TIBCO MFT Platform Server™ Agent Installation and Operations Guide

Software Release 7.1
November 2011

TIBCO provides the two-second advantage™

TIBCO®
The Power of Now®

Important Information

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 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.

TIBCO, The Power of Now, TIBCO Managed File Transfer, TIBCO Managed File Transfer Command Center, TIBCO Managed File Transfer Internet Server, TIBCO Managed File Transfer Platform Server, TIBCO Managed File Transfer Platform Server Agent, Edge Server, RocketStream Accelerator, and Slingshot are either registered trademarks or trademarks of TIBCO Software Inc. or its subsidiaries in the United States and/or other countries.

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

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.

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.

TIBCO® Managed File Transfer Internet Server with RocketStream® Accelerator is entitled TIBCO® Managed File Transfer Internet Server in certain other product documentation and in user interfaces of the product.

Contents

Preface

This user's guide explains how to use TIBCO MFT Platform Server™ Agent.

Topics

Related Documentation

This section lists documentation you may find useful.

TIBCO MFT Platform Server™ Agent Documentation

The following documents form the TIBCO MFT Platform Server™ Agent documentation set:

- *TIBCO MFT Platform Server™ Agent Installations and Operations Guide* Read this manual for instructions on site preparation, installation, and on using the product to perform transfer requests and more between other Platform Server products.
- *TIBCO MFT Platform Server™ Agent Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

How to Contact TIBCO Customer Support

For comments or problems with this manual or the software it addresses, contact TIBCO Support, as follows:

- For an overview of the TIBCO Support and information on getting started with TIBCO Support, visit http://www.tibco.com/services/support
- If you already have a valid maintenance or support contract, visit https://support.tibco.com

  Entry to this site requires a user name and password. If you do not have to login credentials, click Register with Support.
- Technical Support email address support@tibco.com
- Technical Support Call Centers:
  - North and South America: +1.650.846.5724 or +1.877.724.8227 (1.877.724.TACS)
  - EMEA (Europe, Middle East, Africa): +44 (0) 870.909.3893
  - Australia: +61.2.4379.9318 or 1.800.184.226
  - Asia: +61 2 4379 9318

Installation

This section explains how to install TIBCO MFT Platform Server™ Agent.

Topics

System Requirements

Java

The Oracle Java JDK (Java SE JDK) version 1.6.0_xx or greater, must be downloaded and installed to the server if it is not already in place. This JDK is required to install PSA and allow transfer users to execute transactions with MFT Platform Server™ Agent. The Java JDK can be downloaded using the following link:
http://www.oracle.com/technetwork/java/javase/downloads/index.html

Java Policy Files

Along with the Java JDK applied to your system, you will also need to download and install Oracle's Java's policy files for 256bit encryption (a.k.a Java Cryptography Extension) for the version of Oracle Java JDK you have installed. Using the link from the Oracle Java JDK section above search for "Java Cryptography Extension", if you do not see the policy files for the version of Java you are using look for "Previous Releases".

Note: For MacOS X system's running Leopard v10.5 or Snow Leopard v10.6 the jdk is installed by default. No addition files are needed.

License Files

PSA requires a license key that is based on the machine name. While the PSA product comes with a 60 day trial license key, to receive a license key with a later expiration date please contact TIBCO's technical support team and supply the machine name of the server you will be installing the product on. If you are

installing on a UNIX system the machine hostname can be obtained by running the following UNIX command:

uname -n

*Note: Support is provided by TIBCO only for the vendor's generally supported release versions. Once the operating system goes into extended support mode, or the vendor no longer supports a version, it will cease to be supported by TIBCO Technical Support.

Install

This section will step you through the installation of MFT Platform Server™ Agent 7.1. The Installation involves downloading the installation files. Please note that these are general steps and may not apply for all environments. You can install PSA using default settings or you can choose to run a custom install. Both are demonstrated in this section.

Uploading PSA to your server

You should have received the MFT Platform Server™ Agent software jar file named psa-install.jar. You will need to upload the software to your system. If you are using a UNIX platform we have included an example below that uploads the software using an FTP connection:

Step1:

From your windows machine you would open up Command Prompt and navigated to the directory where the psa-install.jar file is located. Then you would establish an FTP connection to your UNIX platform with the following command:

ftp <UNIX machine name or ip address>

Step 2:

You would log on to the UNIX machine with the proper UNIX credentials.

Step 3:

We change to your default directory to unjar the installation files to. We used **/usr/tmp** in the following command:

cd /usr/tmp

Step 4:

We then put our connection into binary mode with the following command:

bin

Step 5:

Finally we uploaded our PSA software to our UNIX box by using the following put command:

put psa-install.jar

When the file finishes copying, you exit from the FTP session by typing the bye command:

bye

Unjar psa-install.jar

Once you have your psa-install.jar file on the machine you will be installing the product on, your next step is to unjar the file using the following command:

jar –xvf psa-install.jar

You should see the following files after the unjar command completes:

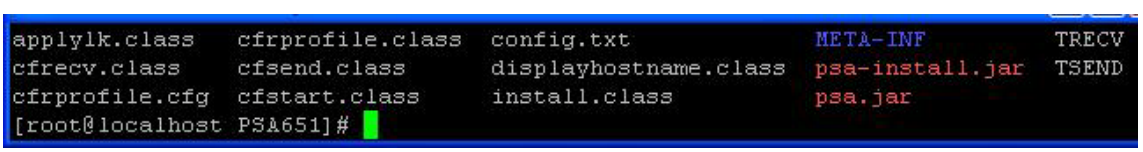


Figure 1

Default Install

The MFT Platform Server Agent can be installed by any user including root. To install PSA into the default ./PSA directory using the 60 day trial license key run the following command:

java –cp psa.jar install

You will be presented with an End User License Agreement (EULA) that you must accept before the installation will continue. If you choose not to accept the EULA the installation will stop.

You will be shown the directory the product will be installed in and asked if you want to install in that directory. If you say no you will be given the opportunity to choose a different directory.

Custom Install

Running a custom install allows you to choose a different default directory to install the product in as well as allow you to include a new license key (See the section Apply License Key for more information on receiving a new license key.) on the command line at this time. If you do have a new license key at this time you can apply one to the system at a later time using the **applylk.class** which is discussed later in this section. Use the following command to perform a custom install:

java –cp psa.jar install –d InstallDirectory –k LicenseKey

-d, Directory to install PSA in to *(Note: If your install directory path contains any spaces it must be contained in quotes. e.g.; "Install Directory")*

-k, New license key to be applied during the install to replace 60 day trial key

-?, Help

Below is example output from a custom installation:

