the IP address of the remote system. Do not use this parameter when the node parameter is defined.</p>
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. This parameter is only supported on send transfers.</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="602 842 1414 940" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: You must set the ASCII_to_EBCDIC parameter to Y to use this parameter.</p> </div> <p>For more information, see Conversion Tables/Custom Code Conversion.</p>
MaintainFiletimeStamp	<p>Defines the option to maintain file creation time stamps between servers. Valid values are Yes and No.</p> <p>When the value is set to Yes,</p> <ul style="list-style-type: none"> • if a file is sent, then the newly created file on the remote side will have the same date/time as the local one. • if a file is received, then the newly created file on the local side will have the same date/time as the remote one.
PermittedActions (pa)	<p>Defines the permitted actions for the transferred files.</p> <div data-bbox="602 1545 1414 1644" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The following values are Windows specific and are only valid when you transfer files to Windows.</p> </div> <p>The valid values are:</p> <ul style="list-style-type: none"> • S SYSTEM_FILE: a system file that can only be viewed by

Parameter (Alternate Specification)	Description
	<p>the operating system and not the user.</p> <ul style="list-style-type: none"> • 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. You should not define this parameter when the node parameter is defined.</p> <p>The valid values are from 1024 to 65535.</p>
Post_Action (ppa)	<p>Defines preprocessing or postprocessing the command you want to use.</p>
Post_Action1 (ppa1)	<p>You can define this command up to four times with the following command:</p>
Post_Action2 (ppa2)	
Post_Action3 (ppa3)	
Post_Action4 (ppa4)	
	<pre>ppa="P,L R,COMMAND,command data"</pre>
	<pre>ppa="S F,L R,COMMAND,command data"</pre>
	<p>Where:</p>
	<p>P means preprocessing</p>
	<p>S F means postprocessing success or failure.</p>

Parameter (Alternate Specification)	Description
	<p>L R means local or remote.</p> <p>COMMAND refers to the command you want to execute. 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 <code>Post_Action</code> command. If the remote system is a mainframe, then <code>CALLJCL</code>, <code>CALLPGM</code>, and <code>SUBMIT</code> parameters are also supported besides <code>COMMAND</code>.</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. This parameter is valid only when <code>TransfeType</code> is set to <code>Command</code>.</p> <p>The default value is <code>No</code>. The valid values are <code>N</code> or the command you want to use.</p>
RemoteCTFile (rct)	<p>Defines the name of the remote conversion table file.</p>

Parameter (Alternate Specification)	Description
RemotePrinterName (rp)	<p>Note: Note: When defining this parameter, you have to set the LocalCTFile as NULL, if you want all conversion to take place on the remote machine.</p> <p>For more information, see Conversion Tables/Custom Code Conversion.</p> <p>Defines the name of the remote printer to which you send the job.</p>
RetryInterval (ri)	<p>Defines the interval in minutes after which a retryable transaction can be retried.</p> <p>Note: 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. The default is to retry after waiting one minute.
ScanSubDir (ssd)	<p>Defines whether all subdirectories from the file path are scanned for files to transfer.</p> <p>Note: 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>

Parameter (Alternate Specification)	Description
SilentMode (sm)	<p data-bbox="621 352 1386 485">Note: 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> <p data-bbox="602 558 1377 627">Defines whether the progress message (in bytes) is displayed on the output screen on the initiator side.</p> <p data-bbox="602 659 862 686">The valid values are:</p> <ul data-bbox="651 718 1419 926" style="list-style-type: none"> <li data-bbox="651 718 1419 787">• Y: the progress message (in bytes) is not displayed on the output screen on the initiator side. <li data-bbox="651 816 1419 926">• N: the progress message (in bytes) is displayed along with the typical output on the screen. This is the default value. <p data-bbox="621 974 1373 1073">Note: You can best observe the transmission progress in bytes when the file you transfer is large. When a transfer is executed by DNI, you should set SilentMode to Y.</p>
StopOnFailure (sonf)	<p data-bbox="602 1146 1344 1215">Defines whether to stop transferring the rest of files in the directory when the current file transfer fails.</p> <p data-bbox="621 1262 1328 1331">Note: This parameter applies to directory transfers and distribution list transfers.</p> <p data-bbox="602 1356 948 1383">The valid values are Y or N.</p>
Template (t)	<p data-bbox="602 1436 1192 1463">Defines the file name of the transfer template.</p> <p data-bbox="621 1509 1292 1579">Note: This parameter cannot be set inside a transfer template.</p> <p data-bbox="602 1625 1284 1656">For more information, see Transfers Using Templates.</p>
Test	<p data-bbox="602 1703 1390 1772">Defines whether to display the local and remote file names to verify if the file names are correct, instead of doing the actual</p>

Parameter (Alternate Specification)	Description
	<p>transfers.</p> <p>Note: 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. This parameter overrides the TIMEOUT value configured in the config.txt.</p> <p>Once this amount of time is reached, the connection ends.</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 TLS/SSL communication • Y: uses TLS/SSL communication • T: uses TLS tunnel communication <p>For more information, see TLS/SSL Certificates Setup.</p>
TraceLevelTransfer	<p>Defines tracing to be set for a transfer on the initiator and on the responder. If the responder supports this parameter, it also turns on tracing for the transfer.</p> <ul style="list-style-type: none"> • Y: tracing is turned on for a transfer • N: tracing is not turned on for a transfer. This is the default value.
TransferType (trtype)	<p>Defines the type of transfer you want to perform.</p> <p>The default value is File.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • F File: you can send your local file to the remote file.

Parameter (Alternate Specification)	Description
	<ul style="list-style-type: none"> • J Job: 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. • C Command: you can send the command to the remote system and receive a result. If you specify <code>LocalFileName</code>, 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 style="border: 1px solid #ccc; background-color: #f0f0f0; padding: 5px; margin: 10px 0;"> <p>Note: The C Command option only works with the <code>cfsend</code> command.</p> </div> <ul style="list-style-type: none"> • P Print : 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 (<code>trunc</code>)	<p>The valid values are:</p> <ul style="list-style-type: none"> • YES: when a record longer than the z/OS LRECL is received, the record is truncated. • NO: when a record longer than the z/OS LRECL is received, the transfer is terminated with an error. • WRAP: when a record longer than the z/OS LRECL is received, the record is split into multiple records.
TryNumber (<code>trynum</code>)	<p>Defines the number of times the transfer can be attempted.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • 0 U UnLimited: the transfer can be attempted 9999 times or until it is successful. It restarts the transfer from the beginning unless the <code>CheckPoint/Restart</code> option is set.

Parameter (Alternate Specification)	Description
	<ul style="list-style-type: none"> • N 1: the transfer can be attempted only one time. The default value is 1. • 2 to 9998: the transfer can be attempted for the specified number of times.
UNIXPermissions (uperm)	<p>Defines the UNIX permissions for the file. 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, operating system UMASK parameter is used to set the file permissions. If no values are passed in the control record, the responder uses the system default permissions.</p> <div data-bbox="602 1228 1409 1402" style="background-color: #f0f0f0; padding: 10px;"> <p>Note: 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 a 25-character field for user comments, and it can contain any alphabetic, numeric, or national characters.</p>
UTF8BOM (bom)	<p>When performing iconv conversion on Platform Server for z/OS, defines if BOM (Byte Order Marks) are added to, or removed</p>

Parameter (Alternate Specification)	Description
	<p>from, the target file.</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.

z/OS Specific Transfer Parameters

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

i Note: 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>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>

Parameter (Alternate Specification)	Description
	<p>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>
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.
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>
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>

Parameter (Alternate Specification)	Description
DELIM (cr)	<p>Defines the file delimiter. This parameter is the same as the CRLF parameter.</p> <p>Note: The first character must be an alphabetic or national character.</p> <p>In addition, you can use this parameter to indirectly select file attributes such as record format and logical record length.</p> <p>Note: 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> <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 execute.
LENGTH (orl lrecl)	<p>Defines the remote file record length.</p> <p>The valid values are from 1 to 32760.</p>

Parameter (Alternate Specification)	Description
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>
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 (orf)	<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

Parameter (Alternate Specification)	Description
	<ul style="list-style-type: none"> • 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>
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> <p>Note: This parameter is only supported for send transfers to a z/OS system.</p>
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> <p>Note: This parameter is only valid when you receive a file using the <i>cfrcv</i> command from a z/OS system.</p>

Parameter (Alternate Specification)	Description
STORCLASS (sc)	<p data-bbox="529 338 1218 407">Defines the z/OS storage class as specified in the Data Facility/System Managed Storage.</p> <p data-bbox="529 436 1373 548">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 data-bbox="529 577 1341 646">The maximum length of the defined value is 8 characters, it can contain numeric, alphabetic, or national characters (\$, #, @).</p> <div data-bbox="548 688 1305 758" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The first character must be an alphabetic or national character.</p> </div>
SUBMIT (sj)	<p data-bbox="529 831 1057 858">Defines the z/OS JCL you want to submit.</p> <p data-bbox="529 888 1271 915">The valid values are N or a file that contains the z/OS JCL.</p>
SysOutClass (s1)	<p data-bbox="529 974 1187 1001">Defines the class to which the JES output is routed.</p> <p data-bbox="529 1031 1393 1142">On z/OS systems, the printer queues are organized around a printer class instead of a specific printer. This class has a 1-character name that is either alphabetic or numeric.</p>
SysOutCopies (sp)	<p data-bbox="529 1194 1357 1264">Defines the number of copies to print of a particular report on a remote computer.</p> <div data-bbox="548 1306 1357 1375" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: This parameter is only valid when the remote platform is z/OS.</p> </div>
SysOutDestination (sd)	<p data-bbox="529 1446 1330 1516">Defines the destination for the job that you submit to the z/OS internal reader.</p> <div data-bbox="548 1558 1357 1627" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: This parameter is only valid when the remote platform is z/OS.</p> </div>
SysOutFcb (sb)	<p data-bbox="529 1698 1336 1726">Defines the name of the form control buffer as specified in JES.</p>

Parameter (Alternate Specification)	Description
SysOutUserName (si)	Defines the user name assigned to a job that you submit to the z/OS internal reader. <div data-bbox="548 352 1357 417" style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> Note: This parameter is only valid when the remote platform is z/OS. </div>
SysOutWriter (sw)	Defines the external writer name that you use to process this printer file on z/OS. <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 data-bbox="548 999 1341 1064" style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> Note: Do not specify a value for this parameter unless you are instructed by the system analyst of the z/OS system. </div>
UNIT (du)	Defines the remote file z/OS unit name. <p>The maximum length of the defined value is 8 characters.</p>
VOLUME (dv vol volser)	Defines the remote file z/OS volume name. <p>The maximum length of the defined value is 6 characters, and it must be alpha-numeric.</p>
zOS	Defines whether you can perform transfers to or from z/OS. <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. <p>Note: Accelerator uses a proprietary compression compatible with ZLIB compression.</p>
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>

Parameter (Alternate Specification)	Description
ACCMaxSpeed (ACCMAX)	<p>Note: 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.</p> <p>For more information, see Configuration Parameters.</p>
ACCPort	<p>Defines the port number on which the platform server listens for transfers using the Accelerator technology.</p> <p>The valid values are N or <i>port_number</i>. Default value is 9099.</p> <p>Note: 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 N and you have Accelerate set to Y, the value configured for ACCPort in <code>config.txt</code> file is used.</p> <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

Parameter (Alternate Specification)	Description
	Protocol
	• PDP: Parallel Delivery Protocol

i **Note:** TIBCO Accelerator support on Linux is discontinued from version 8.1.0. However, the client part is still preserved for now. This means PSU transfers can be accelerated or rerouted to TIBCO Accelerator Server on a PSW machine.

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 located in the `$CFROOT/config` directory. You can use a text editor to modify the configuration file.

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.

Changes in these fields are reflected in real time, except for the following fields:

- ListenAdapterIP
- ListenAdapterIPv6
- Port
- PortIPv6
- SSLPort
- SSLPortIPv6
- TunnelPort
- TunnelPortIPv6
- ConnectAdapterIP
- ConnectAdapterIPv6

You must restart CyberResp for changes in the above fields.

Server Configuration

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:             10                   { 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 (in alphabetical order) that are used to configure TIBCO MFT Platform Server as a responder.

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>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>
AllowRoot	<p>Defines whether the root account is considered as a valid user ID for incoming transfers.</p> <p>Allowed values are:</p> <ul style="list-style-type: none"> • All: Incoming transfers can execute as Root • No: Incoming transfers cannot execute as Root. This is the suggested value. • Password: Incoming transfers can execute as root only when the root password is validated. <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>, the root account can be used as a valid user ID when the password is provided. • When AllowRoot is set to <i>all</i>, the root account can be used as a valid user ID.

