



bTrade TDAccess CLI and API User Guide

Release 2.2

© 2003 bTrade
All rights reserved.
Document 2003.UG006.03
Printed in the USA.

CONFIDENTIAL. All rights reserved. This document, including any writing, drawings, notes, or verbal representation made or shown in the course of this communication is confidential and proprietary to bTrade. No part of the materials included in this communication should be: a) reproduced; b) published in any form by any means, electronic or mechanical, including photocopy or information storage or retrieval system; or c) disclosed to third parties, without the express written authorization of bTrade.

Contents

Contents	3
1 Purpose.....	4
1.1 Other References.....	4
2 Command File Usage.....	5
3 Keywords	7
3.1 Command Line Only.....	7
3.1.1 Logging.....	8
3.2 Specifying Tasks to Perform.....	9
3.3 Failure Notification.....	12
3.4 Firewall Parameters	13
3.5 Database Parameters	14
3.6 Email Message Handling.....	15
3.7 EDI-INT (AS1/AS2) Message Handling.....	16
3.8 File Viewing Option (Windows TDIImage Only).....	17
3.9 Overriding Network Parameters	17
3.10 File Transfer Execution.....	22
3.11 Overriding the Default Parameters Used to Send Files	25
3.12 Compression, Decompression and Security	27
3.13 Overriding the Default Parameters Used to Receive Files	37
3.14 Send/Receive Pre/Post Processing While Creating a Transfer.....	38
3.14.1 PRE_SEND, POST_SEND, PRE_RECEIVE, and POST_RECEIVE	
Keywords.....	38
3.15 Handling Dial Connections (Windows platforms only)	40
3.16 Specifying Auto-Retry Options	41
3.17 Adding/Modifying Entries in a Trading Partner Address Book.....	41
4 Canceling a Running Transfer (Command Line Only).....	44
5 API Interfaces	45
5.1 TDAccess API	45
5.1.1 Extracting the API.....	45
5.1.2 Using the API.....	45
5.2 TDCompress API.....	47
5.2.1 Using the Command-Line API (COMPRESS and DECOMP)	48
5.2.2 Using the Interactive API (compprog and dcmpprog).....	49
Appendix A. Command File Example	52

1 Purpose

This document contains actual working examples on how to use keywords associated with the bTrade TDAccess Command Line Interface (CLI). Keywords are parameters used as a string in the command line scripts when sending or receiving files from various network styles.

The keywords are also the functions that can be called via the bTrade TDAccess Application Programming Interfaces (APIs).

The document is broken into different sections, including

- Command File Usage
- Keyword Descriptions
- API Usage

1.1 Other References

Please refer to the following documentation and web site for information about error codes and terminology used.

- bTrade TDAccess API Error Messages and Codes
- bTrade Glossary of Terminology (more up to the minute and general terminology can be found here: http://www.btrade.com/resource_center/glossary/G0.asp)

2 Command File Usage

A command file is specified using the CMDFILE command-line keyword as the bTrade TDAccess program is being invoked.

The following are the rules of any command file:

1. Text comments to the right of any '#' character is ignored unless the # occurs with text offset by single or double quotes.
2. Blank lines are ignored.
3. Command-line and command-file parameters can be delimited using single or double quotes, parentheses, or square or curly braces. Sub-expressions can be delimited within main expressions using a different delimiter. For example, `TRANSFER=(name='my transfer'... OTHER_COMP_PARMS='parm1 parm2'...)`
4. Spaces are used only to separate keyword/value pairs and are otherwise ignored unless they are within a delimited expression. For example: `TRANSFER= "My transfer"` is the same as `TRANSFER="My Transfer"`, but `TRANSFER = "My Transfer"` is illegal (`TRANSFER=` is the keyword – no spaces allowed within the keyword itself), and `TRANSFER= My Transfer` is also illegal (it is saying to use transfer-name "My", not "My Transfer") because there are no quotes around the transfer-name.
5. The end-of-line has no special significance. This means that you can put all your keywords on one line, or spread them out across multiple lines.
6. The keywords are NOT case-sensitive, but the values may be, depending on the server with which you are communicating. For example
transfer=
and
Transfer=
are the same as
TRANSFER=
but the server may not agree that
LOGINUSERDID=user1
is the same as
LOGINUSERDID=USER1
7. Up to 200 transfers may be CREATED within the command-file. Each transfer created is added to the list of transfers to be run.
8. Up to 200 EXISTING transfers may be specified in the command-file. Each specified existing transfer is added to the list of transfers to be run.

9. An unlimited number of trading partners may be added to your Trading Partner Address book using the TPBOOK keyword.

An example command file is located in the appendix.