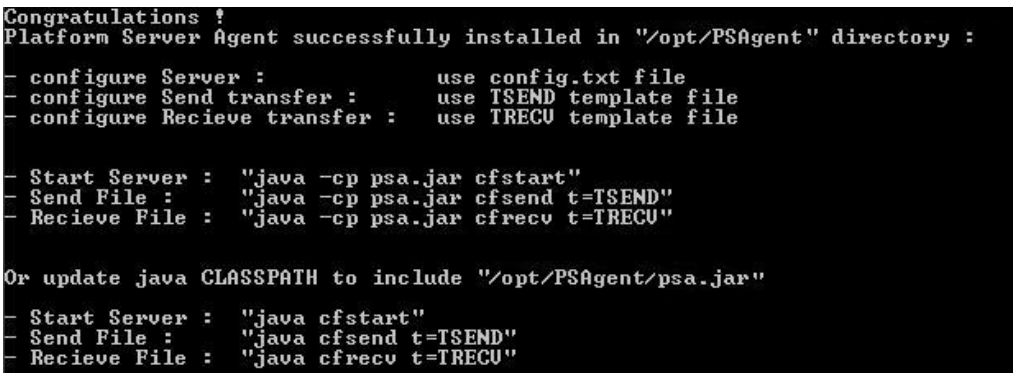


Figure 2

Set the CLASSPATH

Notice in the example command we have shown we include the classpath each time (e.g., -cp psa.jar). If you do not want to set the classpath on the command line each time you run a command you can either add/update the CLASSPATH environment variable on your system and set it to the full path including the psa.jar file name. For example, to start PSA you would run the following command:

java –cp psa.jar cfsend t:TSEND

If the CLASSPATH is configured you would run

java cfsend t:TSEND

Apply License Key

The PSA product comes with a default 60 day trial license key. When you need to install a new license key you will need to provide TIBCO with your operating systems machine name. Please run the following command and send the output to the TIBCO Technical Support team:

java –cp psa.jar displayhostname

To apply a new license key after you have installed the program you would use the **applylk.class** command:

java –cp psa.jar applylk –k LicenseKey

-d, Directory where PSA is installed *(Note: If your install directory path contains any spaces it must be contained in quotes. e.g.; "Install Directory")*

-k, New license key to be applied to PSA

-?, Help

Note: If you have already installed the program in the default directory you only need to define the –k option on the command line to set a new LicenseKey.

Configure MFT Platform Server™ Agent

This section describes how to configure your MFT Platform Server™ Agent to be used as a Responder (Server) and as an Initiator (Client). When you install MFT Platform Server™ Agent it is able to run as it is as long as the default port 47474 is available on your server. Because each environment is different we provide an easy to use configuration file to allow you to fine tune your MFT Platform Server™ Agent settings.

Topics

Config.txt

The PSA program can act as both a server and client. To configure the server options open the file **config.txt** located in the directory <InstallDirectory>/config/. Below we discuss configuring the server components and the various options that are available.

Parameter Settings

The Server configurations are broken up into three sections, Server Parameters, Client Parameters, and a Common Parameters section that the server and client share. Whenever you make a change to these parameters you must restart the server for the changes to take place. In the tables below is a brief description for each parameter.

Server Parameters

These parameters are used by the server component of PSA.

Parameter	Definition
Port	The port the server is listening on. Default is **47474**
TraceLevel	The PSA server component comes with 3 levels of tracing: **L** – Low, **M** – Medium, **H** – High. This tracing is used for capturing data on incoming transactions. By default this tracing is turned off (**N**). When tracing is turned on messages will be redirected to trace files located in the trace directory defined in the TracePath server parameter.
TracePath	The path where trace files will be written if tracing is turned on. The default path is <InstallDirectroy>**/trace/Responder**.
Timeout	This controls the amount of minutes before a transfer will timeout should the transfer fail for some reason. Default is **120** minutes.

Client Parameters

These parameters are user by the Client component of PSA.

Parameter	Definition
TraceLevelClient	The PSA client component comes with 3 levels of tracing: **L** – Low, **M** – Medium, **H** – High. This tracing is used for capturing data on outgoing transactions. By default this tracing is turned off (**N**). When tracing is turned on messages will be redirected to trace files located in the trace directory defined in the TracePathClient server parameter.
TracePathClient	The path where trace files will be written if tracing is turned on. The default path is <InstallDirectroy>/**trace/Initiator**.
TimeoutClient	This controls the amount of minutes before a transfer will timeout should the transfer fail for some reason. Default is **120** minutes.
LogTempErrors	Indicates whether all transfer attempts will be logged or should only the final attempt to be logged when a transfer is retried.

Common Parameters

These parameters are used by both the Server and the Client components of PSA.

Parameter	Definition
LogEventPath	Path where log files will be written daily with information for each transaction that occurs with the server and client. The log files written will use the following format:

Starting and Stopping PSA

Start Server

Once you have completed making any edits to the PSA server's config.txt and have saved the file as discussed in section 2.4 you can start the PSA Responder. Note: Transfer requests coming in to the responder will be run under the user account that starts the PSA Responder. The following command will start the server:

java –cp psa.jar cfstart

If any of the program commands are run by users other than the user id that started the application, file permissions may need to be edited for the others to use the program successfully.

Note: A dedicated session connection is required as the program is running in the foreground.

Stop Server

To stop PSA running in the foreground you would press the <Ctrl> + <C> keys at the same time. If the program is running in the background for instance on a UNIX platform you can use the following commands to stop the deamon:

ps –ef | grep cfstart

Then you would use the PID in the following kill command:

kill -9 PID

Responder Profiles

PSA uses responder user profiles for authentication on incoming requests. This is necessary if your PSA instance will be responding to transaction requests from remote systems. A responder user profile needs to exist for the user(s) the transactions will be using to login to the system with.

Adding a Profile

To create a responder profile you would run the following command:

java –cp psa.jar cfrprofile

You will be prompted to enter the remote uid (No domain name is needed) and the users remote password. For instance if the incoming transaction will be coming in from a windows user with the user ID DomainName\JohnDoe you would enter the following:

Enter remote uid: JohnDoe
Enter remote password:

The user ID and password (encrypted) are stored in the <InstallDirectory>/config/cfrprofile.cfg file.

Deleting a Profile

To delete a responder profile you would run the following delete command:

java –cp psa.jar cfrprofile –d *username*

Listing User Profiles

To list all the responder profiles you would run the following command:

```
java –cp psa.jar cfrprofile -l
```

Help

At any time you can display the online help using the following command:

```
java –cp psa.jar cfrprofile -?
```

or

```
java –cp psa.jar cfrprofile -h
```

Sending and Receiving Files

This section describes how to conduct file transfers using MFT Platform Server Agent client. The client component of PSA is used to initiate file transactions. These transactions can be sent to other systems running an MFT Platform Server™ Agent as well as be sent to a system running TIBCO's MFT Platform Server™ Agent, MFT Platform Server™ or MFT Internet Server.

Topics

Command Line Transfers

To send files manually you can setup transfers on the command line using the cfsend or cfrecv command along with required and optional parameters with their values in a command on the command line.

To send a file using PSA you would use the cfsend command:

java –cp psa.jar cfsend RequiredParameters:Value OptionalParameters:Value

Example:

java –cp psa.jar cfsend ip:10.1.2.200 port:47474 lf:testfile.txt rf:”c:\incoming\testfile.txt” uid:JohnD pwd:q1w2e3r4

To receive a file using PSA you would use the cfrecv command:

java –cp psa.jar cfrecv RequiredParameters:Value OptionalParameters:Value

Example:

java –cp psa.jar cfrecv ip:10.1.2.200 port:47474 lf:ReceivingFile.txt rf:”c:\outgoing\testfile.txt” uid:JohnD pwd:q1w2e3r4

If you need to refresh your memory on what parameters can be used by the cfsend or cfrecv commands you can get a list by running the help command for either. Below is an example:

java –cp psa.jar cfsend -?

You can review the full list of the parameters available at the end of this section for the cfsend and cfrecv utilities along with a brief description for each.

Note: Depending on the operating system you are sending or receiving from either single or double quotes may be needed around the remote file name as seen in our examples above.