Parameter Name	Description
	<ul style="list-style-type: none"> When <code>AllowRoot</code> is set to <code>no</code>, the root account cannot be used as a valid user ID.
<code>ClassOfService</code>	Defines the default Class of Service entry name.
<code>ConvTbl</code>	<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 <code>Comtblg.dat</code> which is located in the <code>\$CFROOT</code> directory. For more information, see Conversion Tables.</p>
<code>ExitPrgm</code>	<p>Defines the path to the Exit program on the local machine.</p> <p>Using the Exit program, you can do customized post processing. For more information, see User Exits.</p>
<code>ListenAdapterIP</code>	<p>Defines the IP address at which the responder binds to listen 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>
<code>ListenAdapterIPv6</code>	<p>Defines the IPv6 address at which the responder binds to listen for incoming IPv6 connections.</p> <p>The default value for this parameter is <code>All</code>, which means IPv6 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>
<code>PamAuth</code>	Defines the PAM service file name or the PAM service entry name.

Parameter Name	Description
	<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>
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 N or a number ranging from 1024 to 65535, because lower ports are usually reserved for standard applications.</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 incoming IP address requires a defined node. If the incoming address is not defined in a node, the responder rejects the transfer request and sends back an error message.</p>
ResponderProfile	<p>Defines whether credentials for incoming requests are validated against the responder profiles or the operating systems credential store.</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 perform a transfer.</p> <div data-bbox="602 1598 1409 1734" style="background-color: #f0f0f0; padding: 10px;"> <p>Note: ResponderProfile checking routine is always done before AcceptVerifiedUser checking. So if both are set up, ResponderProfile takes precedence.</p> </div>

Parameter Name	Description
	<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.</p> <p>Note: The ResponderProfile parameter of the incoming Node definition overrides the Global ResponderProfile setting.</p>
RunCyberRespAsNonRoot	<p>Defines whether you can run the CyberResp daemon as a non-root user.</p> <p>Note: 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>
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>Note: Once the timeout value is reached, the connection ends.</p>
TraceLevel	<p>Defines the level of tracing.</p> <p>The valid values are N, Low L, Medium M, or High H. The default value is N.</p> <p>Usually tracing has only to be turned on at the request of for troubleshooting purpose.</p> <p>Note: When the TraceLevel is used, the TraceSizeServer must be defined.</p>
TracePath	<p>Defines the name of the path that holds the responder trace</p>

Parameter Name	Description
	<p>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 <code>cfstart</code> 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>
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> <div data-bbox="602 1199 1414 1339" style="background-color: #f0f0f0; padding: 10px;"> <p>Note: When <code>TraceSizeServer</code> is used, <code>TraceLevel</code> in the server part must be defined.</p> </div>
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 <code>Umask_Default</code> is set to N, then the file permission can be set according to the operating system user mask. There is a <code>Umask_User</code> parameter in CLIENT section for Initiator transfers.</p>

Parameter Name	Description
	<p>Note: There is no command line or template option for <code>Umask</code>. This parameter is only contained in the <code>config.txt</code> file.</p>
<code>Uperm_Default</code>	<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 <code>Uperm_Default</code> parameter is used when a user sends a new file to the UNIX server and has not provided the <code>uperm</code> parameter. (This is not possible on a UNIX initiator, as the <code>uperm</code> parameter is set by default to the file permissions of the sending file).</p> <p>If the <code>Uperm_Default</code> is set to N, the file permissions for the newly created file are set according to the <code>Umask_Default</code> parameter.</p> <p>Note: 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>

Server SSL Communications Parameters

The following table lists parameters (in alphabetical order) that are only used when performing SSL or Tunnel transfers.

Parameter Name	Description
<code>AuthorizationFileName</code>	<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</p>

Parameter Name	Description
CAPath	<p>Platform Server does not perform additional authentication to the client certificate. This parameter is only valid when <code>ClientVerification</code> 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>
CertificateFileName	<p>Defines the path to the certificate file used for an SSL or Tunnel transfer.</p> <div data-bbox="634 915 1411 1056" style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p>Note: 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>
CheckCRL	<p>Defines whether incoming TIBCO MFT Platform Server SSL or Tunnel checks the <code>CAPath</code> field for the hashed CRL files.</p> <p>For more information, see CRL Support.</p>
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 <code>openssl</code> command.</p>

Parameter Name	Description
	<p>Examples:</p> <p>List TLS Ciphers:</p> <pre data-bbox="634 411 1414 489">\$CFROOT/util/openssl ciphers -tls1</pre> <p>List "high" encryption TLS Cipher suites:</p> <pre data-bbox="634 569 1414 646">\$CFROOT/util/openssl ciphers -tls1 HIGH</pre>
ClientVerification	<p>Defines whether TIBCO MFT Platform Server performs SSL client authentication.</p> <p>The valid values are:</p> <ul data-bbox="683 856 1357 982" 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>
PrivateKeyFileName	<p>Defines the path to the file with the private key that is associated with the SSL certificate.</p> <div data-bbox="634 1192 1414 1339" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: 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 createPwd.exe file in the \$CFROOT/util directory.</p>

Parameter Name	Description
SSLEnabledProtocols	<p data-bbox="651 310 1390 447">Note: 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> <p data-bbox="634 499 1256 531">For more information, see SSL Certificates Setup.</p> <p data-bbox="634 575 1308 651">Defines which SSL protocols are supported when the platform server runs as a responder.</p> <p data-bbox="634 680 1317 751">The valid values are any combination of the following options:</p> <p data-bbox="634 781 1268 852">TLSV1,TLSV1.1,TLSV1.2. The default value is TLSV1,TLSV1.1,TLSV1.2. The comma means "and".</p>
SSLPort	<p data-bbox="634 898 1349 970">Defines the IP port on which TIBCO MFT Platform Server listens on for incoming SSL requests.</p> <p data-bbox="634 999 1414 1115">The valid values are from 1024 to 65535, because lower ports are usually reserved for standard applications. The default value is 56565.</p>
SSLPortIPv6	<p data-bbox="634 1161 1377 1232">Defines the IPv6 port on which TIBCO MFT Platform Server listens on for incoming SSL requests.</p> <p data-bbox="634 1262 1409 1413">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 data-bbox="634 1442 1385 1556">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 data-bbox="634 1585 1365 1619">This field must be different than the PortIPv6 parameter.</p>
SSLTraceLevel	<p data-bbox="634 1665 1349 1698">Defines whether tracing is turned on for an SSL transfer.</p> <p data-bbox="634 1728 1365 1757">Usually tracing has only to be turned on at the request of</p>

Parameter Name	Description
	TIBCO Support for troubleshooting purpose.
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.</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 server and a TIBCO MFT Platform Server client.</p> <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p>Note: 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>
TunnelPort	<p>Defines the IP port on which 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 the 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 <code>PortIPv6</code> parameter.</p>

Client Configuration

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: 10 { Transfer timeout in min }
ExecPostProc: Child { Parent, Child }
ExecPreProc: Child { Parent, Child }
ACCHost: N { N, Host }
ACCPort: 9099 { Port Number }
ClassOfService: Default { from cfcos.cfg }
StopOnFailure: Y { Y, N }
MaintainFileTimestamp: N { N, Y }

# 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 (in alphabetical order) that are used by TIBCO MFT Platform Server to configure cfsend and cfrecv client (i.e. Initiator) transfers.

Parameter	Description
ACCHost	Defines the IP address or host name of the Remote Accelerator server.

Parameter	Description
ACCPort	<p>Defines the port number on which the Remote Accelerator server listens for transfers using the Accelerator technology.</p> <p>The default number is 9099.</p>
ClassOfService	<p>Defines the default class of service entry name for initiator transfers.</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>
ExecPostProc	<p>Defines whether to have the postprocessing action executes after each file in a directory is transferred or after the entire directory is transferred.</p> <div data-bbox="602 1455 1414 1669" style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p>Note: If ExecPreProc or ExecPostProc is set to Parent, then %TRN for Dir Transfer is resolved as the transaction ID of the parent transfer. If ExecPreProc or ExecPostProc is set to Child, then %TRN for Dir Transfer is resolved as the transaction ID of the individual transfer.</p> </div> <p>The valid values are Child or Parent. The default value is Child.</p>

Parameter	Description
ExecPreProc	<p>Child: Postprocessing is performed after each transfer.</p> <p>Parent: Postprocessing is performed after all files have been transferred.</p> <p>When this parameter is set to Parent, the following rules apply:</p> <ul style="list-style-type: none"> • <code>StopsOnFailure</code> is automatically set to Yes. Transfer stops on the first failed transfer. • <code>Failure PPA</code> runs on the first failed transfer. • <code>Successful PPA</code> runs only on the last transfer (assuming it is successful). <p>Note: Since this is a global parameter, it affects all transfers. This parameter can be overridden on the <code>cfsend</code> or <code>cfrecv</code> command line by the <code>epop</code> parameter.</p>
MaintainFiletimeStamp	<p>Defines whether to have the preprocessing action executes before each file in a directory or only once before any file transfer starts.</p> <p>The valid values are <code>Child</code> or <code>Parent</code>. The default value is <code>Child</code>.</p> <p>Child: Preprocessing is performed before each file transfer.</p> <p>Parent: Preprocessing is performed once per directory transfer, before each individual file transfer is started.</p> <p>Note: Since this is a global parameter, it affects all transfers. This parameter can be overridden on the <code>cfsend</code> or <code>cfrecv</code> command line by the <code>epep</code> parameter.</p> <p>Defines the option to maintain file creation time stamps between servers. Valid values are <code>Yes</code> and <code>No</code>.</p> <p>When the value is set to <code>Yes</code>,</p>

Parameter	Description
	<ul style="list-style-type: none"> • if a file is sent, then the newly created file on the remote side will have the same date/time as the local one. • if a file is received, then the newly created file on the local side will have the same date/time as the remote one.
RequiredNodeDefinition	<p>Defines whether a node definition is required for TIBCO MFT Platform Server when initiating transfers to a remote system.</p> <p>Value Y means that the <code>cfsend</code> or <code>cfrecv</code> requires a defined node. If the node parameter is not defined, the initiator rejects the transfer and displays an error message.</p>
StopOnFailure	<p>Defines the default value for <code>StopOnFailure</code> when this parameter is not specified by the <code>cfsend</code> or <code>cfrecv</code> commands. <code>StopOnFailure</code> defines whether a directory transfer or distribution list transfer should stop on the first failure (Y) or continue if a transfer fails.</p>
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 10 minutes. Once the timeout value is reached, the transfer terminates with a retry-able error.</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

Parameter	Description
TracePathClient	<p data-bbox="683 296 1300 321">information provided by the medium level trace.</p> <p data-bbox="602 359 1338 426">Usually tracing has only to be turned on at the request of TIBCO Support for troubleshooting purpose.</p> <div data-bbox="602 457 1414 684" style="background-color: #f0f0f0; padding: 10px;"> <p data-bbox="623 478 695 504">Note:</p> <ul data-bbox="672 512 1317 663" style="list-style-type: none"> <li data-bbox="672 512 1317 579">• This parameter cannot be used within a transfer template. <li data-bbox="672 600 1219 663">• When the <code>TraceLevelClient</code> is used, the <code>TraceSizeClient</code> must be defined. </div> <p data-bbox="602 730 1393 863">Defines the name of the path that holds the client trace file.</p> <p data-bbox="602 793 1393 863">A unique trace file is created for each file transfer. A file name contains the local transaction number and Process ID (PID).</p> <div data-bbox="602 894 1414 995" style="background-color: #f0f0f0; padding: 10px;"> <p data-bbox="623 915 1317 974">Note: This parameter cannot be used within a transfer template.</p> </div>
TraceSizeClient	<p data-bbox="602 1045 1230 1071">Defines the size limit of the trace file in kilobytes.</p> <p data-bbox="602 1108 1414 1373">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 data-bbox="602 1404 1414 1631" style="background-color: #f0f0f0; padding: 10px;"> <p data-bbox="623 1425 695 1451">Note:</p> <ul data-bbox="672 1459 1382 1610" style="list-style-type: none"> <li data-bbox="672 1459 1317 1526">• This parameter cannot be used within a transfer template. <li data-bbox="672 1547 1382 1610">• When <code>TraceSizeClient</code> is used, <code>TraceLevelClient</code> in the client part must be defined. </div>
Umask_User	<p data-bbox="602 1680 1414 1747">Defines whether to apply the user's umask to the incoming files (<code>cfrecv</code> only).</p>

Parameter	Description
	<p>Note: This parameter only applies to the initiator that is doing a receive transfer.</p> <ul style="list-style-type: none"> • If <code>Umask_User</code> is set to N, the user's umask is ignored on incoming files. • If <code>Umask_User</code> is set to Y, the user's umask is applied to incoming files.