3 Keywords

Keywords are functions that can be called either via the command line or via a command file. Keywords provide the following capabilities during a run of TDAccess:

- override default settings
- change the way information is sent/received
- specify tasks to be executed

These Keywords are also the functions exposed in the API.

3.1 Command Line Only

The following keywords appear on the command line only as the bTrade TDAccess program is being invoked.

Keyword Usage	
Keyword	Usage
CMDFILE=	<p>Specify the location of a TDAccess command-file to be used.</p> <p>The command-file will completely control the work done during the program run. When running the command-line version of TDAccess, if no command-file is specified via the CMDFILE=keyword, then the program will assume the work to be done has been previously set up using the GUI or by hand via an editor and is specified in the tdclient.ini and exfer.ini files. The CMDFILE contains all the specific keywords required for a send, receive or query.</p> <p>Example: tdclientc cmdfile=send.cmd reset</p>
DISABLE_DIALER (Windows only)	<p>Instructs TDAccess to turn off all logic pertaining to Dial-Up Networking: Useful if your system does not have the Windows DLLs required by TDAccess for establishing dial-up connections. This parameter can also be modified in the tdclient.ini file.</p>
HELP	<p>Displays on-line usage guidelines:</p> <p>Example: tdclientc help more</p>
IEBASE	<p>Executes "iebase" functionality: reads and parses an "IBM EXPEDITE-style" file, basein.msg, and creates transfers to be executed. Note that either the CMDFILE= or the IEBASE keywords may be used, but not both together during a given program run. Note this keyword requires no value.</p>

INIPATH=	<p>Override location of TDAccess 'root' directory.</p> <p>This is the directory containing the tdclient.ini and exfer.ini files as well as the various sub-directories required (e.g. security, runtime, temp and so on). If the INIPATH= keyword is not specified, then the program assumes the current working directory is the TDAccess root. The INIPATH can be used to run various network styles by creating specific network configurations using a single instance of TDAccess.</p> <p>Example: tdclientc inipath=c:\users\rh1\tdclient.ini</p>
MODE= BATCH or GUI	<p>Applies to the GUI version of TDAccess only, specifies that the GUI program is to execute in command-line mode. MODE=GUI is the default, causing the GUI to execute in GUI mode.</p>
RESET	<p>Instructs TDAccess to ignore previously failed transfers, which would otherwise attempt to restart on the next execution of the application.</p>
RESPLOG=	<p>Specifies a filename to receive the "response log" generated when sending or receiving files. The response log is used primarily for integrating the application into the user's application. It provides information on all send and receive activity during the program run.</p>
VALIDATE_TRANSFERS_ONLY	<p>Causes TDAccess to validate Stored Transfers (transfers located in the exfer.ini file), report on their validity, and exit. This parameter is used as a verification process.</p> <p>Example: tdclientc validate_transfers_only reset</p>

Notes:

1. Either the CMDFILE or the IEBASE keyword may be used, but not both together during a given program run. Use of the keyword RESET is required unless the previously initiated transfer is to execute first.
 2. The HELP keyword requires no value. It is also recommended using a page separator when using this command, such as | more.
-

3.1.1 Logging

The following keywords are used to specify and change logging parameters, and can only be used on the command line.

For all of these keywords, the allowed values are 0 (off) to 6 (maximum). In addition, values of 'N' (off) and 'Y' (corresponds to numeric value 3) are supported. Finally, to

enable 'flush after every write' logging, prefix the value with a minus sign. For example, LOG_XFER=-6 means turn on maximum logging and flush after every write.

Keyword Usage	
Keyword	Usage
LOG_EASYACC=	Controls the Overall program run log. Produces file ea2k.log in the TDClient root directory, containing details of the entire program run.
LOG_INI=	Controls the 'Ini' file processing log. Produces file eaini.log, containing details of the start-up processing of the tdclient.ini file.
LOG_XFER=	Controls the Transfer processing log. Produces file temp/eaxfer.log, containing details of the last send or receive transfer, query_list, or audit report request.
LOG_FTP=	Controls the Communications session log. Produces file temp/eacomm.log, containing details of the last FTP, AS1 (SMTP/POP + S/MIME), or AS2 (HTTP + S/MIME) communications session.
LOG_SOCKET=	Controls the SSL session log. Written into the temp/eacomm.log (in addition to the Communications session text), contains very low level details of the SSL encrypted communications session when using SSL with FTP or HTTP.
LOG_MEM=	Controls the Memory usage log. Produces the file eamem.log in the TDClient root directory, containing details of memory allocation/deallocation during the program run.