Transfers using Templates

Up until now we have discussed file transfers that are executed from the command line manually. However, you could configure transfers in a file to be run at a later date by using a template file. These templates make sending and receiving transfers easier for a user because the defined transfer can be saved to run again and again. Information such as the local and remote file names, IP Address, IP port, user id and password are all defined in the template file.

We have included two sample templates called TSEND and TRECV which you will find in your installation directory. Open one up and you will see all the parameters available for a transfer. Simply fill in the file transfer information needed for the transfer and save the file. Below is an example cfsend command to run the transfer using a template:

java –cp psa.jar cfsend t:*TransferTemplateName*

Note: If the template file(s) are being stored for transfers in a different directory then the installation directory then you must define the entire path. If just a template file name is defined in the parameter then you must be running the transfer from the install directory.

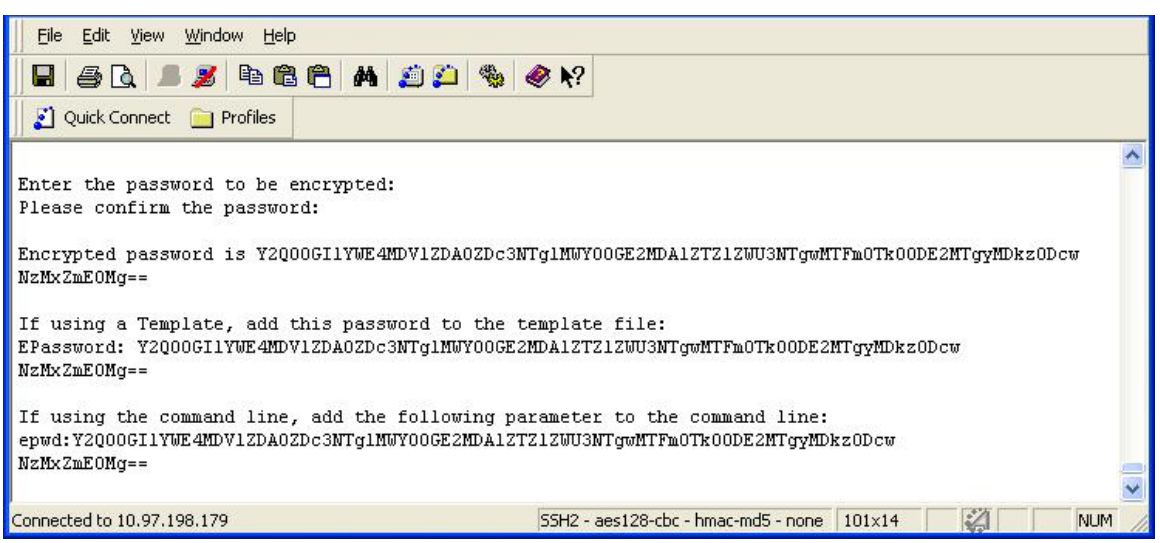
Encrypting Passwords

In the previous send and receive transfer examples for the Command Line and transfer templates we have shown the user id's password being used in clear text format. To use an encrypted password in a transfer command line or template you can use the cfencrypt command to produce a 64 bit encrypted password to be used in place of the clear text password.

To encrypt a transfer users password you would run the following PSA java command, notice there are no parameters required for this command:

java –cp psa.jar cfencrypt

You will be prompted to enter a password and then again to confirm the password. This will produce the encrypted password as well as the value being used in two different parameters to be used in a template or on a command line to as you will seen in our example below (we used password: abcdef):



Note: Your screen may break the encrypted password up over more than 1 line as you can see in our example screenshot above. When you copy and paste the password to a location be careful not to copy any spaces at the end of each line that appears on your screen and save the password in one continuous line.

Now we can edit the example we showed earlier with JohnDoe's password being shown in clear text we can now use (Take note the parameter pwd is now epwd):

java –cp psa.jar cfsend ip:10.1.2.200 port:47474 lf:testfile.txt rf:"c:\incoming\testfile.txt" uid:JohnD epwd:Y2Q0OGI1YWE4MDVlZDA0ZDc3NTg1MWY0OGE2MDA1ZTZlZWU3NTgwMTFmOTk0ODE2MTgyMDkzODcw NzMxZmE0Mg==

Either the long format of **EPassword** or the short parameter format of **epwd** can be used on the command line or in a transfer template.

Transfer Parameters

Required Parameters

The parameters below are required to be set when sending or receiving files.

Parameter (Shortcut Parm)	Definition
LocalFileName (lf)	The name of the local file that is to be transmitted. This parameter is case sensitive.
RemoteFileName (rf)	The name of the file or data set at the remote location. If backslashes are used in the file name, such as in a file going to a Windows system, then the remote file name should be enclosed in double quotes if specified on the command line. If a member name is being used, as in a dataset going to a mainframe, then the entire dataset may be enclosed in double quotes or backslashes may be used before the parentheses when specifying the remote file name on the command line.
RemoteHost (ip)	This is the IP Name or the IP Address of the remote location.
port	Is the IP Port on which is listening on the remote system. Valid values are 1024 to 65535.
UserId (uid)	The remote UserID for the file transfer, this ID must exist on the remote system and have permission to access to the remote file (rf). For Windows security systems use the format DOMAIN/USER. If this parameter is not specified, then PSA will treat the user as if they are a Verified User (*VER).
Password (pwd)	The password for the remote UserID (uid). This password is used on the remote system to validate the User has credentials to access the remote file. This field should be left blank when using verified users.

Parameter (Shortcut Parm)	Definition
EPassword (epwd)	The encrypted password that was generated using the cfencrypt command to be used in place of the remote UserID (uid) Password. This password is used on the remote system to validate the User has credentials to access the remote file.
TemplateFileName (t)	Set a send or receive template name to be used to give PSA all the required and optional parameter settings for a transfer request. For more information on templates see Appendix C – Transfers Using Templates

Additional File Transfer Parameters

The parameters listed below are additional settings that can be used in transfer requests.

Parameter (Shortcut Parm)	Definition
CR_LF (crlf)	Carriage Return (CR) Line Feed (LF) control for transfers between UNIX and Windows operating systems using **cfsend** or **cfrecv**. The valid values for this parameter are: Y | CRLF - CR will be deleted on a UNIX system when doing a **cfrecv**. The CR will be added before the LF when doing a **cfsend** of a text file from a UNIX system to another platform. LF - Records are delimited by LF. This is typically used when transmitting text data to z/OS. Note that the line conversion is done on the z/OS platform. No processing is done by PSA. (Only use when sending to z/OS) CRLFY - Means that CR will not be added to LF for a file transfer using **cfsend**. Likewise, CR will not be removed for the file for **cfrecv**. This is done in the rare event that a UNIX file contains CRLF, or if the application requires CRLF instead of LF. N - There are no record delimiters in the file. This is typically done for a binary transfer.