Client SSL Communications Parameters

The following table lists parameters that are only used when performing SSL or Tunnel transactions.

Parameter	Description
<code>CertificateFileName</code>	<p>Defines the path to the certificate file used for an SSL and Tunnel 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>
<code>PrivateKeyFileName</code>	<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>
<code>PrivateKeyPwdFileName</code>	<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</p>

Parameter	Description
	<p>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 data-bbox="634 1037 1414 1178" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: This parameter cannot be used within a transfer template.</p> </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 any SSL-related problems with TIBCO Support.</p> <div data-bbox="634 1461 1414 1566" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: 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>

Parameter	Description
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 by Platform Server Initiator.</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>
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.</p>

Common Configuration

The common configuration defines parameters used by both initiator and responder transfers.

```
# System Configuration File.

# [ COMMON ]
SecurityPolicy:          N                { None, HIPAA, FIPS140 }
ConfigDirectory:        /mftps/config
PQFDirectory:           /mftps/PQF                { DirName }
TransnumFileName:       /mftps/trn/transnum        { FileName }
LogEventFileName:       /mftps/log/Log.txt         { N, FileName }
LogMessageFileName:     /mftps/log/message/Message.txt { N, FileName }
LogAdminFileName:       /mftps/log/admin/Admin.txt   { N, FileName }
AuditTempErrors:        N                    { N, Y }
SMTPServer:             N                    { IpName/Address:Port, N }
FromAddress:            N                    { Email Address, N }
Subject:                N                    { Subject String, N }
```

```

SuccessSubject:      N                { Subject String, N }
FailureSubject:     N                { Subject String, N }
CfgPostProc:        N                { N, FileName }
AccessControlConfig: N                { N, FileName }
AliasConfig:         N                { N, FileName }
AdminGroup:          cfadmin          { group name }
BrowseGroup:         cfbrowse         { group name }
TransferGroup:       cftransfer       { group name }
TraceGroupMember:   N                { N, Y }
StrictGroupChecking: N              { N, Y }
LogDirectoryTransfers: Y              { Y, N, Errors }
Encode:              N                { non utf8 code page }
CRC:                 N                { Y, N }

# CyberMgr Rpc Service
CyberMgrPortLocal:  46678             { PortNumber }
CyberMgrTraceLevel: N                {N, Yes|Y, Detailed|D }
CyberMgrTracePath:  /mftps/trace/Responder { N, Path }
SemaphoreKey:       0x07e9368b
SemaphoreMaxWaitTime: 20             { 10 - 120 seconds }
VRefreshInterval:  0                  { # of sec, 0 turn off visibility }
RpcSynchIntervalHA: 60               { # of sec, 0 turn off time synch }
RpcMaxWaitConnectTimeHA: 3           {# of sec, 0 unlimited wait connect }
HADirectory:        /mnt/HADir        { N, DirName }
HACyberMgrPrimary:  10.97.XXX.X:46678 { IpName/Address:Port }
HACyberMgrSecondary: 10.97.XXX.Y:46678 { IpName/Address:Port }

# Password Rules for Responder Profile
PasswordRuleChecking: N              { Y, N }
PasswordRequireUpperAndLower: N      { Y, N }
PasswordMinLength:   8                { 3 - 64 }
PasswordMinUnique:   3                { 0 - PasswordMinLength }
PasswordMinLetters:  3                { 0 - PasswordMinLength }
PasswordMinNumber:   0                { 0 - PasswordMinLength }
PasswordMinSpecial:  0                { 0 - PasswordMinLength }

# Parse Commands
protect_cctransfer:  exec              { none|reject|exec }
protect_cfdir:       reject            { none|reject|exec }
protect_fusutil:     reject            { none|reject|exec }
protect_receivdir:   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 (in alphabetical order) that are used to configure all transfer requests.

Parameter	Description
AccessControlConfig	Defines the path to the AccessControl.cfg in the

Parameter	Description
	<p><code>CFROOT/config</code> directory.</p> <p>You can change the default directory for a file based on the <code>USERID</code>, <code>NODE</code>, or <code>IPADDR</code> parameters on responder transfer requests only.</p> <p>For more information, see Access Control.</p>
<code>AdminGroup</code>	<p>Defines the group name that holds users who can configure nodes, profiles, and responder profiles, as well as view audit records from all users.</p>
<code>AliasConfig</code>	<p>Defines the path to the <code>CfAlias.cfg</code> file in the <code>CFROOT/config</code> directory.</p> <p>You can use an alias file name based on the <code>USERID</code>, <code>NODE</code> or <code>IPADDR</code> parameters for responder transfer requests only.</p> <p>For more information, see CfAlias.</p>
<code>AuditTempErrors</code>	<p>Defines whether all transfer attempts or only the final attempt is logged.</p>
<code>BrowseGroup</code>	<p>Defines the group name that holds users who can view audit records from all users.</p> <div style="border: 1px solid #ccc; padding: 5px; background-color: #f9f9f9;"> <p>Note: If this group exists, then users who are not in the specified browse group can only view transactions that they conducted.</p> </div>
<code>CfgPostProc</code>	<p>Defines the name of the file that holds the postprocessing configuration.</p> <p>For more information, see Configured Post Processing.</p>
<code>ConfigDirectory</code>	<p>Defines the path to a directory with all configuration files (with exception of <code>config.txt</code> file). By default, this is the <code>CFROOT/config</code> folder. This might be different</p>

Parameter	Description
	<p>in HA mode, or in docker container environment.</p> <p>Note: Config.txt is always located in \$CFROOT/config, regardless of ConfigDirectory value.</p>
CRC	<p>Defines whether to perform a CRC check.</p> <p>The valid values are N or Y.</p>
CyberMgrPortLocal	<p>Defines the local port of CyberMgr. The default port number is 46678.</p>
CyberMgrTraceLevel	<p>Defines the trace level of CyberMgr. Valid values are Y, N or Detailed. The default value is N.</p> <p>CyberMgrTraceLevel: Y {N, Yes Y, Detailed D}</p>
CyberMgrTracePath	<p>Defines the path of the trace file of CyberMgr. The default path is /mftps/trace/Responder.</p>
Encode	<p>Defines how file names are translated when sent to or received from 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</p>

Parameter	Description
	<p>characters, then you should set this parameter to the character set of the local UNIX machine.</p> <p>Encode : EUCKR</p> <p>Note: Encode works same way as iconv Unix command. Encode works only for the <code>FileNames</code>. It is not applicable to the file data.</p>
FailureSubject	<p>Defines the email subject for a failed transfer.</p> <p>The valid values are N or any string value. The default value is N.</p>
FromAddress	<p>Defines the value of the From field in the email notification.</p>
HADirectory	<p>HADirectory: /mnt/HADir { N, DirName }</p> <p>This parameter defines the directory where the HA files are located. All systems that want to be in the same HA Cluster must have access to this directory.</p>
HACyberMgrPrimary	<p>IpName/Address:Port</p> <p>This parameter will appear in config.txt only if you have converted to HA mode. It defines the Primary CyberMgr Rpc Server host:port.</p> <p>Note: It is very important that the values are identical on all machines participating in an HA cluster.</p>
HACyberMgrSecondary	<p>IpName/Address:Port</p> <p>This parameter will appear in config.txt only if you have converted to HA mode. It defines the Secondary CyberMgr Rpc Server host:port.</p>

Parameter	Description
	<p>Note: It is very important that the values are identical on all machines participating in HA cluster.</p>
LogAdminFileName	<p>Defines the name of the log admin message file. Valid values are <code>N</code> and <code>file name</code>. The default path and name is <code>/mftps/log/admin/Admin.txt</code>.</p> <p>One file is created per day, in the format of <code>Admin.txt.YYYYMMDD</code>.</p>
LogDirectoryTransfers	<p>Defines whether to log <code>cfdir</code> requests when doing directory transfers.</p> <p>The valid values are <code>Y</code>, <code>N</code> or <code>Errors</code>. The default value is <code>Y</code>. <code>Errors</code> means the <code>cfdir</code> request is logged only when an error occurs.</p>
LogEventFileName	<p>Defines the name of the file that holds the transaction history log. When running in a container, this parameter should point to a file in persistent storage.</p> <p>By default, the log file is <code>\$CFROOT/log/Log.txt</code>.</p> <p>One file is created per day in the format of <code>Log.txt.YYYYMMDD</code>.</p>
LogMessageFileName	<p>Defines the name of the log message file. Valid values are <code>N</code> and <code>file name</code>. The default path and name is: <code>/mftps/log/message/Message.txt</code></p> <p>One file is created per day, in the format of <code>Message.txt.YYYYMMDD</code>.</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. When running in</p>

Parameter	Description
	<p>HAMode, this parameter must define persistent storage that is accessible to all Platform Server instances in the HA cluster. Otherwise, transfer restart may fail.</p>
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. <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>Note: 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> </div>

Parameter	Description
SemaphoreKey	<p data-bbox="711 310 1308 378">Note: If the SecurityPolicy is set to FIPS140, all CyberResp daemons must be restarted.</p> <p data-bbox="691 445 1256 470">Defines the key used to create a semaphore.</p> <p data-bbox="691 508 1380 697">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.txt file.</p> <p data-bbox="691 730 1380 877">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>
SemaphoreMaxWaitTime	<p data-bbox="691 928 1380 1352">Defines how long to wait if the lock is taken by another thread. Checking the semaphore (also known as "lock") by each thread is done at 0.25 sec intervals upto the configured SemaphoreMaxWaitTime. If you have a high-volume transfer environment and consider that some logging requests might be 'lost/not logged' because the maximum wait time has passed and the Log.txt file is still taken by other threads, you should gradually increase the value. Valid values are from 10 - 120 seconds (2 minutes). The default value is 20 seconds.</p> <p data-bbox="691 1386 1299 1453">To reset SemaphoreMaxWaitTime value without recycling CyberMgr:</p> <ol data-bbox="727 1491 1380 1667" style="list-style-type: none"> <li data-bbox="727 1491 1299 1558">1. Open the config.txt file and change the value. <li data-bbox="727 1596 1380 1663">2. Enter <code>cfinq mgr=u</code> to tell CyberMgr daemon to reset the SemaphoreMaxWaitTime value.
SMTPServer	Defines the name of the email server and the port

Parameter	Description
	<p>that is used to send out email notifications.</p> <p>The format to define the port is:</p> <pre>your.smtp.server:port</pre> <p>If the port is not defined, it defaults to port 25.</p> <p>Example:</p> <pre>your.smtp.server:25</pre>
StrictGroupChecking	<p>Defines if strict group checking is required. If you want to deny certain requests when 'cftransfer' and 'cfbrowse' group were not created, then turn on this parameter. The default value is N.</p>
Subject	<p>Defines the subject line of the email notification.</p> <p>The maximum length of the defined value is 256 characters.</p>
SuccessSubject	<p>Defines the email subject for a successful transfer.</p> <p>The valid values are N or any string value. The default value is N.</p>
TraceGroupMember	<p>Defines whether to trace all system calls that check whether or not user is a member of a certain group. Set this parameter only when directed to by TIBCO Support.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • N: tracing is not turned on. This is the default value. There might be a lot of system calls so tracing should be done only 'on demand'. • Y: tracing is turned on and all system calls issued to determine a user's membership group are traced in log/admin or log/message PCI files.

Parameter	Description
TransferGroup	<p>Defines the group name that holds users who can conduct platform to platform file transfers initiated from Command Center.</p> <div data-bbox="711 453 1349 659" style="border: 1px solid #ccc; padding: 5px; background-color: #f9f9f9;"> <p>Note: If this group does not exist and a transfer request comes in from Command Center, the transfer can succeed. 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>
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 must be set when running in a container because this file must be saved in persistent storage. Otherwise, you should use the default value. When running in a container, this parameter should point to persistent storage.</p>
VRefreshInterval	<p>Defines the visibility refresh interval for CyberMgr. The default value is 10 seconds. 0 seconds turns off the visibility.</p> <p>By default, this is turned off. To turn it on, set it to 10 seconds or higher.</p>
RpcSynchIntervalHA	<p>Defines the time range that is out of sync between RpcServer (aka CyberMgr) and any other client app (like CyberResp cfinq, cfsend/cfrecv, and so on) on another HA cluster machine. The rpc call fails with an error "rpc security time failure" when the time is out of sync between the RPC client and the CyberMgr machines.</p> <p>The supported range is 30 seconds to 86400 seconds. The default value is 60 seconds.</p> <p>RpcSynchInterval: 60 { # of sec, 0 turn off</p>

Parameter	Description
	time synch }
RpcMaxWaitConnectTimeHA	<p>RpcMaxWaitConnectTimeHA: 1 { # of sec, 0 unlimited wait connect }</p> <p>This parameter controls for how long any RPC client waits to connect to CyberMgr.</p> <div style="border: 1px solid #ccc; padding: 10px; background-color: #f9f9f9;"> <p>Note: Each RPC call consists of two parts: connect to server; if successful, then issue the actual call. RpcMaxMaitMonnectTimeHA controls only the first part, connect to CyberMgr. The actual calls have their own Timeout value, which we have set based on the importance of the call. RPC calls to get TransactionNumber and log transfer record have a timeout of 60 seconds. All other RPC calls have a timeout of 10 seconds.</p> </div>

Password Rules for Responder Profile