Notes:

1. When set to the maximum log level (6) a very large amount (megabytes) of logged text can result. This will slow down program operation.
2. Setting the 'flush after every write' option will also slow down program operation (sometimes dramatically). So, while logging is useful to isolate configuration or operational problems, you probably do not want to run with extensive logging much of the time.

3.2 Specifying Tasks to Perform

The following keywords are used either at the command line or within a command file.

Keyword Usage	
Keyword	Usage
AUDIT_END_DATE=	Used in conjunction with RECEIVE_AUDIT_LOGS to specify the interval of time the audit report should cover.

AUDIT_FILE=	Specifies the qualified file name of the file to receive the audit logs. If not present, the audit logs are written to the default file, audit.log, in the TDAccess 'temp' directory.
AUDIT_START_DATE=	Used in conjunction with RECEIVE_AUDIT_LOGS to specify the interval of time the audit report should cover.
EXPORT_CERTS_TO_TP=	(Trading partner address book entry) Specifies that a PKCS-7 'certs-only' message is to be constructed containing the public certificate for the configured user, and then transmitted to the specified trading partner as specified in the trading partner address book. The configured user is identified by the EDINAME entry in the [IDENTIFY] section of the tdclient.ini file.
FILTER_EXPORTED_CERTS=Y or N	Specifies whether the exported certificate will be filtered (base64 encoded) or not. Applies only in conjunction with EXPORT_CERTS_TO_FILE
GENKEYS	Instructs TDAccess to generate a public/private key pair, to bundle these into a certificate request, and to create and execute a transfer to send the certificate request to a location designated in the SECURITY section of your tdclient.ini file.
HFS_DIR=	Specifies work-file directory; required for MVS.
NETWORK=	Specifies an existing network is to be used as the basis for the communications session. The text must match one of the networks already defined in your tdclient.ini file. This keyword tells TDAccess to use the specified network in the tdclient.ini file as the current network. Example: tdclientc network=IGN
QUERY_FILE=	Specifies a qualified file name of the file to receive the server file list when executing this keyword. If not present, the file list is written to the default file, list.fil, in the TDAccess 'temp' directory. To use this parameter, the query_list parameter MUST be specified first. Example: tdclientc query_list query_file=c:\temp.dat reset
QUERY_LIST	Instructs TDAccess to create and execute a transfer to receive a list of available files from the current server.

<p>QUERY_STATUS=</p>	<p>This keyword specifies that action QUERY_LIST should return a list of files with the specified status only. The allowed status values are:</p> <p style="padding-left: 40px;">AVAILABLE - files which have not yet been received</p> <p style="padding-left: 40px;">RECEIVED - files which have been already received</p> <p style="padding-left: 40px;">DELETED - files which have been deleted</p> <p>If not present, then the file list will contain entries for any status (just as before the introduction of the QUERY_STATUS keyword. To use this parameter, the query_list parameter MUST be specified first.)</p> <p>Example: tdclientc query_list query_status=available reset</p>
<p>SAVE</p>	<p>Instructs TDAccess to save the network data you specify on the command-line or in the command-file in the tdclient.ini file, causing the data to be permanently in effect until you change it.</p> <p>In the default case, the network data only applies for the duration of the current program run. This is not true for transfers that are created on the command-line or command-file; these are written to the exfer.ini file and are automatically available to you in subsequent runs of the program.</p>
<p>SAVE_ONLY</p>	<p>Acts like the SAVE keyword, except the program exits after saving the specified network data in the tdclient.ini file.</p>
<p>RECEIVE_AUDIT_LOGS</p>	<p>Instructs TDAccess to create and execute a transfer to receive an audit report from the current server of files sent and received.</p>
<p>RECEIVE_RUNTIMES</p>	<p>Instructs TDAccess to create and execute a transfer to receive your security runtimes from a location designated in the SECURITY section of your tdclient.ini file. The runtimes file is then automatically installed giving TDAccess access to the public keys of your trading partners.</p>
<p>RETRY = Y or N</p>	<p>Enables or disables Auto-Retry.</p>
<p>TPBOOK=</p>	<p>Specifies that an entry in the Trading Partner Address Book is to be added or edited.</p> <p>Here is an example command-file entry to create/modify a Trading Partner:</p> <p>Example: tdclientc tpbook=(name=MyPartner network1=BTrade mailbox1=MyPartnersMailbox) reset</p> <p>Note: The previously executed transfer will execute using this keyword.</p>