Parameter (Shortcut Parm)	Definition
ASCII_EBCDIC (eb)	This determines the type of data translation that is required for the remote system, valid values are: Binary | N - File is binary and does not require any translation. Text | Y - File is ASCII and the remote system requires EBCDIC, the data will be translated by PSA on the UNIX platform. Typically used for transfers to a z/OS system. This command is the command that turns on the use of the Local and Remote Conversion tables. For more information on using Conversion Tables see the section Appendix A -Conversion Tables.
CreationOption (co)	Remote File creation options, valid values are: R - Replace an existing file. This will only work if the remote file already exists. A - Append to an existing file. This will only work if the remote file already exists. C - Create a new file at the remote location. This will only work if the remote file does NOT exist. CR - Create a new file or Replace an existing file. This is the default option. CA - Create a new file or Append to an existing file. CRN - Create a new file and if necessary create the directory path to this file or replace an existing file.
TryNumber (trynum)	This is the number of times the transfer will be attempted. The default is 1. 0 |U |Unlimited - The transfer will be attempted 9999 times or until it is successful. It will restart the transfer from the beginning unless the CheckPoint/Restart option is set. N | 1 - The transfer will be performed only one time. 2 – 10 - Number of times the transfer will be attempted.
RetryInterval (ri)	If the TryNumber parameter in the Template file is greater than 1 then a failed transaction will be retried after this time interval. If the value of this parameter is N, the transaction will be retried immediately. Valid Values: {N, Y|1, # of min more than 1 }
EncryptionType (en)	Type of encryption that should be used for this transfer, valid values are: N - No encryption is performed DES - DES encryption is used Rijndael | RJ | AES - Rijndael encryption

Parameter (Shortcut Parm)	Definition
Compression (cmp)	Tthe type of compression to be used. Valid Values: N – No compression ZLIB2 | Y ZLIB1 - ZLIB9. ZLIB1 through ZLIB9 refer to levels of zlib compression. Level 1 is very fast but hardly compresses. Levels 7 to 9 yield the best compression but is much slower and produces the best quality of compression. Level 7 (ZLIB7) typically offers the best compromise of compression and speed.
LocalCTFile	The full path and file name of the customized local conversion table to be used for the transfer if the custom table is being stored in a different directory then the installation directory. (If just a file name is defined in the parameter you must be running the cfsend or cfrecv from the install directory.) Requires the ASCII to EBCDIC parameter to be set to Text | Y to be used. Please refer to Appendix A - Conversion Tables for more information.
RemoteCTFile	The name of the file used as the remote conversion table. Please refer to Appendix A - Conversion Tables for more information. *Note: When defining the RemoteCTFile you must also define the LocalCTFile:NULL so no translation takes place locally.*
ProcessName (pn)	This defines the 1 to 8 character process name used for the file transfer.
UserDate (ud)	This is a 25 character field for user comments.

Parameter (Shortcut Parm)	Definition
UnixPermissions (uperm)	When a file is created on a UNIX system, PSA has the ability to set the UNIX Permissions on the file. UNIX permissions are defined by a three digit number such as 777 (the same as for chmod command). The default value for this parameter is the file permissions of the file being sent or received. This works differently for a Send and for a Receive. If a Send is initiated and the UNIXPermissions value is defined, PSA use that value on the remote system. If UNIXPermissions is not defined, the permissions of the file being sent should be in the control record. If no values are passed in the control record, the responder will use the system default permissions. Note: Permissions will be set up under the file only if file was created. In other words UNIXPermissions work only with Create, CreateReplace and CreateReplaceNew file options when the file is being created.
Post_Action1 (ppa1) Post_Action2 (ppa2) Post_Action3 (ppa3) Post_Action4 (ppa4)	This is a command that will be executed upon the completion of a transfer. This command can be defined up to four times with the following format: Post_Action="S|F,L|R,COMMAND,<command data>" S | F – Success or Failure L | R – Local or Remote COMMAND – The only option currently supported by UNIX as a responder. data – The absolute path and file name of command and any parameters to be passed. This is limited to 256 bytes. No spaces are allowed in the Post_Action command. If the remote system is a mainframe, then CALLJCL, CALLPGM and SUBMIT are also supported in place of COMMAND. Please refer to the MFT Platform Server™ for z/OS documentation for more information. For more detailed information on Post Processing Actions (PPA) see the section Appendix B - Post Processing Action.

Parameter (Shortcut Parm)	Definition
SilentMode (sm)	When set to Y, no PSA messages will display on the Initiator's side screen during the file transfer. The return of the command prompt indicates that the transaction has completed. The default for this parameter is N. If the value of this parameter is N, the Local and Remote Transaction Number will be displayed on the Initiator side upon the transaction completing. If a transaction failed before the Responder side sends out the Remote Transaction Number, than it will not display. Valid values: Y | N
TimeOut (to)	Specifies the amount of time (minutes) a connection will stay open while waiting for a response from the remote side. Once the time is reached the connection is ended. Valid Values 0-1440. Default if not set is 120 minutes.

Optional Parameters For z/OS Transfers

The parameters listed below are only used when working with z/OS file transactions.

Parameter (Shortcut Parm)	Definition
RemoveTrail (rmtrail)	This parameter is only valid when you are receiving a file/s using the cfrecv command. This will remove all trailing spaces and nulls before the file is transmitted. Valid values: Y or N.
RECFM (orf)	Remote File Record format. Valid values are: F Fixed FB Fixed Blocked V Variable VB Variable Blocked U Undefined
LENGTH or LRECL (orl)	Remote File Record Length. Valid values: 1 to 32760.
BLKSIZE (obs)	Remote File Block Size. Valid values: 0 to 32760.

Parameter (Shortcut Parm)	Definition
ALLOC_TYPE (at)	Remote File Allocation Type. Valid values are: T – for Tracks, C – for Cylinders, M – for Megabytes, K – for Kilobytes
ALLOC_PRI (ap)	Remote File Primary Allocation. Valid values: 0 – 32000. PSA supports auto assign for ALLOC_PRI when the ALLOC_TYPE is M or K. If you set this value to zero, then the appropriate number of Megabytes or Kilobytes will be assigned, respectively.
ALLOC_SEC (as)	Remote File Secondary Allocation. Valid values: 0 – 32000.
VOLUME (dv)	Remote File volume name. Valid value, any 1-6 character alpha-numeric volume name.
UNIT (du)	Remote File unit name. Valid value, any 1-8 character name.
AVAIL (da)	Remote File volume availability, z/OS only. Valid values: I | Immediate, D | Deferred
DATACLASS (dc)	This represents the z/OS Data Class as defined to the Data Facility/System Managed Storage. In addition, it is used to indirectly select file attributes such as Record Format and Logical Record Length. This is a 1 to 8 character value, which contains numeric, alphabetic, or national characters (in the United States the national characters are $, #, or @). The first character must be an alphabetic or national character.
MGMTCLASS (mc)	This represents the z/OS Management Class as defined to the Data Facility/System Managed Storage. This is a 1 to 8 character value which contains numeric, alphabetic, or national characters (in the United States these are $, #, or @). The first character must be an alphabetic or national character.

Parameter (Shortcut Parm)	Definition
STORCLASS (sc)	This represents the z/OS Storage Class as defined to the Data Facility /System Managed Storage, which is used to indicate the host file's media type and the installation's backup, restore, and archive policies. This 1–8 character value must contain either numeric, alphabetic, or national characters (in the United States these are $, #, or @). The first character must be alphabetic or national.
RetenPeriod_ExpDate (rp_ed)	This parameter will be the retention period or expiration date of the file on the remote system. The format of the value entered will determine whether the parameter will be used as a retention period or as an expiration date. Retention Period is the number of days, after which the file will expire. Expiration Date is the date, in Julian format, on which the file will expire. This expiration parameter is typically used on z/OS systems for tape processing to prevent a tape from being overwritten. Care should be taken when using this with a disk file. The default is no expiration date on the file. Valid Values: { # of days, yyyy/ddd }

To display the version of the PSA product you are using at anytime you would run the following command on either the cfsend or cfrecv:

java –cp psa.jar cfsend –v
or
java –cp psa.jar cfrecv –v

Appendix A – Conversion Tables

Conversion tables allow users to convert text files between various character-set specifications. With PSA, we provide a file called **<InstallDirectory>/comtblg.dat**. This file contains the table which converts data between the ASCII and EBCDIC character sets.