```
# Password Rules for Responder Profile
PasswordRuleChecking:      N                { Y, N }
PasswordRequireUpperAndLower: N            { Y, N }
PasswordMinLength:        8                { 3 - 64 }
PasswordMinUnique:        3                { 0 - PasswordMinLength }
PasswordMinLetters:       3                { 0 - PasswordMinLength }
PasswordMinNumber:        0                { 0 - PasswordMinLength }
PasswordMinSpecial:       0                { 0 - PasswordMinLength }
```

Parameter	Description
PasswordRuleChecking	<p>Defines whether to enable password rule checking for the remote password of the responder profile.</p> <p>The valid values are:</p> <p>Y: password rule checking is enabled.</p>

Parameter	Description
	N: password rule checking is disabled. N is the default value.
PasswordRequireUpperAndLower	<p>Defines whether the remote password of the responder profile must include uppercase and lowercase characters.</p> <p>The valid values are:</p> <p>Y: requires uppercase and lowercase characters in the remote password.</p> <p>N: does not require uppercase and lowercase characters in the remote password. N is the default value.</p>
PasswordMinLength	Defines the length of the remote password of the responder profile. The password length can be between 3 - 64 characters. The default value is 8 characters.
PasswordMinUnique	Defines the minimum number of unique characters in the remote password of the responder profile. Valid values are from 0 - PasswordMinLength. The default value is 3 characters.
PasswordMinNumber	Defines the minimum number of numeric characters in the remote password of the responder profile. Valid values are from 0 - PasswordMinLength. The default value is 0.
PasswordMinLetters	Defines the minimum number of letters (A-Z and a-z) in the remote password of the responder profile. Valid values are from 0 - PasswordMinLength. The default value is 3.
PasswordMinSpecial	Defines the minimum number of special characters in the remote password of the responder profile. Valid values are from 0 - PasswordMinLength. The default value is 0.

Parameter	Description
	<p>Note: If you are upgrading from a version prior to version 8.0.0 into the same \$CFROOT directory, the password validation parameters are added automatically.</p>

Parse Commands

```
# Parse Commands
protect_cctransfer:      exec           { none|reject|exec }
protect_cfdir:          reject          { none|reject|exec }
protect_fusutil:        reject          { none|reject|exec }
protect_receivdir:      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 }
```

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> <table border="1"> <thead> <tr> <th>Parameter Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>none</td> <td>No additional parsing is performed when Platform Server executes a system command.</td> </tr> <tr> <td>reject</td> <td>If a command to be executed includes any of the characters defined by the rejectcmdcharacters parameter, the</td> </tr> </tbody> </table>	Parameter Value	Description	none	No additional parsing is performed when Platform Server executes a system command.	reject	If a command to be executed includes any of the characters defined by the rejectcmdcharacters parameter, the
Parameter Value	Description						
none	No additional parsing is performed when Platform Server executes a system command.						
reject	If a command to be executed includes any of the characters defined by the rejectcmdcharacters parameter, the						

Parameter	Description								
	<table border="1"> <thead> <tr> <th>Parameter Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td></td> <td>command terminates with an error.</td> </tr> <tr> <td>exec</td> <td> <p>The command to be executed calls the <code>exec</code> function. This call does not allow multiple commands to be executed in a single command string.</p> <p>The <code>exec</code> option supports up to 100 command line parameters.</p> </td> </tr> </tbody> </table> <p>The default value is <code>exec</code>.</p>	Parameter Value	Description		command terminates with an error.	exec	<p>The command to be executed calls the <code>exec</code> function. This call does not allow multiple commands to be executed in a single command string.</p> <p>The <code>exec</code> option supports up to 100 command line parameters.</p>		
Parameter Value	Description								
	command terminates with an error.								
exec	<p>The command to be executed calls the <code>exec</code> function. This call does not allow multiple commands to be executed in a single command string.</p> <p>The <code>exec</code> option supports up to 100 command line parameters.</p>								
<code>protect_cfdir</code>	<p>Defines the processing performed when a <code>cfdir</code> request is received. A <code>cfdir</code> request prompts Platform Server to return a directory list or the status of a file or directory.</p> <p>The valid values are:</p> <table border="1"> <thead> <tr> <th>Parameter Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>none</code></td> <td>No additional parsing is performed when Platform Server executes a system command.</td> </tr> <tr> <td><code>reject</code></td> <td>If a command to be executed includes any of the characters defined by the <code>rejectcmdcharacters</code> parameter, the command terminates with an error.</td> </tr> <tr> <td><code>exec</code></td> <td>The command to be executed calls the <code>exec</code> function. This call does not allow multiple</td> </tr> </tbody> </table>	Parameter Value	Description	<code>none</code>	No additional parsing is performed when Platform Server executes a system command.	<code>reject</code>	If a command to be executed includes any of the characters defined by the <code>rejectcmdcharacters</code> parameter, the command terminates with an error.	<code>exec</code>	The command to be executed calls the <code>exec</code> function. This call does not allow multiple
Parameter Value	Description								
<code>none</code>	No additional parsing is performed when Platform Server executes a system command.								
<code>reject</code>	If a command to be executed includes any of the characters defined by the <code>rejectcmdcharacters</code> parameter, the command terminates with an error.								
<code>exec</code>	The command to be executed calls the <code>exec</code> function. This call does not allow multiple								

Parameter	Description						
	<table border="1"> <thead> <tr> <th>Parameter Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td></td> <td> <p>commands to be executed in a single command string.</p> <p>The exec option supports up to 100 command line parameters.</p> </td> </tr> </tbody> </table> <p>The default value is reject.</p>	Parameter Value	Description		<p>commands to be executed in a single command string.</p> <p>The exec option supports up to 100 command line parameters.</p>		
Parameter Value	Description						
	<p>commands to be executed in a single command string.</p> <p>The exec option supports up to 100 command line parameters.</p>						
protect_cfgpostproc	<p>Defines the processing performed when a configured Postprocessing command is executed.</p> <p>The valid values are:</p> <table border="1"> <thead> <tr> <th>Parameter Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>none</td> <td> <p>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> </td> </tr> <tr> <td>token</td> <td> <p>If a PPA or substitutable postprocessing token includes any of the characters defined by the rejectcmdcharacters parameter, the command terminates with an error.</p> <p>The following PPA or configured</p> </td> </tr> </tbody> </table>	Parameter Value	Description	none	<p>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>	token	<p>If a PPA or substitutable postprocessing token includes any of the characters defined by the rejectcmdcharacters parameter, the command terminates with an error.</p> <p>The following PPA or configured</p>
Parameter Value	Description						
none	<p>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>						
token	<p>If a PPA or substitutable postprocessing token includes any of the characters defined by the rejectcmdcharacters parameter, the command terminates with an error.</p> <p>The following PPA or configured</p>						

Parameter	Description								
	<table border="1"> <thead> <tr> <th>Parameter Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td></td> <td> <p>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 </td> </tr> <tr> <td><code>exec</code></td> <td> <p>The command to be executed calls the <code>exec</code> 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 <code>exec</code> call to complete. The <code>exec</code> option supports up to 100 command line parameters.</p> </td> </tr> <tr> <td></td> <td> <p>The default value is <code>token</code>.</p> </td> </tr> </tbody> </table>	Parameter Value	Description		<p>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 	<code>exec</code>	<p>The command to be executed calls the <code>exec</code> 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 <code>exec</code> call to complete. The <code>exec</code> option supports up to 100 command line parameters.</p>		<p>The default value is <code>token</code>.</p>
Parameter Value	Description								
	<p>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 								
<code>exec</code>	<p>The command to be executed calls the <code>exec</code> 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 <code>exec</code> call to complete. The <code>exec</code> option supports up to 100 command line parameters.</p>								
	<p>The default value is <code>token</code>.</p>								
<code>protect_fusutil</code>	<p>Defines the processing performed when a <code>fusutil</code> request is received. A <code>fusutil</code> 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>								

Parameter	Description								
	<table border="1"> <thead> <tr> <th>Parameter Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>none</td> <td>No additional parsing is performed when Platform Server executes a system command.</td> </tr> <tr> <td>reject</td> <td>If a command to be executed includes any of the characters defined by the <code>rejectcmdcharacters</code> parameter, the command terminates with an error.</td> </tr> <tr> <td>exec</td> <td>The command to be executed calls the <code>exec</code> function. This call does not allow multiple commands to be executed in a single command string. The <code>exec</code> option supports up to 100 command line parameters.</td> </tr> </tbody> </table> <p>The default value is <code>reject</code>.</p>	Parameter Value	Description	none	No additional parsing is performed when Platform Server executes a system command.	reject	If a command to be executed includes any of the characters defined by the <code>rejectcmdcharacters</code> parameter, the command terminates with an error.	exec	The command to be executed calls the <code>exec</code> function. This call does not allow multiple commands to be executed in a single command string. The <code>exec</code> option supports up to 100 command line parameters.
Parameter Value	Description								
none	No additional parsing is performed when Platform Server executes a system command.								
reject	If a command to be executed includes any of the characters defined by the <code>rejectcmdcharacters</code> parameter, the command terminates with an error.								
exec	The command to be executed calls the <code>exec</code> function. This call does not allow multiple commands to be executed in a single command string. The <code>exec</code> option supports up to 100 command line parameters.								
<code>protect_ppa</code>	<p>Defines the processing performed when a PPA command is executed.</p> <p>The valid values are:</p> <table border="1"> <thead> <tr> <th>Parameter Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>none</td> <td>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</td> </tr> </tbody> </table>	Parameter Value	Description	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				
Parameter Value	Description								
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								

Parameter	Description								
	<table border="1"> <thead> <tr> <th>Parameter Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td></td> <td>prompt Platform Server to wait for the system call to complete.</td> </tr> <tr> <td>token</td> <td> <p>If a PPA or configured postprocessing token includes any of the characters defined by the <code>rejectcmdcharacters</code> 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 </td> </tr> <tr> <td>exec</td> <td> <p>The command to be executed calls the <code>exec</code> 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 <code>exec</code> call to complete. The <code>exec</code> option supports up to 100 command line parameters.</p> </td> </tr> </tbody> </table>	Parameter Value	Description		prompt Platform Server to wait for the system call to complete.	token	<p>If a PPA or configured postprocessing token includes any of the characters defined by the <code>rejectcmdcharacters</code> 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 	exec	<p>The command to be executed calls the <code>exec</code> 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 <code>exec</code> call to complete. The <code>exec</code> option supports up to 100 command line parameters.</p>
Parameter Value	Description								
	prompt Platform Server to wait for the system call to complete.								
token	<p>If a PPA or configured postprocessing token includes any of the characters defined by the <code>rejectcmdcharacters</code> 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 								
exec	<p>The command to be executed calls the <code>exec</code> 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 <code>exec</code> call to complete. The <code>exec</code> option supports up to 100 command line parameters.</p>								
	The default value is token.								
protect_rcmd	<p>Defines the processing performed when receiving a command other than <code>cfdir</code> or <code>fusutil</code>.</p> <p>The valid values are:</p>								

Parameter	Description								
	<table border="1"> <thead> <tr> <th>Parameter Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>none</td> <td>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.</td> </tr> <tr> <td>reject</td> <td>If a command to be executed includes any of the characters defined by the <code>rejectcmdcharacters</code> parameter, the command terminates with an error. Note: You can use the ampersand sign (&) to tell Platform Server to execute the command in background.</td> </tr> <tr> <td>exec</td> <td>The command to be executed calls the <code>exec</code> 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 <code>exec</code> call to complete. The <code>exec</code> option supports up to 100 command line parameters.</td> </tr> </tbody> </table> <p>The default value is <code>reject</code>.</p>	Parameter Value	Description	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.	reject	If a command to be executed includes any of the characters defined by the <code>rejectcmdcharacters</code> parameter, the command terminates with an error. Note: You can use the ampersand sign (&) to tell Platform Server to execute the command in background.	exec	The command to be executed calls the <code>exec</code> 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 <code>exec</code> call to complete. The <code>exec</code> option supports up to 100 command line parameters.
Parameter Value	Description								
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.								
reject	If a command to be executed includes any of the characters defined by the <code>rejectcmdcharacters</code> parameter, the command terminates with an error. Note: You can use the ampersand sign (&) to tell Platform Server to execute the command in background.								
exec	The command to be executed calls the <code>exec</code> 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 <code>exec</code> call to complete. The <code>exec</code> option supports up to 100 command line parameters.								
<code>protect_receivedir</code>	Defines the processing performed when executing a receive								

Parameter	Description								
	<p>directory and Platform Server issues a "cfscmd trytpe:c..." command to request a directory list.</p> <p>The valid values are:</p> <table border="1"> <thead> <tr> <th>Parameter Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>none</td> <td>No additional parsing is performed when Platform Server executes a system command.</td> </tr> <tr> <td>reject</td> <td>If a command to be executed includes any of the characters defined by the <code>rejectcmdcharacters</code> parameter, the command terminates with an error.</td> </tr> <tr> <td>exec</td> <td>The command to be executed calls the <code>exec</code> function. This call does not allow multiple commands to be executed in a single command string.</td> </tr> </tbody> </table> <p>The default value is <code>exec</code>.</p>	Parameter Value	Description	none	No additional parsing is performed when Platform Server executes a system command.	reject	If a command to be executed includes any of the characters defined by the <code>rejectcmdcharacters</code> parameter, the command terminates with an error.	exec	The command to be executed calls the <code>exec</code> function. This call does not allow multiple commands to be executed in a single command string.
Parameter Value	Description								
none	No additional parsing is performed when Platform Server executes a system command.								
reject	If a command to be executed includes any of the characters defined by the <code>rejectcmdcharacters</code> parameter, the command terminates with an error.								
exec	The command to be executed calls the <code>exec</code> function. This call does not allow multiple commands to be executed in a single command string.								
<code>rejectcmdcharacters</code>	<p>Defines the characters that are validated when <code>reject</code> or <code>token</code> 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 <code>;& </code></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 Platform Servers.
- [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.
- [Preprocessing](#)
You can use this feature to use preprocessing actions for Platform Server transfer clients.
- [Configured Post Processing](#)
You can use this feature to configure postprocessing 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 and cfdos2unix.exe Utilities](#)

You can use this feature (these utilities) to convert a text file from UNIX format to DOS format from DOS format to UNIX format.

- [Active Transfers Support](#)

You can see active transfers and cancel any active transfer.

i **Note:** 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 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 terminates prematurely, the initiator tries three times to restart the transfer from this checkpoint. Because the `RetryInterval` parameter is set to 2 minutes, The Platform Server initiator waits for 2 minutes between each retry attempt.

Conversion Tables/Custom Code Conversion

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

i Note: 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.)

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

00010203372D2E2F16050A0B0C0D0E0F	ASCII-EBCDIC portion of the translation table
101112133C3D322618193F27221D351F	
405A7F7B5B6C507D4D5D5C4E6B604B61	
F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F	
7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6	
D7D8D9E2E3E4E5E6E7E8E9BAE0BBB06D	
79818283848586878889919293949596	
979899A2A3A4A5A6A7A8A9C04FD0A107	
9F000000000000000000000000000000	
00000000000000000000000000000000	
41AA4AB100B26AB5BDB49A8A5FCAAFBC	
908FEAFABEA0B6B39DDA9B8BB7B8B9AB	
6465626663679E687471727378757677	
AC69EDEEEBEFECBF80FDFEFBFCADAE59	
4445424643479C485451525358555657	
8C49CDCECBCFCCE170DDDEDBDC8D8EDF	
002E2E2E2E2E2E2E2E2E2E2E2E2E2E	EBCDIC-ASCII portion of the translation table
2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E	
2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E	
2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E	
2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E	
20A0E2E4E0E1E3E5E7F1A22E3C282B7C	
26E9EAE8E8E8E8E8E8E8E8E8E8E8E8	
2D2FC2C4C0C1C3C5C7D1A62C255F3E3F	
F8C9CACBC8CDCECFCC603A2340273D22	
D8616263646566676869ABBBF0FDFEB1	
B06A6B6C6D6E6F707172AABAE6B8C680	
B57E737475767778797AA1BFD0DDDEAE	
5EA3A5B7A9A7B6BCBDBE5B5DAFA8B4D7	
7B414243444546474849ADF4F6F2F3F5	
7D4A4B4C4D4E4F505152B9FBFCF9FAFF	
5CF7535455565758595AB2D4D6D2D3D5	
30313233343536373839B3DEBDC9DA2E	

To activate conversion tables, you have to turn the `ASCII_to_EBCDIC` parameter on. For more information, see `ASCII_to_EBCDIC` parameter in [Optional Transfer Properties](#).

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.

The first 16 lines in this file are used when sending data to a partner, that is, converting ASCII data to EBCDIC. The last 16 lines are used when receiving data from a partner, that is, converting EBCDIC to ASCII. ASCII to EBCDIC translation is typically performed on the ASCII systems (Windows and UNIX) and not on the EBCDIC systems (z/OS and IBM i). You can specify two conversion tables: one on the local side (LCT), and one on the remote side (RCT). If you want to perform translation on one side of the transfer and not on the other side, specify a translation table of `None`.

Example: `LCT:CustomTable RCT:None` or `LCT:None RCT:CustomTable`

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.

- For UNIX, the directories are searched in the following order: the directory where CyberResp is running, 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 MFT Platform Server working directory.

i Note: 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 16 lines of the conversion table is used for ASCII to EBCDIC conversion. For a File Receive, the bottom 16 lines of the conversion table is used for EBCDIC to ASCII conversion. 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 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.

i Note: 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 do not support 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](#).

i Note: 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]
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:   NA1DEVMFTYZ01
Key Expiration: Unknown
Version:
Ftms32.DLL, Version 8.0 (Build 28 HF-007 UNICODE)
FtmsDni.DLL, Version 8.0 (Build 28 HF-007 UNICODE)
FtmsTcpS.DLL, Version 8.0 (Build 28 HF-007 UNICODE)
FtmsVer.DLL, Version 8.0 (Build 28 HF-007 UNICODE)
FusionMs.DLL, Version 8.0 (Build 28 HF-007 UNICODE)
HoLib.DLL, Version 8.0 (Build 28 HF-007 UNICODE)
HOTrace.DLL, Version 8.0 (Build 28 HF-007 UNICODE)
SMTPD11.DLL, Version 8.0 (Build 28 HF-007)
FtmsMgr.EXE, Version 8.0 (Build 28 HF-007 UNICODE)
FtmsCmd.EXE, Version 8.0 (Build 28 HF-007 UNICODE)
FtmsMon.EXE, Version 8.0 (Build 28 HF-007 UNICODE)
FtmsSvr.EXE, Version 8.0 (Build 28 HF-007 UNICODE)
FusionVer.EXE, Version 8.0 (Build 28 HF-007 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]
```