<p>TRANSFER=</p>	<p>Used to specify an existing transfer, or create a new transfer, and add it to the list of transfers to be run.</p> <p>To specify an existing transfer, use the form:</p> <p>TRANSFER=name</p> <p>or</p> <p>TRANSFER="name"</p> <p>(quotes are required if 'name' contains spaces).</p> <p>To create a new transfer, use the form:</p> <p>TRANSFER=(keyword=value keyword=value... keyword=value)</p> <p>where all the various keywords that define the transfer appear within the (required) parentheses.</p> <p>Example: tdclientc "transfer=(name=test send=c:\test.dat)" reset</p> <p>(In this example, the name of the transfer is test, the transfer is sending test.dat).</p>
<p>PASSLOC=</p>	<p>Specifies the passphrase location for encrypting keys and provides the user with the ability to specify the location of a security token. If not specified, then a default value is used.</p>

Note:

Any overrides to this network's data (via the keywords above) will only apply for a single program run unless the SAVE keyword is also used.

No other transfers will be executed during the program run if the "queryList" keyword is specified -- it supersedes the execution of all send/receive transfers.

No other transfers will be executed during the program run if the "receive_runtimes" keyword is specified -- it supersedes the execution of all send/receive transfers.

3.3 Failure Notification

To configure the Failure Notification options from the command-line, the following keywords are provided.

Keyword Usage	
Keyword	Usage
NOTIFY_AFTER_ITH_RETRY=	Send failure message every 'i' retries.
NOTIFY_AFTER_LAST_RETRY=	Send failure message only after last retry.

NOTIFY_CMDLINE_USERS =	Enables notification messages for command-line users (only).
NOTIFY_GUI_USERS =	Enables notification messages for GUI users.
NOTIFY_INCLUDE_LOG_FILES =	Include any active log files with failure message.
NOTIFY_INCLUDE_CONFIG_FILES =	Include all 'ini' files with failure message.
NOTIFY_INCLUDE_TEMP_FILES =	Include all relevant temporary files (in temp sub-directory) with the failure message.
NOTIFY_ITH_RETRY =	Specifies the value of 'i' for NOTIFY_AFTER_ITH_RETRY.
NOTIFY_MAX_MSG_SIZE =	Specifies the maximum message size, in kilobytes.
NOTIFY_NETWORK =	Specifies which network instance is to be used to transmit the notification message. Any active network instance may be specified.

Notes:

1. In addition to these values, you must configure the transfer used by the notification process to send the message. Configuring the notification transfer using the command-line can be done with a separate run of the command-line program. For example:

```
tdclientc RESET NETWORK=EMAIL "TRANSFER=(NAME='SEND TDCLIENT NOTIFICATION'  

SEND=somefile SEND_VERIFY=N SENDSUBJECT='Failure!'  

SENDUSERID=myTPBookEntry)" SAVE_ONLY
```

2. The NETWORK is set to the network instance to be used for transmission of the failure notification message. Here, EMAIL is being used.
3. The name of the transfer must be SEND TDCLIENT NOTIFICATION.
4. Although only the sendsubject and senduserid are actually used by the notification process (all other values are controlled by the program), you must include the SEND= keyword to define a 'legal' transfer. Use the SEND_VERIFY=N keyword to allow the specification of any filename. The filename will not be used, since the notification process creates it's own file containing the notification message and uses that filename instead.
5. For other network styles, use the appropriate keyword in place of SENDSUBJECT=. For example, SENDCLASS=, or SENDAPRF= may be used for FTP styles.
6. The SAVE_ONLY keyword is helpful since we do not want to save the transfer and not execute it.

3.4 Firewall Parameters

The following are keywords for specifying firewall parameters.

Keyword Usage	
Keyword	Usage
FIREWALL_HOSTIP=	AS2 network style only; applicable only for FIREWALL_TYPE=2. Text that specifies IP address or domain name of Proxy server.
FIREWALL_PASSWD=	AS2 network style only; applicable only for FIREWALL_TYPE=2. Text that specifies Password for logging onto Proxy server.
FIREWALL_PORT=	AS2 network style only; applicable only for FIREWALL_TYPE=2. Numeric value that specifies the port on which the Proxy server is listening.
FIREWALL_TYPE=	AS2 network style only. Numeric value that specifies if the client must pass thru a proxy server or not to get out to the target AS2 server. Allowed values: 0 => No proxy server 2 => Proxy server in use The TDClient client supports Basic Authentication, Digest Authentication and (Windows only) NTLM Authentication protocols.
FIREWALL_USERID=	AS2 network style only; applicable only for FIREWALL_TYPE=2. Text that specifies User ID for logging onto Proxy server.
LOCAL_HOST_IP_ADDRESS =	Text that specifies the IP address of the local host. This is useful for machines that are multi-hosted, when operating in an environment using Network Address Translation (NAT).