Topics

- *Default Conversion Table*
- *ASCII to EBCDIC Conversion Table Example*
- *Additional Information*

Default Conversion Table

Conversion tables allow users to convert text files between various character-set specifications. With PSA, we provide a file called **<InstallDirectory>/comtblg.dat**. This file contains the table below which converts data between the ASCII and EBCDIC character sets. It looks like this:

```
00010203372D2E2F16050A0B0C0D0E0F
101112133C3D322618193F27221D351F
405A7F7B5B6C507D4D5D5C4E6B604B61
F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F
7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6
D7D8D9E2E3E4E5E6E7E8E9ADE0BD5F6D
79818283848586878889919293949596
979899A2A3A4A5A6A7A8A9C06AD0A107
9F000000000000000000000000000000
000000000000000000000000000000000
000000B100000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
002E2E2E2E2E2E2E2E2E0A2E2E0D2E2E
2E2E2E2E2E0A2E2E2E2E2E2E2E2E2E2E
2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E
2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E2E
202E2E2E2E2E2E2E2E2E632E3C282B7C
262E2E2E2E2E2E2E2E2E21242A293B5E
2D2F2E2E2E2E2E2E2E2E7C2C255F3E3F
2E2E2E2E2E2E2E2E2E603A2340273D22
2E6162636465666768692E2E2E2E2E2E
2E6A6B6C6D6E6F7071722E2E2E2E2E80
2E7E737475767778797A2E2E2E5B2E2E
2EA32E2E2E2E2E2E2E2E2E2E5D2E2E
7B41424344454647484 92E2E2E2E2E2E
7D4A4B4C4D4E4F5051522E2E2E2E2E2E
5C005354555657585 95A2E2E2E2E2E2E
303132333435363738392E2E2E2E2E2E
```

ASCII-EBCDIC portion of the translation table

EBCDIC-ASCII portion of the translation table

Figure 3

The first sixteen lines are the ASCII-EBCDIC translation table, and the next 16 lines are the EBCDIC-ASCII translation table.

ASCII to EBCDIC Conversion Table Example

To make better sense of the table above we have placed the ASCII to EBCDIC portion of the table in an Excel Spreadsheet below for demonstration purposes:

	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	00	00	00	B1	00	00	00	00	00	00	00	00	00	00	00	00
B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Figure 4

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 we go down to row 4. The second hexadecimal digit is 5 so we now move across to column 5. The point at which they meet is the hex value X’C5’. This means the hexadecimal EBCDIC value for E is X’C5’. If we wanted the E to be represented by a different EBCDIC hex value we would edit this value in the table. When a transfer request is done and the data is converted to EBCDIC the new value would be used.

Note: The ASCII character set in the default table is the standard ASCII character set it does not include the extended ASCII range which covers the special characters outside the English alphabet.

Also, this feature does not support wide or multibyte character sets at present.

The EBCDIC to ASCII translation works the same way.

For other conversions besides standard ASCII to EBCDIC, you can copy the default **comtblg.dat** file to create new customized tables of your own to be used. These can be assigned per transfer and when using transfer templates if desired.

Additional Information

- To activate Conversion tables, you must enable the **ASCII_to_EBCDIC** parameter. When enabled, it uses the file names that are specified in the **LocalCTFile,** and **RemoteCTFile** parameters.
- In individual parameters, it is possible to specify two conversion tables; one on the local side, and one on the remote side. This way you can have a standard character set to be used for transmission, without having a conversion table between every two possible character sets.
- The local conversion table is specified with the **LocalCTFile** parameter in templates, and the **lct** parameter at the command line. Similarly, the remote conversion table is specified with the **RemoteCTFile** parameter in templates, and the **rct** parameter at the command line.
- The **lct** and **rct** parameters are capped at a 16 character max for purposes of shrinking the number of bytes sent per transfer. The directory where PSA is installed and running from is the directory that will be searched for the conversion table.
- Always replace a 2-digit hexadecimal number with a 2-digit hexadecimal number. If the table is invalid translation will not be performed. Remember, the table is comprised of two sections that are 16 lines each. (each character is a 2 digit hexadecimal number). Thus the entire file should have 32 lines across and 32 lines down. If it has anything else it will not work!
- For all transfers, if the file is outgoing (a send), then the top half of the conversion table will be used. If the file is incoming, (a receive), then the bottom half of the conversion table will be used. For example, in a **cfsend**, if both the **LocalCTFile** and **RemoteCTFile** parameters are used, then the top half of the **LocalCTFile** will be used on

the local side, and the bottom half of the **RemoteCTFile** will be used on the remote end. The reverse is true for **cfrecv**.

- Remember which table translates for sends and which translates for receives. TIBCO recommends during editing you place a few lines between the two tables to help remember which is which.

Appendix B – Post Processing Actions

Post Processing Actions (PPA) gives you the capability of executing up to 4 commands when a file transfer completes either successfully or unsuccessfully. This section explains how to configure and use Post Processing Actions.

Topics

PPA Setup

Post Processing Actions (PPA) gives you the capability of executing up to 4 commands when a file transfer completes either successfully or unsuccessfully. The PPA parameters are as follows:

```
Post_Action1 or ppa1
Post_Action2 or ppa2
Post_Action3 or ppa3
Post_Action4 or ppa4
```

Post Processing command format is as follows, notice how each section of the command is separated with a comma (,):

```
Post_Action1=”S|F,L|R,COMMAND,<command data>”
```

The first section tells us if we want the action to be run when the file transfer request is either successful or fails:

```
S | F – Success or Failure
```

The second section of the command tells us if the action is supposed to run on the local machine (Initiator) or the remote machine (Responder):

```
L | R – Local or Remote
```

The third section tells us what the action is. If the remote system is a mainframe, then CALLJCL, CALLPGM and SUBMIT are also supported, otherwise you would use the parameter COMMAND.

```
COMMAND | CALLJCL |CALLPGM | SUBMIT
```

The last section of the command is the command data section and would hold the absolute path and file name of command and any parameters to be passed. This is limited to 256 bytes.

No spaces are allowed in the Post Processing Action command. If, at the command line, the **ppa** parameter is used, the

information will go into the next undefined post processing action slot.

```
Example.ppa1:”S,L,COMMAND,batchjob.exe”
```

PPA Substitutable Table Example

PSA supports Substitutable Parameters to allow users to take full advantage of the 256 character maximum on the command data, and to allow users to not have to copy the filename from the LocalFileName or RemoteFileName parameters. Note that we do not support standard MFT Tokens within PPA, because they are relatively long and the substitutable parameters conserve as many bytes as possible within the PPA action data field.
The PPA Substitutable fields use the percent character (%) as the escape character instead of the $ that tokens use. Below is a list of the substitutable parameters that are supported.