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.

Command Name	Description
<code>rdir</code> <code>renamedir</code>	Renames a directory
<code>ddir</code> <code>deletedir</code>	Deletes a directory
<code>rmdir</code> <code>removedir</code>	Removes a directory
<code>mmdir</code> <code>movedir</code>	Moves a directory
<code>mkdir</code> <code>makedir</code>	To make or create a directory
<code>r</code> <code>rename</code>	Renames an existing file
<code>d</code> <code>delete</code>	Deletes a file
<code>m</code> <code>move</code>	Moves the location of a file
<code>e</code> <code>exist</code>	Verifies the existence of files



Note: The maximum length of the `fusutil` command is 1024 characters.

See the following postprocessing 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_filenamenew_filename
```

```
Post_Action4: F,R,COMMAND,fusutil R old_filenamenew_filename
```

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

```
Post_Action3: S,L,COMMAND,fusutil MKDIR directoryname
```

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

```
Post_Action4: S,R,COMMAND,fusutil MVDIR directorynamenew_directoryname
```

```
Post_Action5: F,R,COMMAND,fusutil RDIR old_directorynamenew_directoryname
```

i Note:

- When processing the `EXIST` option, the code also checks if the file is available for use. On UNIX, this is not checked because there is no standard file locking mechanism on UNIX.
- 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 Postprocessing

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

- i Note:** For information on defining postprocessing actions for individual 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 SEND, RECEIVE, or BOTH.
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.
IPADDR	Defines the IP address of the machine that communicates with TIBCO® Managed File Transfer Platform Server for UNIX.
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.

Parameter	Description
	For responder requests, the program scans the list of nodes for matches on the IP address. These entries are then matched against the value specified in this NODE parameter.

i **Note:** If a parameter is not defined, it is considered as a match. If all parameters match, the command will be executed.

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)
&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 Postprocessing Command Examples

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

See the following two configured postprocessing 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.

 **Note:** 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. This is feature only in Responder. 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> <p>Note: DEFAULT indicates a match with any user.</p>
NODE	<p>Defines the name of the node from which the transfer request is initiated.</p> <p>The valid values are <i>nodename</i> or DEFAULT.</p> <p>Note: DEFAULT indicates a match with any node.</p>
IPADDR	<p>Defines the IP address of the server which initiated the transfer request.</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>
FILE	<p>Defines the fully qualified name you want to use instead of the alias file.</p>

Parameter	Description
ALIAS	Defines the file name requested by the initiator.
ALLOW	Determines whether the user who initiates the transfer is allowed to define the actual file name if no match is found.

Note: ALLOW and FILE/ALIAS are mutually exclusive.

The valid values are:

- NO: the user is not allowed to define the actual file name.
- YES: the user is allowed to define the actual file name

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)
%JDATEC	Julian date (CCYYDDD)
%GDATE	Gregorian date (YYMMDD)
%GDATEC	Gregorian date (CCYYMMDD)
%TIMET	Time (HHMMSST)

Substitutable Parameters	Description
%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/abc123.20200718.1601029` 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, 2020, the file can be created as `report.20200718.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. From version 8.1.0, this utility provides improved services such as the capability to reset CyberMgr settings.

cfinq Parameters

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



Note:

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

Parameter (Alternate Specification)	Description
DAYS	Defines the number of days to search.
	Note: DAYS must not exceed 1826 (5 years).

Parameter (Alternate Specification)	Description
	<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 USERDATA.</p> <p>Based on the DESCRIPTION parameter, the <code>cfinq</code> 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 <code>yyyymmdd</code>.</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>Note: To use ENDDATE, you must define STARTDATE or DAYS.</p>
ENDTIME (ETIME)	<p>Defines the end time in the 24 hour format of <code>hhmmss</code>.</p> <p>The default value is 235959.</p> <p>If STIME is not defined, the <code>cfinq</code> 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

Parameter (Alternate Specification)	Description
	<ul style="list-style-type: none"> • S: successful • Default: successful or unsuccessful
LOCALFILE (LF)	Defines the local file name.
LOCALUSER (LUSER)	Defines the local user name (user ID). <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Note: If you specify a user name other than your own, you must have the appropriate security authorization.</p> </div>
LOCTRANSNUM (LTRN)	Defines the unique local transaction number of the transfer. <p>Based on the LOCTRANSNUM parameter, the <code>cfinq</code> 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)	Defines the maximum number of requests that can be returned. <p>The default number is 500. The valid values are from 1 to 100,000.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Note: The oldest transfer information is gathered first. If the total number of records exceeds the MAXXFER value, the information close to the SDATE is not included in the transaction list.</p> </div>
PROCESS (PRO)	Defines the process name.
REMHOST (RHOST)	Defines the remote system name. <p>This can be a node name, host name or IP address</p>
REMTRANSNUM (RTRN)	Defines the remote transaction number.

Parameter (Alternate Specification)	Description
STARTDATE (SDATE)	<p>Defines the start date of the search in the format <code>yyymmdd</code>.</p> <ul style="list-style-type: none"> • <code>SDATE = TOD</code> or <code>SDATE = TODAY</code> means today. • <code>SDATE = YES</code> or <code>SDATE = YESTERDAY</code> means yesterday. <p>The default value is <code>TODAY</code>.</p>
STARTTIME (STIME)	<p>Defines the start time in 24 hour format of <code>hhmmss</code>.</p> <p>The default is <code>000000</code>. If <code>ETIME</code> is not defined, the <code>cfinq</code> utility searches for the transaction only between the <code>STIME - 235959</code> period.</p>
TEMPERROR (TMPERR)	<p>Defines whether to display transfers that retried with temporary errors that are in the audit file.</p> <p>This parameter applies regardless of whether the <code>Print</code> parameter is defined or not. The valid values are <code>Yes</code> or <code>No</code>. The default value is <code>No</code>.</p>

Active Transfers Support

From version 8.1.0, you can monitor active transfers. Command Center **Active Transfers** and `cfinq` can issue RPC calls to CyberMgr to query all active transfers. Command Center **Active Transfers** also can issue a RPC call to cancel an active transfer on demand.

All active transfers send updates to CyberMgr at preconfigured intervals. Updates are sent via RPC calls, packed as TLV data containing all active transfer information (most importantly, the byte count). The update interval is configured in the `config.txt` file by the following parameter:

`VRefreshInterval: 10 { number of seconds, 0} 0` turns off visibility.

Modifying or Turning On/Off Visibility Support

To modify or turn on/off visibility support on the CyberMgr side, use `cfinq mgr=u call`.

- If visibility is turned off on the CyberMgr side, then no active transfers are collected

on the \$CFR00T or on the HA cluster.

- If visibility is turned off only on one HA cluster member (only on one of the Platform Server for UNIX from the HA cluster, but not on the primary CyberMgr instance), then only that specific Platform Server does not report active transfers.

It is a two-step process to modify `VRefreshInterval` on the CyberMgr active process side:

1. Open the `config.txt` file and adjust the value.
2. Issue `cfinq mgr=u` RPC call to notify the active CyberMgr process to adjust its own value.

To issue this call a user must be a member of the 'cfadmin' group. Every time `VRefreshInterval` (or any other setting) is changed on the active CyberMgr, the change is recorded in the PCI log/admin file.

Viewing Active Transfers

You can view active transfers in two ways:

1. By viewing the Command Center View Active Transfers page.
2. By running `cfinq mgr=a` on TIBCO MFT Platform Server for UNIX.

With `cfinq`, you can use "r" to refresh the display, without existing `cfinq`. Transfers state cannot be refreshed more frequently than the configured `VRefreshInterval`. You can also keep track of the progress of an individual transfer by clicking on it till the transfer is reported as completed. After this, the transfer is removed from the active queue and into the `Log.txt` history file which is collected by CC collector or can be viewed by the regular `cfinq` request.

Restrictions for Viewing Active Transfers

- Only members of `root` or `cfadmin` group can see all active transfers.
- If "display active transfers" request is issued via a Command Center request, then the Command Center option for this **CC Node Definition** also is checked. `CommandCenter` should be either `ALL` or `Audit`.

Canceling Active Transfers

You can cancel an active transfer only by canceling it from the Command Center side. `cfinq` does not have such an option. To cancel any active transfer, you must be either a root or a member of the `cfadmin` group. **CC Node Definition** is checked as well. CommandCenter must contain `Alter`. `Alter` is not included in `ALL`. For example,

This `cfnode.cfg` setup allows you to cancel any active transfer on demand:

```
CommandCenter = ALL, ALTER
CommandCenter = AUDIT, ALTER
```

This `cfnode.cfg` setup does not allow you to cancel any active transfer:

```
CommandCenter = ALL
```

All cancellation requests are reported in the PCI log/message file.

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, 2020 at 9:01 am and ending at 3 pm, with a maximum of 1000 records listed.

```
cfinq sdate:20200909 days:20 stime:090100 etime:150000 luser:JohnDoe
EXC:S max:1000
```

i Note: Use either an equal sign or a colon to separate the parameter from the value.

You can also use the `cfinq` menu to display transfer inquiry parameters. 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 [r] [enter] TO REFRESH THE 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 127.0.0.1:46464 /home/a.txt
2 I113500001 Success 127.0.0.1:46464 /home/remotefile
3 I113500002 Success 127.0.0.1:46464 /home/localfile
4 I113500003 Success 127.0.0.1:46464 /home/a.txt
5 I113500004 Success 127.0.0.1:46464 /home/tmp/CG.DAT
6 I113500005 Success 127.0.0.1:46464 /home/tmp/CLEAN.EXE
7 I113500006 Success 127.0.0.1:46464 /home/tmp/RUN.BAK
8 I113500007 Success 127.0.0.1:46464 /home/tmp/HOOK.REG
9 I113500008 Success 127.0.0.1:46464 /home/tmp/RUN.BAK
10 R113500009 Success 127.0.0.1:46464 /home/tmp/WAKE.EXE
11 R113500002 Success 127.0.0.1:46464 /home/tmp/EXPRESS.INIdeta
==>
```

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..... 8.1 build 4. Maint Build 4
Priority..... N/A
Local Transaction Number... I113500002
Remote Transaction Number... R113500003
Transfer Start Time..... 160701
Transfer Start Date..... 20210113
Transfer End Time..... 160705
Transfer End Date..... 20210113
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
CRC..... N/A
TLSProtocol..... N/A
TLSCipher..... N/A
RetainFileStamp..... N
LocalHostName..... N/A
*****
*****

==>

```

```
CRC..... N/A
TLSProtocol..... N/A
TLSCipher..... N/A
RetainFileStamp..... N
LocalHostName..... N/A
```

```
*****
*****
```

```
==>
```

Changing CyberMgr Settings

From version 8.1.0, the `cfing` utility can reset and reconfigure several important settings in the active CyberMgr process. For example, CyberMgr **TraceLevel**, **VisibilityInterval**, and **SemaphoreTimeOut** interval can be reset by issuing the appropriate `cfing` request. CyberMgr RPC daemon does not need to be recycled.

Before you begin

To reset CyberMgr setting, as a `cfing` user, you must be a root user or a member of the `cfadmin` group.

Procedure

1. Open the `config.txt` file on the target CyberMgr machine in a vi editor and adjust or reset the appropriate setting as required. For example, `VRefreshInterval: 20` (instead of the default 0),
2. Run `cfing` with `mgr` option.

`cfing mgr=u` : will cause local CyberMgr daemon to re-read configuration values.

`cfing mgrp=u` : will cause primary CyberMgr daemon to re-read configuration values.

`cfing mgrs=u` : will cause secondary CyberMgr daemon to re-read configuration values.



Note: any `cfing mgr` command should be run with only one option at a time.

As the result, the requested setting is reset in the active CyberMgr daemon. The reset request is logged in the `log/admin` file to comply with PCI requirements.

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.



Note:

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 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, Localc File=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, PPA2
Source=N/A, PPA2Status=N/A, PPA2Data=N/A,
PPA2ReturnCode=N/A, PPA3Action=N/A, PPA3Source=N/A, PPA3Status=N/A, PPA
3Data=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, LocalHostName=NYLinuxHost
```

From version 8.1.0, only `cfroot` (a user under whose account the product is installed) and members of the `cfadmin` group have “write” access to the `Log.txt` file (664) and `log` folder (775). Other users have only “read” access, therefore they cannot modify information in this file.

However, file transfers initiated by non-privileged users still have to be logged via “RPC logging requests”. A new daemon, `CyberMgr` (an RPC Server) has been added to version 8.1.0. All other applications send messages to `CyberMgr`, and `CyberMgr` is solely responsible to perform the requested updates/changes. `CyberMgr` runs under the same account as `CyberResp`.

Note: Two log files, namely `admin` and `message` have been added. All important PSU calls are logged to one of these log files.

Note: `CyberMgr.access` is used to communicate between `CyberMgr` and RPC clients. This file should be the same on `CyberMgr` and all PSU instances in the cluster.

Note: If the connection to the local `CyberMgr` takes a long time to complete, you can set environment variable `RPCMaxWaitConnectTimeLocal`. This environment variable defines the maximum amount of time in seconds that rpc client will wait for a connection to the Local `CyberMgr` RPC Server. If the connection time exceeds the defined number of seconds, the RPC request will not be completed and the transfer might fail.

Log Admin File

User actions which make changes in the existing environment/ setup are logged in `$CFROOT/log/admin/Admin.txt.date` :

The following common events are logged:

- All `cfnode`, `cfprofile`, `cfprofile`, `cc_node`, `cc_profile`, `Command Center_rprofile` requests.
- `cfinq mgr=Active`, `Command Center` requests, to view or cancel active transfers.
- `cfinq mgr=Update`, to update active `CyberMgr` settings.
- `cfstart / cfstop`

The following action types are logged. (This is not a complete list of actions.)

- **DISPLAY:** No message details are provided. This event can be shown multiple times, because a "read" call is often made on other actions as well. For example, after you **DELETE** or **UPDATE**, you still can call `read` to display updated information.
- **ADD/UPDATE:** Shows a detailed change which was made by a user.
- **DELETE:** Shows a detailed change which was made by a user.
- **Reconfig:** Shows a detailed change which was made by a user. For example, `cfinq mgr=Update`, produces a **Reconfig** record in the `Admin.txt` file.

Log Message File

Transfer-related events or other non-admin request that are made are logged in `$CFROOT/log/message/Message.txt.date`

The following common events are logged:

- All `cfsend`, `cfrecv`, `CyberResp` file transfer requests
- `Command Center XFER` transfer request
- `Command Center collector` request to view history (one summary record per collector request)
- `fusping` or `Command Center "ping PS Server"` request

The format of a Log Admin File and Log Message File is as follows:

Time	Level	User	Request Type	Action Type	Message
11:27:31	INFO	tom	CFRPROF	ADD/UPDATE	Writing Responder Profile [cfcc5]. Details: [Local User=tom, Remote User=test_tom]
14:47:37	ERROR	mary	CCNODE	DELETE	User [mary] is not a Member of Group [cfadmin]

- Time: Time of the event
- Level : INFO, WARN, ERROR
- User: Name of the user
- Request Type (most commonly used) : TRANSFER, CFNODE, CFPROF, CFRPROF, CFPING, CFINQ, CCNODE, CCPROF, CCRPROF, PING, CCINQ, MGR, MSG
- Action Type (most commonly used) : DISPLAY, ADD/UPDATE, DELETE, Collector, Active, Reconfig
- Message: Information about the event

Customized Event Logging App

A new app or utility, msgmgr, has been included in the bin folder. This app can be utilized based on your needs. For example, you can call it when you start/stop DNI templates and would like messages to be logged into the admin or message file.

```
usage: msgmgr [ADM|MSG] "text" [action]
```

- ADM - send text to the admin log file.
- MSG - send text to the message log file.
- "text" - text to send. If it includes more than one word, the text must be in quotation marks.
- action - optional: one word action type to be recorded.

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](#).

i Note: 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 `$CFROOT/config` directory.

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

Parameter Name	Description
USERID	Identifies the local user or DEFAULT for all users. Note: You must specify either the USERID or NODE/IPADDR parameter. And you can specify both USERID and NODE/IPADDR.
NODE	Identifies the node name or DEFAULT for all nodes. Note: 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.
IPADDR	Identifies the ipaddr or DEFAULT for all nodes.

Parameter Name	Description
	<p>Note: 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>
DESCRIPTION	Defines a description as a user comment for this access control, at a maximum, the length of the description is 32 bytes.
SEND_DIR	<p>Defines the default send directory.</p> <p>This parameter has no default value.</p>
RECEIVE_DIR	<p>Defines the default receive directory.</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>
SEND_OPTION	<p>Defines the options for sending files.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • ROOT <p>If you have specified a directory in a transfer command, then the directory is appended to the directory defined by the SEND_DIR parameter.</p> • FORCE <p>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.</p> • ALLOW

Parameter Name	Description
	<p>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 <code>SEND_DIR</code> parameter.</p> <ul style="list-style-type: none"> • REJECT <p>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 <code>SEND_DIR</code> parameter.</p> <ul style="list-style-type: none"> • NEVER <p>The user defined with the <code>NODE</code> or <code>USERID</code> parameter can never send files.</p> <ul style="list-style-type: none"> • USE <p>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 started CyberResp (the <code>\$PWD</code> environment variable for CyberResp) is used.</p>
	<p>Note: If <code>SEND_OPTION</code> is not specified, <code>USE</code> is the default setting.</p>
RECEIVE_OPTION	<p>Defines the options for receiving files.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • ROOT <p>If you have specified a directory in a transfer command, then the directory is appended to the directory defined by the <code>RECEIVE_DIR</code> parameter.</p> <ul style="list-style-type: none"> • FORCE <p>If you have specified a directory in a transfer command, then the directory is changed to the directory defined by the <code>RECEIVE_DIR</code> parameter. The directory name defined in the request is ignored. The file name is appended directly to the directory defined by the <code>RECEIVE_DIR</code> parameter.</p>

Parameter Name	Description
	<ul style="list-style-type: none"> • 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 <code>RECEIVE_DIR</code> 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 <code>RECEIVE_DIR</code> parameter. • NEVER The user defined with the <code>NODE</code> or <code>USERID</code> 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 started CyberResp (the <code>\$PWD</code> environment variable for CyberResp) is used. <p>Note: If <code>RECEIVE_OPTION</code> is not specified, <code>USE</code> is the default setting.</p>
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 <code>COMMAND_DIR</code> parameter. • NEVER The user defined with the <code>NODE</code> or <code>USERID</code> parameter can never use commands. • USE

Parameter Name	Description
	<p>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 started CyberResp (the <code>\$PWD</code> environment variable for CyberResp) is used.</p> <p>Note: If <code>COMMAND_OPTION</code> is not specified, <code>USE</code> is the default setting.</p>
<code>SUBMIT_OPTION</code>	<p>Defines the options for submitting jobs.</p> <p>The valid values are:</p> <ul style="list-style-type: none"> • <code>ALLOW</code> <p>The user defined in a transfer command can submit a job.</p> • <code>NEVER</code> <p>The user defined with the <code>NODE</code> or <code>USERID</code> parameter can never submit a job.</p>

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 file defined in the `LocalFileName` parameter is put into the directory defined by the `RECEIVE_DIR` parameter.