3.5 Database Parameters

The following are keywords for overriding database parameters.

The database may be the source of certificates, trading-partner relationships, and other information. Additionally, the database may be used to store AS1/AS2 Message Disposition Notifications (MDNs).

Keyword Usage	
Keyword	Usage
USE_DB_FOR_MDNS=	Specifies the database is to be used to store information about AS1 and AS2 message traffic.
USE_DB_FOR_RUNTIMES=	Specifies the database is to be used as the source of certificates and other trading-partner info.
DB_DSN=	Specifies the DSN for the database (required).

DB_PASSWD=	Specifies the database logon password (optional).
DB_SCHEMA=	Specifies the database schema (optional).
DB_TYPE=	Specifies the database type; allowed values include: "DB2" DB2 database "ODBC" Any ODBC-compliant database "ORACLE" Oracle database
DB_USERID=	Specifies the database logon user ID (optional).

3.6 Email Message Handling

The following are keywords for specifying email message handling.

Keyword Usage	
Keyword	Usage
ALL_PARTS_ONLY=Y or N	For EMAIL and AS1 network styles only. Specifies whether or not partial messages are to be ignored until all the parts for the whole message are available.
AUTO_COMBINE=Y or N	For EMAIL and AS1 network styles only. Specifies whether or not partial messages are to be automatically combined after download. Only valid in combination with ALL_PARTS_ONLY=Y.
AUTO_DELETE=Y or N	For EMAIL and AS1 network styles only. Specifies whether a message is to be auto-deleted from the mail (POP3) server after it has been retrieved.
BREAK_APART=Y or N	For EMAIL and AS1 network styles only. Specifies whether large messages are to be broken into smaller messages before transmitting. This does not affect the content of the message, but allows for the piece-meal transmission of large files through Mail Servers that would otherwise prevent it due to size. The re-assembly of such large messages is performed by the receiving Mail Agent or AS1 client program. Support for message break-apart and re-assembly is not required by the IETF draft AS1 specification, so you should verify your trading partner's Mail Agent or AS1 client supports this feature. The bTrade TDClient, TDAccess, and TDPeer products all support this feature.
BREAK_APART_SIZE=	For EMAIL and AS1 network styles only. Relevant only if BREAK_APART=Y. Threshold size, in kilobytes (KB), for breaking apart large messages into smaller sub-messages. Size must be greater than or equal to 50.

3.7 EDI-INT (AS1/AS2) Message Handling

The following are keywords for specifying EDI-INT message handling.

Keyword Usage	
Keyword	Usage
AS1_MDN_DISP_TO=	AS1 network styles only. Text that specifies an override to use to the default behavior. For AS1, the default behavior is to have the MDN returned to your e-Mail address. For AS2, the default behavior is to have the MDN returned synchronously during the communications session with the server address to which MDNs are to be sent in response to messages sent by TDClient. For AS1 network styles, this is just an e-Mail address (for example, myname@mybusiness.com).
AS2_MDN_DISP_TO=	AS2 network styles only. Text that specifies an override to use to the default behavior. For AS2, the default behavior is to have the MDN returned synchronously during the communications session with the server. If an AS2_MDN_DISP_TO value is specified, then the MDN will be returned asynchronously to the specified address. Since the AS2 protocol supports both AS1 and AS2 style MDNs, you may specify either format here. For example, to request an AS1-style MDN: <pre>AS2_MDN_DISP_TO=mailto:myEmail@MyServer.com</pre> or to request an AS2-style MDN: <pre>AS2_MDN_DISP_TO=http://myAS2Server.Com:80</pre> Or <pre>AS2_MDN_DISP_TO=https://myAS2Server.Com:443</pre>
CERT_IMPORT_DIR=	AS1 and AS2 network styles only. Text that specifies the directory to receive certificates sent to the user's AS1 mailbox or the user's AS2 server via "certificate-only" messages. The extracted (untrusted) certificate is copied into the directory in PKCS7 format, for further (manual or automated) processing by the user or the user's application.
DIVERT_DUP_DATA= Y or N	For EMAIL, AS1, and SMIME network styles only. Specifies whether or not messages received which contain either identical data or an identical message ID as that received in a previous message are to be diverted to the error sub-directory.
MDN_DIR=	AS1 and AS2 network styles only. When specified this indicates that all MDNs sent or received are to be stored in the specified directory.