For our example, assume that we have a file called:
C:\a\b\c\d\config.txt

Parameter	Description	Resolved Name
%DIR	Directory without the file name or drive	a\b\c\d
%DRIVE	Drive Name	C
%FILE	The file name without the directory	config.txt
%GDATE	Gregorian Date (yymmdd)	080929
%GDATEC	Gregorian Date with Century (ccyymmdd)	20080929
%HDIR	The high level directory	a
%HLQ	High level qualifier of file	config
%JDATE	Julian Date (YYDDD)	08273
%JDATEC	Julian Date with Century (CCYYDDD)	2008273
%LFILE	File name with directory	C:\a\b\c\d\config.txt
%LLQ	Low Level Qualifier of file (data after last period(.))	txt
%LUSER	Local User Id	
%NODRIVE	File name without Drive	a\b\c\d\config.txt
%NOHDIR	Directory name w/o high level directory	b\c\d

Parameter	Description	Resolved Name
%NOSDIR	Directory name without lowest directory	a\b\c
%PROC	Process Name	ABC123
%RUSER	Remote User Id	
%SDIR	The lowest level directory	d
%TIME	Time (hhmmss)	165030
%TRN	Transaction Number	I929800001
%UDATA	User Data	USRDATAABC123

Note that there can be multiple PPA parameters within a single PPA data field. Each Substitutable parameter must be processed one at a time before going onto the next byte of PPA data.

Note that some fields do not make sense such as %DRIVE in a UNIX environment. If a field does not make sense in the environment where PPA is being used, then the substitutable data is the text in the name of the parameter without the % sign. If UNIX detects the %DRIVE parameter, then the value DRIVE should be used as substitution. Similarly, %PROC becomes PROC and %UDATA becomes UDATA if not interacting with a z/OS system.

Appendix C – File Name Tokens

PSA supports the use of file tokens in the local and remote file name fields or a transaction. To rename files transferred. This section lists and describes the file name tokens available.

Topics

File Name Tokens Available

PSA supports the use of file tokens in the local and remote file name fields or a transaction. When the file is transferred using tokens they will be translated to a new value within the file name. When sending or receiving a file with PSA you can use any of the following supported file tokens in the name (Tokens use the following format $(*token*)):

Token Name	Definition	Generated Value
$(Date)	Local Date	YYYYMMDD
$(Date1)	Local Date	YYMMDD
$(Date2)	Local Date	MMDDYY
$(Date3)	Local Date	DDMMYY
$(LocalFile)	Local File	the full path and local file name
$(LocalFileName)	Local File Name	the name of the Local File
$(LocalTransactionNumber)	Local Transaction Number	the local transaction number
$(LocalUserid)	Local User Id	id of the local PSA user
$(RemoteFile)	Remote File	the full path and remote file name
$(RemoteFileName)	Remote File Name	the name of the remote file
$(RemoteTransactionNumber)	Remote Transaction Number	the remote transaction number
$(RemoteUserid)	Remote User Id	id of the remote PSA user
$(Time)	Local Time	HHMMSSMSS
$(Time1)	Local Time	HHMMSS
$(Time2)	Local Time	HHMMSST

Examples

These examples show the file names with tokens and how they are resolved based on the following client file name sitting on a UNIX platform and being sent to a Windows platform:

/home/testfile.txt

Remote File Name	Resolved File Name
C:\testfile.$(Date)	C:\testfile.20051231
C:\testfile.$(Date1)	C:\testfile.031231
C:\testfile.$(Date2)	C:\testfile.123103
C:\testfile.$(Date3)	C:\testfile.311203
C:\$(LocalFile)	C:\home\testfile.txt
C:\$(LocalFileName)	C:\testfile.txt
C:\$serverfile.(LocalTransactionNumber).txt	C:\serverfile.I309000013.txt
C:\$(LocalUserid)\testfile.txt	C:\jdoe\testfile.txt
C:\testfile.$(Time)	C:\testfile.245959999
C:\testfile.$(Time1)	C:\testfile.245959
C:\testfile.$(Time2)	C:\testfile.2459599

If you were initiating a file receive from the UNIX platform to a Windows platform you could use:

Local File Name	Resolved File Name
C:\$(RemoteFile)	C:\home\testfile.txt
C:\$(RemoteFileName)	C:\testfile.txt
C:\$serverfile.(RemoteTransactionNumber).txt	C:\serverfile.R308000025.txt
C:\$(RemoteUserid)\testfile.txt	C:\jsmith\testfile.txt

Note: When transfers are initiated from a Macintosh OS x or HP Nonstop system tokens need to be contained in single quotation marks in addition to double quotes around the path. Below is an example:

RemoteFileName:"c:\PSAfiles\FromAMac\"'$(LocalFileName)'

Appendix D - DNI

Directory Named Initiation (DNI) allows you to detect the existence of files that have been placed within a directory and/or sub directories and automatically transfer those files to one or more targeted MFT Platform Server™ remote systems. This section explains how to use DNI with TIBCO MFT Platform Server™ Agent.

Topics

DNI Overview

Directory Named Initiation (DNI) allows you to detect the existence of files that have been placed within a directory and/or sub directories and automatically transfer those files to one or more targeted MFT Platform Server™ remote systems.

When you set up a DNI transfer, it will scan a pre-defined local or directory at a user-defined interval. It will return and save a list of all files in that directory. Any files that have changed between the scans are eligible to be transferred. This is performed in this manner as some systems such as UNIX platforms do not have a standardized way of locking files. This scan allows MFT Platform Server™ Agent to tell whether the file is in use or not. When a DNI file transfer is complete, DNI allows you to delete the original file, move it to a new directory, or leave the file where it is.

Transfer requests that can be processed using DNI are:

DNI Send

DNI Send reads the directory/s defined and executes a command when it detects a file exists within the directory/s and sends the files to a remote system.

DNI processing is done using a Perl script called **dni**. As such, in order to use DNI your system must have a version of Perl installed. The Perl program directory should be defined in the environment variables on your system. If you do not have Perl installed on your computer, it can be downloaded for free from web site: www.perl.org.

Note: The MFT Platform Server™ Agent DNI perl script will run on both Windows and UNIX platforms and as such we provide DNI templates for both the Windows and UNIX platform. In this manual we will only discuss the DNI templates for the UNIX

platform. The DNI templates for the Windows platform do not work with or complement the MFT Platform Server™ Agent for Windows DNI feature in any way. For information using MFT Platform Server™ for Windows DNI, please refer to the MFT Platform Server™ for Windows documentation.

There are many uses for DNI. Some of the more obvious include:

1. To copy entire directory structures from one system to another - You can tell MFT Platform Server™ Agent DNI to copy all files from SystemA to SystemB and create the same directory structure that exists on SystemA on SystemB using the LocalFileName token. In this example, the DNI template on SystemA can be configured with the PPA SuccessAction set to "leave", the write mode set to CRN and SubDirectory parameter set to Yes for the directory structure to be created. *Note: It is important that the DNI is only from SystemA to SystemB. A second DNI configured similarly to copy from SystemB to SystemA could create an infinite loop.*
2. Copy files to another MFT Platform Server™ Agent, MFT Platform Server™, or MFT Internet Server computer. As DNI detects that a file on SystemA has changed, DNI will send the file to SystemB. In this case when the transfer has completed, you can move the file to a backup directory, or delete the file.
3. Execute a system command based on the existence of a file. As DNI detects that a file on SystemA has changed, DNI will execute a pre-defined command. This command can include information on the local file that was the source of the DNI request. It is up to the DNI Administrator to define the actual command that is executed.