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

```
/test/2021/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/2021/accounting/tax.data
```

The actual file name is expected to be:

```
/a/b/test/2021/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 <https://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, use `openssl` app from `$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.

### Before you begin

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

### Procedure

1. On the command line, navigate to the `$CFROOT/util` 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 for incoming SSL or Tunnel transfers 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.

**i Note:** The authorization configuration file checking is in addition to Trusted Authority file SSL checking. This checking is performed only if a certificate is accepted by SSL.

**i Note:** 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.



**Note:** 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 line.
- 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.

**Note:** 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.<br><br>Generic entries are supported. |
| /OU       | Defines the organization unit defined in a certificate. This is also known as the department.<br><br>Generic entries are supported.                          |
| /O        | Defines the organization defined in a certificate. This is also known as the company.<br><br>Generic entries are supported.                                  |
| /L        | Defines the locality defined in a certificate. This is also known as the city.<br><br>Generic entries are supported.                                         |
| /ST       | Defines the state/province defined in a certificate.<br><br>Generic entries are supported.                                                                   |
| /C        | Defines the country defined in a certificate.<br><br>Generic entries are supported.                                                                          |
| /SN       | Defines the serial number defined in a certificate.<br><br>Generic entries are not supported.                                                                |

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

| Parameter | Description                                                                                                                                                                                                                                                                                                                                       |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           | <ul style="list-style-type: none"> <li>• If the end time is after the current date, then authorization file checking turns to the next parameter.</li> <li>• If the end time is before the current date, then the transfer request is terminated, and an error is sent to the remote system.</li> </ul> <p>Generic entries are not supported.</p> |
| /USER     | When defined, the user associated with the SSLAUTH entry is saved and replaces the user ID and password sent by the client. This parameter allows authentication without requiring a user ID and password. /USER=root is not allowed.                                                                                                             |

## 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, 2018 to November 30, 2019, 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=20181201,
 /EDATE=201911300
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 `CfExitData.h` file, which is located in the `$CFROOT/samples` directory. This file contains the data structure with all the information to be passed to your exit program.

For more information, see [CfExitData 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
```

## 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 IpAddress[255];
 char RemoteUserId[255];
 char RemoteDomain[255];
 char LocalFileName[255];
 char RemoteFileName[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"> <li>• x00: Success</li> <li>• x01: Failure</li> <li>• x09: Abort</li> </ul>                                                                                                                                                                        |
| StatusSeverity | BYTE      | Indicates the severity of the transfer status: <ul style="list-style-type: none"> <li>• x00: Success</li> <li>• x01: Informational</li> <li>• x02: Warning</li> <li>• x03: Error</li> <li>• x10: No Checkpoint</li> <li>• x20: No Restart</li> <li>• x80: Retry network error</li> <li>• x81: Retry file error</li> </ul> |

| Field Names      | Data Type | Field Values                                                                                                                                                                                                                                                                       |
|------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Compression      | BYTE      | Indicates the type of compressions: <ul style="list-style-type: none"> <li>• x00: No compression</li> <li>• x11: LZ compression</li> <li>• x12: RLE compression</li> </ul>                                                                                                         |
| RecordFormat     | char[2]   | Indicates the remote file record format: <ul style="list-style-type: none"> <li>• F: Fixed blocked</li> <li>• V: Variable blocked</li> <li>• U: Undefined</li> <li>• X: Fixed</li> <li>• Y: Variable</li> </ul>                                                                    |
| BlockSize        | Short int | Indicates the z/OS dataset block size.                                                                                                                                                                                                                                             |
| PermittedActions | BYTE      | Indicates the type of permitted actions: <ul style="list-style-type: none"> <li>• x00: None</li> <li>• x02: EOF</li> <li>• x04: CRLFEOF</li> <li>• x08: System</li> <li>• x10: Hidden</li> <li>• x20: Archive</li> <li>• x40: Read Only</li> <li>• x80: NTFS Compressed</li> </ul> |
| StatusMsg        | char[255] | Indicates the text message with transfer status.                                                                                                                                                                                                                                   |
| LocalUserId      | char[255] | Indicates the local user ID that initiates the transfer.                                                                                                                                                                                                                           |

| Field Names         | Data Type | Field Values                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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> <p><b>Note:</b> It is a z/OS-specific parameter.</p> |
| EncryptionType      | DWORD     | <p>Indicates the type of encryption:</p> <ul style="list-style-type: none"> <li>• x80: Data is not encrypted (check x00 too)</li> <li>• x40: DES Encryption</li> <li>• x20: 3DE Encryption</li> <li>• x10: Blowfish Encryption</li> <li>• x08: Blowfish Long</li> <li>• x04: Rijndael</li> <li>• x00: No encryption (check x80 too)</li> </ul>                                                                                                      |
| VolSer              | char[7]   | Indicates the remote file volume name.                                                                                                                                                                                                                                                                                                                                                                                                              |

| Field Names    | Data Type | Field Values                                                                                                                                                            |
|----------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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"> <li>• T: Tracks</li> <li>• C: Cylinders</li> <li>• K: Kilobytes</li> <li>• M: Megabytes</li> </ul> |
| NewFileAvail   | char[2]   | Indicates whether the new file is available: <ul style="list-style-type: none"> <li>• I: Immediate</li> <li>• D: Deferred (Tape)</li> </ul>                             |
| IpAddress      | char[255] | Indicates the IP address of the remote system.                                                                                                                          |
| RemoteUserId   | char[255] | Indicates the user ID on 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.                                                                                                                 |
| WriteMode      | char[2]   | Indicates the options for file creation: <ul style="list-style-type: none"> <li>• R: Replace</li> <li>• A: Append</li> <li>• C: Create</li> </ul>                       |

| Field Names       | Data Type | Field Values                                                                                                                                                                                 |
|-------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                   |           | <ul style="list-style-type: none"> <li>• X: Create / Replace</li> <li>• Y: Create / Append</li> <li>• Z: Create / Replace / New</li> </ul>                                                   |
| TransferFunction  | char[2]   | Indicates the transfer types: <ul style="list-style-type: none"> <li>• S: Send</li> <li>• R: Receive</li> </ul>                                                                              |
| 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"> <li>• F: File to File</li> <li>• P: File to Print</li> <li>• C: Remote Command</li> <li>• J: File to Job</li> </ul> |
| CheckPointRestart | char[2]   | Indicates whether to use checkpoint: <ul style="list-style-type: none"> <li>• Y: Checkpoint used</li> <li>• N: Checkpoint not used</li> </ul>                                                |
| CR_LF             | char[2]   | Indicates whether to use the CR/LF delimitation: <ul style="list-style-type: none"> <li>• Y: Yes</li> <li>• N: No</li> <li>• L: Line Feed only</li> </ul>                                    |
| DataType          | int       | Indicates the type of data: <ul style="list-style-type: none"> <li>• 0: Binary</li> <li>• 1: ASCII</li> </ul>                                                                                |

| Field Names  | Data Type | Field Values                                                                                                                                                                                                                                                                                                                                         |
|--------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              |           | <ul style="list-style-type: none"> <li>• 2: EBCDIC</li> </ul>                                                                                                                                                                                                                                                                                        |
| Port         | int       | Indicates the IP port number used for transfer.                                                                                                                                                                                                                                                                                                      |
| 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"> <li>• 0: transfer is successful and never retried.</li> <li>• 4: transfer is unsuccessful due to a network error and to retry later.</li> <li>• 8: transfer is unsuccessful due to something other than a network error and to retry later.</li> </ul> |
| TryCount     | int       | Indicates the value assigned to the TryNumber parameter.                                                                                                                                                                                                                                                                                             |
| TriedCount   | int       | Indicates the number of attempts for transfer.                                                                                                                                                                                                                                                                                                       |

## cfunix2dos.exe and cfdos2unix.exe Utilities

You can use the `cfunix2dos.exe` utility to convert a text file from UNIX format to DOS format; and, you can use the `cfdos2unix.exe` utility to convert a text file from DOS format to UNIX format.

### cfunix2dos.exe Utility

Use the `cfunix2dos.exe` utility to convert LF to CRLF characters. 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
```

## `cfdos2unix.exe` Utility

Use the `cfdos2unix.exe` utility to convert CRLF to LF characters. As a result, you can send the file to Windows in binary format.

When using the `./cfdos2unix.exe` utility, the format is:

```
./cfdos2unix.exe filename
```

## Example of `cfdos2unix.exe` Utility

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

```
./cfdos2unix.exe /usr/tmp/file.txt
cfdos2unix complete for file ==> /usr/tmp/file.txt
Input bytes=3131
Output bytes=3074
```

# Appendix A: 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.

**i Note:** 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
```

## Appendix B: Activating CyberMgrBackup in Non-HA Environment

---

After the installation completes, the `bin` folder includes a `CyberMgrBackup` file, which is a soft link to `CyberMgr`. By running the `backupinst` script, the secondary daemon becomes available. This feature provides a backup to the existing `CyberMgr`. If the `CyberMgr` process stops, `CyberMgrBackup` takes over till `CyberMgr` is restarted by running the `startmgr` script.

This feature is merely a matter of convenience and does not provide any functionality gain. In an HA environment, the `CyberMgr` Secondary RPC Server is used instead for backing up Primary `CyberMgr`.

### Procedure

1. Stop the current active process.  
`cfstop`
2. Run the backup configuration setup script.
  - Configure with the default backup port number (46679).  
`backupinst`
  - Or, configure with any other backup port number (for example, 46677).  
`backupinst 46677`

```
---new entries will appear in your $CFROOT/config/config.txt---
--
CyberMgrPortLocalBackup: 46679 { N, PortNumber
}
HADirectory: /mftps { N, DirName
}
HACyberMgrPrimary: 127.0.0.1:46678 { IpName/Address:Port
}
HACyberMgrSecondary: 127.0.0.1:46679 { IpName/Address:Port
}
```

3. Start `CyberMgrBackup` process.  
`startmgr backup`

```
MFT Platform Server Manager Is Started Fri Sep 25 10:40:43 EDT
2020
```

- Restart TIBCO MFT Platform Server with backup availability.

```
cfstart
```

Now, CyberMgrBackup feature is installed and available to use.

**i Note:** "cfstart" script only starts CyberMgr and CyberResp. "startmgr backup" starts CyberMgrBackup. "startmgr backup" can be run at any time before or after "cfstart". CyberMgrBackup feature is optional. You do not have to activate it, even if it is installed.