SECURITY_LEVEL=	<p>AS1 and AS2 only. Required minimum level of security on incoming AS1 and AS2 messages; allowed values include:</p> <p style="padding-left: 40px;">1 => accept any data</p> <p style="padding-left: 40px;">2 => required data be signed</p> <p style="padding-left: 40px;">3 => require data be encrypted</p> <p style="padding-left: 40px;">4 => require data be signed and encrypted</p> <p>If a file is received without the required level of security, a negative MDN is generated (if requested), and the message is copied to the Error directory.</p>
------------------------	---

3.8 File Viewing Option (Windows TDIImage Only)

The following keyword specifies a file viewing option.

Keyword Usage	
Keyword	Usage
AUTO_VIEW_IMAGES=Y or N	Specifies whether or not an appropriate file viewer should be launched for each image file received. The file viewer is selected based on the image type, and the registered file types available in the Windows registry.

3.9 Overriding Network Parameters

The following keywords are used either at the command line or within a command file.

Keyword Usage	
Keyword	Usage
AS2NAME=	Your "AS2 name". When sending AS2 messages, this is the value which appears in the 'AS2-To:' field.
AS2TIMEOUT=	<p>(AS2 style only) Sets the client's AS2 session timeout. Allowed values include:</p> <p style="padding-left: 40px;">0 => use default settings (45 minutes)</p> <p style="padding-left: 40px;">Integer value greater than zero => timeout in seconds.</p>
CASE=U or L	Overrides the case-sensitivity setting for the network. CASE=U tells TDAccess to convert everything sent to the server to upper case. CASE=L tells TDAccess to send data to the server unchanged.
CONTROLPORT=	Overrides the Command-channel port number for communicating with the server (FTP only).

DATA_OVER_COMMAND =Y or N	EAFTP style only.
DHONLY=	Specify whether or not the client is to do ONLY Diffie-Helman non-certificate-based SSL. (Options are Y or N: Yes or No)
EDIREPLYBUF	This site command fixes a problem when sending an EDI file with a large number of interchanges that caused the FTP session to hang due to a backlog of server responses. (IGN network styles only)
EMAIL_ADDRESS =	Your email address. For EMAIL and AS1 styles, your email address is put in the 'From:' field in all messages created. For AS1, your email address is the default address used when specifying where Message Disposition Notifications (MDNs) should be sent by the receiver of messages you send. For AS2, the email address is used merely to indicate that an MDN is being requested. This is a required field for AS1 and AS2; it is stored in the [SECURITY] section of your tdclient.ini file.
ENABLE_ALIAS_PROBE= Y or N	IGN (FTP) network style only. If Y, then when sending EDI data to the IGM I/E FTP Gateway, the SITE EDIALIASPROBE 1 command is issued allowing the transmission of EDI files containing multiple envelopes, some of which are addressed to regular I/E mailboxes, and some of which are addressed to mailboxes via Alias Table(s). When set to N, then the SITE EDIALIASONLY 1 command is issued, as in previous releases of TDClient.
FTPPASSWD=	Text name. Same as PASSWD=. Kept for backward compatibility.
FTPTIMEOUT=	(FTP styles only) Sets the client's FTP session timeout. Allowed values include: 0 => use default settings (60 seconds) Integer value greater than zero => timeout in seconds.
FTPUSERID=	Text name. Same as USERID. Kept for backward compatibility.
HIGH_CLIENT_PORT=	Numeric value. Together with LOW_CLIENT_PORT specifies a range of client ports to be used. If LOW_CLIENT_PORT=0, then any available client port will be used; Else if HIGH_CLIENT_PORT=0, or LOW_CLIENT_PORT=HIGH_CLIENT_PORT, then only the port specified by LOW_CLIENT_PORT will be used. Else, a port in the range from LOW_CLIENT_PORT to HIGH_CLIENT_PORT (inclusive) will be used. If no port is available in the specified range, then a failure is reported and the communications session is ended.
IP=	Hostname or IP address that overrides the server address.
IP2=	Hostname or IP address that overrides the backup server address.

LOW_CLIENT_PORT=	Numeric value. See HIGH_CLIENT_PORT for description.
MAILBOXPASSWD=	(OPEN*NET network style only) Text. Overrides the server mailbox password.
MAILBOXUSERID=	(OPEN*NET network style only) Text name. Overrides the server mailbox userid.
NETSTYLE=	<p>Overrides the communications style used to communicate with the server.</p> <p>Supported FTP styles are:</p> <ul style="list-style-type: none"> - "CONNECTMAIL" (Sterling) - "EAFTP" - "EDI*Express" (GEIS) - "EDISwitch" (GEIS) - "EDS*ELIT" - "FEDEXNET" - "FTP-EDI" - "GE-INTEGRATION-LINK" - "GEIS-ENTERPRISE" - "GENERIC" - "GENERIC-DOS" - "GENERIC_MVS" - "GENERIC_SSL" - "GISB-CLIENT" - "GISB-SERVER" - "ICC-NET" - "IGN-IE" - "MARK_III" (GEIS) - "MCI-Edi*Net" - "OPEN*NET" (GEIS) - "STERLING-ENTERPRISE" - "Sterling-Commerce" - "WALMART" <p>Supported SMTP/POP3 styles are:</p> <ul style="list-style-type: none"> - "AS1" - "EMAIL" <p>Supported style for compression/encryption without any file transfer is:</p> <ul style="list-style-type: none"> - "LOCAL-ARCHIVE" <p>Other Supported Network Styles:</p> <ul style="list-style-type: none"> - "AS2" - "SMIME"