Installing the DNI Program

DNI is distributed as a tar file called **dni.tar**. It can be found in the **(InstallDirectory)/dni** directory. To extract the files, execute the following tar command on the UNIX platform

```
tar xvf dni.tar
```

Upon successful execution of this command, you will find the following five files:

```
dni  - The MFT Platform Server DNI perl script program
dnitemplate.send  - The DNI template for a DNI Send
dnitemplate.recv  - The DNI template for a DNI Receive (Will
  not be discussed as it is not supported for MFT PSA)
dnitemplate.wsend - For Windows DNI Send (Will not be
  discussed)
dnitemplate.wrecv - For Windows DNI Receive (Will not be
  discussed)
```

Note a few things about the dni template files:

1. Any data in a line after the # is considered a comment.
2. If a line consists of all blanks, then the line is ignored.
3. All parameters are in the format of xxx: yyy where xxx is the parameter name, and yyy is the parameter value.
4. Parameter names and values are not case sensitive, although we suggest leaving them mixed case for easier reading.
5. The LocalFileName (lf) and RemoteFileName (rf) parameters must be specified in the DNI template. If you are using a transfer template within your DNI template the LocalFileName (lf) and RemoteFileName (rf) parameters defined in the DNI template will override these values defined in a transfer template.

DNI Help Screens

There are four help screens that give information on how the MFT Platform Server™ Agent DNI script can be used. This can be executed by entering the following commands:

perl dni help	General help information
perl dni help template	Template configuration information
perl dni help tokens	Substitutable Token information
perl dni help encrypt	Creates an encryption $(Password) token

Running the DNI Program

The following command is used to execute the DNI perl script:

`perl dni t:dnitemplatename {optional parameters}`

If the dnitemplate is not in the current working directory, then you must define the fully qualified template name so that DNI can read the contents of the template file.

DNI Script Optional Parameters:

Parameter	Description
cold	The “Cold” option bypasses the reading of the “Leave” file that contains the list of files already transferred that is saved when the SuccessAction and/or FailureAction is set to “leave”. As defined in the warm option description. As such, DNI will transfer any file in the scan directory when the script is started. This option should be used with great care.
warm	This parameter is only used when a DNI SuccessAction or FailureAction is defined as “Leave”. Due to this DNI will keep track of files that have been transferred and store the information in a file. The warm option tells DNI to read the “Leave” file and at start up and process the files normally as if MFT Platform Server™ Agent did not come down. This makes sure that a file is not transferred multiple times by mistake. (This is the default when no option is set.)
hot	When defined, DNI will add all files detected on the first directory scan to the “Leave” file. Therefore the contents of the directory when DNI is started will not be transferred. Note that a Hot start creates the “Leave” file with the contents of the directory after the first scan. If DNI Receive is unable to get this list from the remote system due to a network error, then Hot start processing will not be performed.
debugon	Turns on DNI debugging. This parameter overrides the “debug” parameter in the MFT Platform Server™

Parameter	Description
	Agent template. This parameter should only be used when instructed to do so by TIBCO Technical Support.
debugoff	Turns off DNI debugging. This parameter overrides the “debug” parameter in the MFT Platform Server™ Agent template. This parameter should only be used when instructed to do so by TIBCO Technical Support.

DNI Template Parameters

Each DNI template is broken up into sections. The first one defining the directory to be scanned and what transfer command is to be executed when a file is found in that directory.

Below we have placed a figure of the contents in the **dnitemplate.send** template so as to discuss each section of the file:

```
# Define DNI transfer information:
#  LocalDirectory      Local Directory to scan for files
#  DNICommand          Command to execute when file is detected
#  SubDirectory        Defines if subdirectory scanning is on (Yes) or off (No)

# Valid DNI Tokens for DNICommand, SuccessFile and FailureFile
#    $(YYYYMMDD)      ==> Current Gregorian date in format YYYYMMDD
#    $(YYMMDD)        ==> Current Gregorian date in format YYMMDD
#    $(YYYYDDD)       ==> Current Julian date in format YYYYDDD
#    $(YYDDD)         ==> Current Julian date in format YYDDD
#    $(YYYY)          ==> Current Year in format YYYY
#    $(MM)            ==> Current Month of Year in format MM
#    $(DD)            ==> Current Day of Month in format DD
#    $(HHMMSS)        ==> Current Time in format HHMMSS
#    $(HH24)          ==> Current Hours in 24 hour format HH
#    $(MI)            ==> Current Minutes in format MM
#    $(SS)            ==> Current Minutes in format SS
#    $(LocalFile)     ==> Local File name without Path Name
#    $(LocalFileBase) ==> Local File name without Extension and Path Name
#    $(LocalFileExt)  ==> Local File name extension (after last .)
#    $(LocalFileName) ==> Local File name with Path Name
#    $(LocalFilePath) ==> Local File Path Name

#    $(LocalHLQ)      ==> Local File Name High Level Qualifier
#    $(LocalNoHLQ)    ==> Local File Name without High Level Qualifier
#    $(SubDir)        ==> Subdirectory name without leading /
LocalDirectory:         /DNI/Files/
DNICommand:             cfsend t:cftemplate lf:$(LocalFileName)
rf:/DNI/Target/$
(LocalFile) sm:y
SubDirectory:           N
```

Figure 5

DNI Tokens

These can be used for **DNICommand, SuccessFile** and **FailureFile** parameter values:

Token Name	Description
$(YYYYMMDD)	Current date in the format YYYYMMDD (Gregorian date)
$(YYMMDD)	Current date in the format YYMMDD (Gregorian Date)
$(YYYYDDD)	Current date in the format YYYYDDD (Julian date)
$(YYDDD)	Current date in the format YYDDD (Julian date)
$(YYYY)	Current Year
$(MM)	Current Month
$(DD)	Current Day within Month
$(HHMMSS)	Current Time in format HHMMSS (24 hour format)
$(HH24)	Current Hour in 24 hour format
$(MI)	Current Minutes within the current hour
$(SS)	Current Seconds within the current minute
$(LocalFile)	File name without the directory. DNI Send only.
$(LocalFileName)	Fully qualified file name (with directory). DNI Send only.
$(LocalFileBase)	File name without the extension (file name before last period(.)). DNI Send only.
$(LocalFileExt)	File name extension (file name after last period(.)). DNI Send only.
$(LocalFilePath)	Path name of the file. DNI Send only.
$(LocalHLQ)	Local File Name High Level Qualifier
$(LocalNoHLQ)	Local File Name without the High Level Qualifier
$(RemoteFile)	File name without the directory. DNI Receive only.
$(RemoteFileName)	Fully qualified file name (with directory). DNI Receive only.
$(RemoteFileBase)	File name without the extension (file name before last period(.)). DNI Receive only.