**i Note:** "cfstop" stops all three daemons: CyberMgr, CyberMgrBackup, CyberResp.

## Uninstalling CyberMgrBackup

You can uninstall CyberMgrBackup at any time by running the command:

```
backupuninst $CFROOT/config/config.txt
```

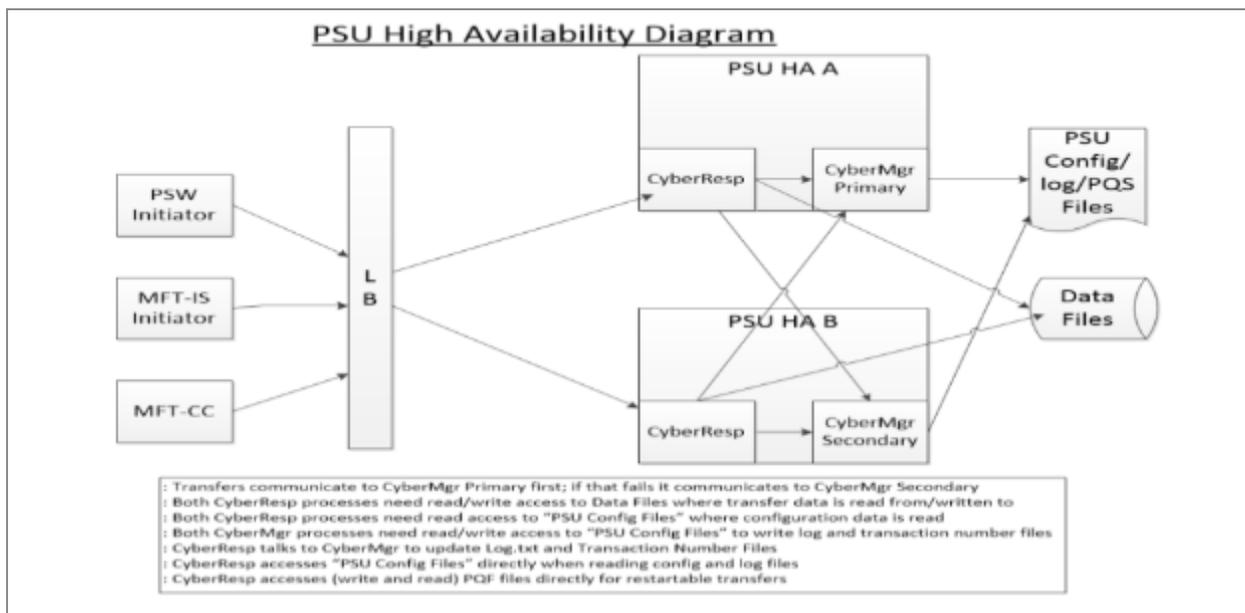
```
Uninstall backup configuration completed.
```

```
-----entries will be removed from your $CFROOT/config/config.txt-----
CyberMgrPortLocalBackup: 46679 { N, PortNumber }
HADirectory: /mftps { N, DirName }
HACyberMgrPrimary: 127.0.0.1:46678 { IpName/Address:Port }
HACyberMgrSecondary: 127.0.0.1:46679 { IpName/Address:Port }
```

**i Note:** While running with CyberMgrBackup only (CyberMgr primary is down), admin and message log files are not populated. Restart CyberMgr as soon as possible to obtain the full capability.

## Appendix C: Active/Active High Availability Support

TIBCO MFT Platform Server for UNIX now supports Active/Active High Availability (HA) when running behind a load balancer. This allows multiple systems to appear to the transfer partner as a single system. The following figure illustrates how high availability works:



- Platform Server clients, Internet Server clients, and Command Center administrative functions connect to a load balancer.
- The load balancer connects the client to any of the Platform Server instances in the cluster.
- The Platform Server instance contacts the Primary CyberMgr to get transaction numbers and write audit logs. If the Primary CyberMgr is not running, then it contacts the Secondary CyberMgr.
- Platform Server can directly read the configuration files from a common location accessible to all instances in the cluster.
- Transfers can execute on any of the instances in the cluster.
- Platform Transfer restarts can execute on any of the instances in the cluster.

- Audit inquiry can execute on any of the instances in the cluster.
- Transfers initiated by HA cluster Platform Server Clients have not changed, except that they use CyberMgr to get transaction numbers and write audit records.

These are the goals of the Platform Server HA feature:

- Multiple (two or more) PSU servers appear to be a single PSU Server
- Share configuration files
- Unique transaction numbers
- Support checkpoint Restart
- No changes to PS Client User Experience
- Share Log.txt (i.e. Transfer Audit) files
- BW-initiated transfers can poll target servers for transfer completion
- Clients can connect to any Platform Server in the cluster
- Command Center Collection works seamlessly even when connecting to different Platform Servers.

## Key Components of Platform Server HA

Single location for files and data shared by multiple server in the cluster are as follows:

- Config files, (except for config.txt) nodes, and profiles
- Log.txt files
- Transaction Number file
- PQF files
- Audit file inquiry (cfinq or Command Center initiated)
- Active transfer inquiry

CyberMgr RPC Server handles all access to configuration and log files. The primary and secondary CyberMgr servers provide failover in case of a failure.

## Key Features of CyberMgr

CyberMgr is started on each PSU Instance. It writes to all the files that need single threading. These files are: Transaction Number, Log.txt, and Messages.txt.

CyberMgr is used for both HA and non-HA installations. CyberMgr runs with the same authority as CyberResp, which is root, when CyberResp runs as root.

The `config.txt` parameter that defines the TCP Port CyberMgr listens on `CyberMgrPortLocal`: 46678.

Transfers fail if both primary and secondary CyberMgr are down.

CyberMgr performs the following tasks:

- Gets Transaction Number
- Writes Log.txt
- Writes message log (always done by Local CyberMgr)
- Writes audit log (always done by Local CyberMgr)

## HA Parameters (`config.txt`)

The following parameters define the HA configuration parameters

```
[COMMON]
CyberMgrPortLocal: 46678
.
HADirectory: /common/directory/HATest
HACyberMgrPrimary: primary.cybermgr.acme.com:46678
HACyberMgrSecondary: secondary.cybermgr.acme.com:46678
CyberMgrTraceLevel: N { N,Y,D }
CyberMgrTracePath: /data/ps810/trace/Responder
VRefreshInterval: 10 { # of sec, 0 turn off visibility }
```

The following parameters define configuration parameters used by HA and non-HA mode

`ConfigDirectory`: `/common/directory/HATest/config`

`PQFDirectory`: `/common/directory/HATest/PQF`

`TransnumFileName`: `/common/directory/HATest/trn/transnum`

`LogEventFileName`: `/common/directory/HATest/log/Log.txt`

| <b>Config.txt Parameter</b> | <b>Description</b>                                                                                                                                                                                                                                                                |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CyberMgrPortLocal           | Defines the port that the local CyberMgr listens on. This parameter is used in both HA and non-HA mode.                                                                                                                                                                           |
| HADirectory                 | Defines a common location that all Platform servers in the cluster can access. To disable HA mode, set the value to "N".                                                                                                                                                          |
| HACyberMgrPrimary           | Defines the IP name or IP address and port of the Primary CyberMgr.                                                                                                                                                                                                               |
| HACyberMgrSecondary         | Defines the IP name or IP address and port of the Secondary CyberMgr.                                                                                                                                                                                                             |
| CyberMgrTraceLevel          | Defines the trace level for CyberMgr. Set parameter only if directed to by TIBCO Support.                                                                                                                                                                                         |
| CyberMgrTracePath           | Defines the CyberMgr trace directory.                                                                                                                                                                                                                                             |
| VRefreshInterval            | Defines the frequency of active transfers reporting their state to CyberMgr. To disable active transfers tracking, set the value to 0.                                                                                                                                            |
| ConfigDirectory             | Defines the directory where configuration files, (other than config.txt) are located. In HA mode, this name should be the same as the HADirectory suffixed with "/config".                                                                                                        |
| PQFDirectory                | Defines the directory where the PQF (restart) files are located. When running in HA mode, this directory should be accessible for read/write from all instances in the HA cluster.                                                                                                |
| TransnumFileName            | Defines the fully qualified name of the transaction number file. When running in HA mode, this file must be accessible for read/write by CyberMgr Primary and Secondary daemons in the HA cluster.                                                                                |
| LogEventFileName            | Defines the fully qualified name of the audit log file. When running in HA mode, this file must be accessible for read from all instances in the HA cluster and for write by CyberMgr Primary and Secondary daemons. The current date ".YYYYMMDD" is suffixed to the LogFileName. |

# HA Requirements

To enable HA mode, the following requirements must be met:

- All Platform Servers in the HA Cluster must define the same `HACyberMgrPrimary` and `HACyberMgrSecondary` parameters in the `config.txt` in the same order.
- Each Platform Server must run a local `CyberMgr` daemon.
- All PSU in the HA cluster must have access to the same shared Config directory.
- All PSU must have read/write access to the files being transferred.
- All PSU must have read/write access to the shared directories where the PQF, Transnum, and Log.txt files are located.
- Firewalls must allow communication to the Primary and Secondary `CyberMgr` ports.
- Config files should be in a common shared directory and each system in the cluster should have read/write access to the config file directory and files.
- Platform Servers in a HA cluster should be the same endian (little endian or big endian) server type. For example, You should not mix AIX (big endian) and Linux x86 (little endian) servers in the same cluster.
- More than two Platform Servers instances can be in the same cluster. Each Platform server must run a local `CyberMgr` even if they point to `HACyberMgrPrimary` and `HACyberMgrSecondary` on different machines.

The end result:

The PS client thinks that two (or more) PSU servers appear to be one PSU Server

## Installing Platform Server in HA Mode

To install Platform Server in HA mode, complete the following steps:

### Procedure

Run the `install` or `install.noroot` script.

The `install` or `install.noroot` script prompts for HA mode installation.

 **Note:** You can configure the silent installer to enable HA mode install as well.

If you have installed Platform server in a non-HA mode, you can use the `hainstall` script to configure HA mode. The `hainstall` script can be used to change parameters on existing HA mode as well.

## Starting and stopping CyberMgr

### Starting CyberMgr

CyberMgr is started the first time `cfstart` is executed.

Each `cfstart` (`-ssl`, `-tunnel`, and so on) checks if CyberMgr is started automatically.

The `startmgr` script can be used to start CyberMgr manually.

### Stopping CyberMgr

CyberMgr is stopped when the last CyberResp is stopped.

Each `cfstop` (`-ssl`, `-tunnel`, and so on) checks if CyberMgr is active. If all responders are stopped, CyberMgr is stopped.

There is no script to stop CyberMgr manually. You can use a `"kill -9 pid"` command to manually stop CyberMgr.

## Options for Configuring Primary and Secondary CyberMgr

There are two options for configuring the Primary and Secondary CyberMgr:

1. Configure a Direct Connection to the Primary and Secondary CyberMgr

Define the `config.txt` parameters to point directly to the CyberMgr instances.

```
HACyberMgrPrimary: primary.cybermgr.acme.com:46678
HACyberMgrSecondary: secondary.cybermgr.acme.com:46678
```

In this case, Platform Server would always connect to the Primary CyberMgr. If connections to the Primary CyberMgr fail, Platform Server attempts to connect to the Secondary CyberMgr.

This option requires network connectivity to the Primary and Secondary CyberMgr.

The only disadvantage to this approach is that if the Primary CyberMgr is down, connections to the Primary CyberMgr may take a few seconds to timeout, thus causing transfers to take longer.

2. Configure the Connection to the Primary and Secondary CyberMgr to run through a load balancer in Active/Passive mode

Define the config.txt parameters to point directly to the load balancer.

```
HACyberMgrPrimary: loadbalancer.cybermgr.acme.com:46678
HACyberMgrSecondary: loadbalancer.cybermgr.acme.com:46678
```

In this case, Platform Server would always connect to the Primary CyberMgr which in this case is a load balancer.

The load balancer must be configured to connect to the Primary and Secondary CyberMgr in Active/Passive Mode.

The load balancer keeps track of which CyberMgr is active and routes the request to the Primary CyberMgr when active. It only routes the request to the Secondary CyberMgr if the Primary CyberMgr is unreachable.

The advantage to this approach is that when the Primary CyberMgr is down, the load balancer automatically routes requests to the Secondary CyberMgr without a delay.

This option requires network connectivity from the Platform Servers to the load balancer. The load balancer must have connectivity to the Primary and Secondary CyberMgr instances.

**i Note:** When using a load balancer to connect to CyberMgr, the load balancer must be configured to work in Active/Passive mode. Otherwise, duplicate transaction numbers and possible corrupted Log.txt (audit) files are generated.

# TIBCO Documentation and Support Services

---

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

## How to Access TIBCO Documentation

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

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

## Product-Specific Documentation

The following documentation 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 Release Notes*
- *TIBCO® Managed File Transfer Platform Server for UNIX Managed File Transfer Overview*
- *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 Security Guide*
- *TIBCO® Managed File Transfer Platform Server for UNIX Docker Container Deployment*

## 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, TIBCO Managed File Transfer Suite, TIBCO Managed File Transfer Command Center, TIBCO Managed File Transfer Internet Server, TIBCO Managed File Transfer Platform Server 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 Corporation and/or its affiliates.

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

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

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

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

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

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. 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.