NETWORK=	Specifies an existing network to use as the basis for the communications session. The text must match one of the networks already defined in your <code>tdclient.ini</code> file. This keyword tells TDAccess to use the specified network in the <code>tdclient.ini</code> file as the current network. Note that any overrides to this network's data (via the keywords described below) will only apply for a single program run unless the <code>SAVE</code> keyword is also used.
PASSIVE= Y or N	Overrides the "passive" mode setting for the session (FTP only). <code>PASSIVE=Y</code> is used in certain circumstances to allow communications through a firewall.
PASSWD=	Overrides the server login password.
POP_PASSWD=	Overrides POP3 server login password.
POP_USERID=	Overrides the POP3 server login userid.
POP3_SERVER= Same as IP2. (AS1)	Hostname or IP address of POP3 server.
PROXY_PASSWD=	Text. The password passed to a <code>PROXY_TYPE=1</code> proxy server.
PROXY_TYPE=	<p>Non-negative numeric value. When non-zero, specifies the type of FTP PROXY server to be negotiated before connecting to the final destination FTP server. Supported types are limited to just one, currently:</p> <p><code>PROXY_TYPE=1</code>: First the application logs onto the Proxy Server using the proxy-userid and proxy-password. The userid has the form:</p> <p style="text-align: center;"><code>proxy-userid@destination-address</code></p> <p>After validating the userid and password, the Proxy Server extracts the final destination IP address (or domain name) from the userid, and establishes the connection to the destination server. At this point the destination server is essentially handed control of the session, and typically prompts for it's own userid and password. Note that for this type of proxy server, the IP address or domain name of the proxy server is specified using the <code>IP=</code> keyword (as opposed to the non-proxy case where <code>IP=</code> specifies the address of the final destination FTP server). Also note that the <code>USERID=</code> (or <code>FTPUSERID=</code>) and <code>PASSWD=</code> (or <code>FTPPASSWD=</code>) keywords still specify the login data for the final destination FTP server.</p>
PROXY_USERID=	Text. The userid passed to a <code>PROXY_TYPE=1</code> proxy server. The value must take the form <code>userid@destination</code> address to be compatible with a type 1 proxy server.
SERVER_RESPONSE_BUFFERING=Y or N	IGN (FTP) network style only. If Y, then TDClient instructs the IBM I/E FTP Gateway to buffer it's response(s) when it is receiving EDI data until the entire file has been received. If N, then TDClient instructs the IBM I/E FTP Gateway to send a response to each interchange as it is processed by the IBM server.

SITEDELAY=	Specifies a value, in seconds, to wait prior to sending each command (FTP only). A value of zero is usual. Infrequently a server will require a delay to handle certain timing problems.
SMTP_SERVER=	Hostname or IP address of SMTP server (AS1 and Email).
SSL=Y or N	Tells TDAccess whether to use SSL 3.0 when establishing a session with the server (FTP only).
SUNIQUE=	Numeric value. FTP-based network styles only. Determines whether or not to tell the server to use unique filenames. Valid values include: 0 => do not use storeUnique functionality 1 => use storeUnique WITH serverFileSpec as arg on STOU command 2 => use storeUnique WITHOUT serverFileSpec as arg on STOU command
TIMEOUT=	Synonym for FTPTIMEOUT= and AS2TIMEOUT= keywords.
URI=	URI portion of AS2 server address (same as IP2).
URL=	Hostname or IP address of AS2 HTTP or HTTPS server.
USERID=	Overrides the server login userid.