$(RemoteFileExt)	File name extension (file name after last period(.)). DNI Receive only.
$(RemoteFilePath)	Path name of the file. DNI Receive only.
$(RemoteHLQ)	Remote File Name High Level Qualifier
$(RemoteNoHLQ)	Remote File Name without the High Level Qualifier
$(SubDir)	Subdirectory of the file (directory name after removing the LocalDirectory. This template does not contain a leading slash (/ or \ depending on whether you are running on UNIX or Windows). This token is unique and can be used in a special way. Please read the section Special SubDir Token Use. *(Note: Not available for MFT Platform Server Agent)*
$(TIME) **Used for Process Name Only**	Current Time in format HHMMSS

DNI Transfer Information Parameters

Parameter	Description
LocalDirectory	Used in the DNI Send template. Defines the local directory that DNI will scan to see if there are any files. DNI will read all normal files from this directory. If the SubDirectory parameter is defined as Yes, then DNI will also read normal files from all subdirectories. DNI Tokens can be used to customize this parameter.
RemoteDirectory	Used in the DNI Receive template. It defines the remote directory that DNI will scan to see if there are any files. DNI will send a command to the remote system to read all normal files in this directory. If the sub directory parameter is defined as Yes, then DNI will read all normal files from all subdirectories.
DNICommand	Defines the command that DNI will execute when a file is detected. Note that the default is to execute a CFSEND for DNI Send and cfrecv for DNI Receive command. Note that the default DNICommand defines in the template has DNI

Parameter	Description
	tokens such as $(LocalFileName) and $(LocalFile). These tokens are defined in the section “DNI Tokens”. Note that this command can be customized to add as many parameters as needed. An alternate way is to define all of the file transfer commands within the template.
SubDirectory	Defines whether subdirectories are scanned for files. The default value of “No” indicates that subdirectories will not be scanned for files, while specifying “Yes” will cause DNI to scan all subdirectories for files.

Post Processing Parameters

The only disadvantage of SuccessAction or FailureAction “Leave” is that it keeps a copy of the directory entry for each file that is successfully transferred. If you transfer many different files (over 10,000) then the memory requirements of the in-storage copy of the LeaveFile can get very high.

```
# Post Processing Actions:
#  SuccessAction        Valid SuccessAction: leave, move, delete
#  SuccessFile          Required if SuccessAction:Move - Defines Target File Name for
move
#  FailureAction        Valid values: leave, move
#  FailureFile          Required if FailureAction:Move - Defines Target File Name for
move
#  NetworkErrorAction   Valid values: leave, move
#  NetworkErrorFile     Required if NetworkErrorAction:Move - Defines Target File Name
for move
SuccessAction:          move
SuccessFile:            /DNI/Success/$(LocalFile).$(YYYYMMDD).$(HHMMSS)
FailureAction:          leave
FailureFile:            /DNI/Failure/$(LocalFile).$(YYYYDDD).$(HHMMSS)
NetworkErrorAction:     leave
NetworkErrorFile:       /DNI/Failure/$(LocalFile).$(YYYYDDD).$(HHMMSS)
```

Figure 6

Parameter	Description
SuccessAction	Defines what is done with the file when the DNICommand is executed successfully. Valid values: leave | move | delete
SuccessFailure	This parameter is only used when

Parameter	Description
	SucessAction is defined as “move”. It defines the fully qualified name of the target file for the move command executed as a result of a successful DNICommand execution. You can use DNI tokens to customize the SuccessFile name. DNI Tokens can be used to customize this parameter.
FailureAction	Defines what is done with the file when the DNICommand fails. Valid values: leave | move
FailureFile	This parameter is only used when FailureAction is defined as “move”. It defines the fully qualified name of the target file for the move command executed as a result of an unsuccessful DNICommand execution. You can use DNI templates to customize the FailureFile name. DNI Tokens can be used to customize this parameter.
NetworkErrorAction	Defines what is done with the file when the DNICommand fails due to a network error. Valid values: leave | move
NetworkFile	This parameter is only used when NetworkErrorAction is defined as “move”. It defines the fully qualified name of the target file for the move command executed as a result of an unsuccessful DNICommand execution. You can use DNI templates to customize the NetworkFile name.

DNI Scan Scheduling

This section allows you to define the times of each day that DNI will scan the LocalDirectory (DNI Send). The days of the week must be defined using the English spelling.

```
# DNI Scan Schedule:
# Each day can be defined with a Start and Stop time in the format:
#    HHMM:hhmm
# where HHMM is the start time from 0000 to 2359
#       hhmm is the stop  time from 0000 to 2359
# These times determine when DNI scans the LocalDirectory for files.
# To turn off scanning for a day, specify 0000:0000
# Note that the English spelling of these names is required.

Sunday:                 0000:2359
Monday:                 0000:2359
Tuesday:                0000:2359
Wednesday:              0000:2359
Thursday:               0000:2359
Friday:                 0000:2359
Saturday:               0000:2359
```

Figure 7

Miscellaneous Parameters

```
# Miscellaneous parameters:
#  ScanInterval         Defines the number of minutes between directory
scans
#  LeaveFileName        Defines Unique name of file when
SuccessAction:Leave

#  Debug                Defines whether DNI debugging is on (Yes) or off
(No)
ScanInterval:           5
LeaveFileName:          ./leavefile.001
Debug:                  No
```

Figure 8

Parameter	Description
ScanInterval	Defines the number of minutes between scans of the LocalDirectory. The default value is 5 minutes. The processing of this value depends on whether DNI Send or DNI Receive is being executed: **DNI Send** waits this interval between scans of the directory. Additionally, DNI Send will make sure that a file has not been updated within this amount of time before it will process the DNICommand for that file. If a file has been changed in that interval, then DNI will not process the file. Since the file cannot be locked, this is how MFT Platform Server™ Agent can tell whether the file is in use or not. For testing, you may want to use the minimum value of

Parameter	Description
	1. For production, unless the timeliness of the files is critical, a higher value is usually appropriate.
LeaveFileName	Defines the unique name of the leave file where DNI saves information about the DNICommand requests that have completed successfully. This file is only required when SuccessAction is defined as “leave”. If you have multiple scripts running with SuccessAction “leave” then this file MUST be unique across all executions of DNI. This file is used to save the LeaveFile information across different executions of DNI. If DNI comes down, or the system is booted, then DNI will read the information from the LeaveFile so that the DNICommand is not executed multiple times.
Debug	Defines whether DNI debugging is turned on. This parameter should only be set when instructed to do so by TIBCO Technical Support.
TempDirectory	It is used only by DNI Receive. It defines the name of a directory where DNI will save temporary files. DNI must be able to create and read files in this directory. This parameter defaults to the current working directory. *(Note: Not available for MFT Platform Server Agent)*

DNI Encrypting Passwords

Within a transfer template a userid and password can be defined to be used when connecting to a remote system. However this would leave the password in the file in clear text. You could define the userid and password within the DNICommand parameter but this again leaves the password in clear text. MFT Platform Server Agent includes an executable that will encrypt the password that should be used with a dnitemplate by using the token $(Password). In our example below we have defined a command using the token:

```
DNICommand:  java -cp psa.jar cfsend t:cftemplate
lf:"$(LocalFileName)"
rf:"/DNI/Target/$(LocalFile)" uid:JohnDoe
pwd:$(Password) sm:y
```

Once you have defined the $(Password) token for the password field and your dnitemplate is ready to be used you would run the following perl command:

```
perl dni encrypt t:dnitemplate
```

You will be prompted to type in the userid's password and asked to confirm the password. Once the password is encrypted you will see a similar message to the one below:

```
Adding line 74 to template: EncryptedPassword:
OVCu11P9wxaeOo2Rr2OaKXMYpzXukAzXqOvn/Pu7qaY=
```

Now your dnitemplate has an encrypted password associated with it. You would run this dnitemplate the same as you would any other dnitemplate you have. For more information on encrypting a dnitemplate password you can view the encrypt help screen with the following perl command:

```
perl dni help encrypt
```