Notes:

1. The SMTP server should be configured to handle messages with signatures and encrypted messages. Some SMTP servers, such as Microsoft's Exchange Server must be specifically configured to allow signed messages to pass thru the SMTP server without being modified by the server.
 2. If you use the passwd, pop_passwd or mailboxpwd keywords, your password is entered as clear text on the command-line or from the command-file. **YOU SHOULD CHECK WITH YOUR SECURITY MANAGER BEFORE USING THIS KEYWORD.**
 3. If the network server supports the ability to change your password, this may be accomplished using either PASSWD=old-password/new-password, or PASSWD=old-password/new-password/new-password, depending on the syntax expected by the network server.
 4. For data_over_command - if 'Y', then use "data-over-command" variant of FTP, which uses a single socket connection. If 'N', then uses conventional FTP command and data socket connections. (EAFTP network style only)
 5. The default is NOT TO SAVE THE NETWORK DATA.
-

3.10 File Transfer Execution

The following keywords are used either at the command line or within a command file to define “transfers” which send and/or receive files.

Keyword Usage	
Keyword	Usage
DELETE_SERVER_FILE=	Specifies that the specified file be deleted on the server. In both cases, the filename specified with these keywords is the server's name for the file.
ELIT_RECEIVE_FILENAME=	(EDS*ELIT only) Specifies the ELIT Filename for a Receive transfer. Note that the EDS*ELIT style uses the ELIT_RECEIVE_FILENAME=, ELIT_RECEIVE_REFNO= and ELIT_RECEIVE_OPTIONS= instead of the RECEIVECLASS, RECEIVEAPRF, or RECEIVESUBJECT keywords.
ELIT_RECEIVE_OPTIONS=	(EDS*ELIT only) Specifies further ELIT options for a Receive transfer, including: <p style="text-align: center;">DSN=<DataSet name>, CTLNO=<ELIT Control Number>, FD Specifies that only files sent using the FROMDATA keyword (i.e. sent using the SendEDI option) are to be processed.</p>
ELIT_RECEIVE_REFNO=	(EDS*ELIT only) Specifies the ELIT Reference # for a Receive transfer.
ELIT_SEND_FILENAME=	(EDS*ELIT only) Specifies the ELIT Filename for a Send or SendEDI transfer. Note that the EDS*ELIT style uses the ELIT_SEND_FILENAME=, ELIT_SEND_REFNO= and ELIT_SEND_OPTIONS= instead of the SENDCLASS, SENDAPRF, or SENDSUBJECT keywords.
ELIT_SEND_OPTIONS=	(EDS*ELIT only) Specifies further ELIT options for a Send or SendEDI transfer, including: <p style="text-align: center;">SID=<SenderId>, FB=<Feedback file name>, LRECL=<MVS logical record length>, and STRIPLF Specifies CRLF characters are to be removed</p> <p style="text-align: center;">KEEPLF Specifies CRLF characters are to be retained where STRIPLF and KEEPLF are mutually exclusive and apply only when SENDASCII=Y</p>

bTrade TDAccess
 CLI and API User Guide

ELIT_SEND_REFNO=	(EDS*ELIT only) Specifies the ELIT Reference # for a Send or SendEDI transfer.
FROM_ARCHIVE=	(DATAGUARD only) Specifies directory and file name of the secured file to be accessed.
LOGINPASSWD=	Specifies a server login other than the default network login is to be used prior to executing the transfer.
LOGINUSERID=	Specifies a server login other than the default network login is to be used prior to executing the transfer.
NAME=	REQUIRED. Specifies the name of the transfer.
RECEIVE=	Specifies the file to receive a mailbox entry being downloaded from the server.
RECEIVE_SERVER_FILE=	Allows the user to specify that only the specified file be retrieved from the server.
RECEIVEAPRF=	Same as RECEIVECLASS.
RECEIVECLASS=	Specifies that only files sent to the specified Class or APRF to be downloaded. For EMAIL and AS1, specifies that only files with the specified subject-line are to be downloaded.
RECEIVEEDI=	Specifies the file to receive an EDI mailbox entry being downloaded from the server.
RECEIVESUBJECT=	Email and AS1 only. Same as RECEIVECLASS.
RECEIVEUSERID=	Specifies that only files sent to your mailbox from the specified userid are to be downloaded.
SEND=	Specifies the qualified filename of a file to be sent to the server. A transfer may either specify SEND or SENDEDI, but not both.
SENDAPRF=	Same as SENDCLASS.
SENDCLASS=	Specifies the Class or APRF to receive the file being sent. For EMAIL, AS1 and AS2, specifies the Subject line in the message being sent.
SENDEDI=	Specifies the qualified filename of an EDI file to be sent to the server.
SENDIMAGE=	(TDImage product only) Specifies the qualified filename of an image file to be sent to the server. Supported image formats include: BMP (Windows bitmap), JPG (JPEG), and JP2 (JPEG2000).
SENDSUBJECT=	For Email and AS1 only. Same as SENDCLASS.

SENDUSERID=	Name of userid (mailbox) on the server to receive the file being sent. For IGN, if SENDEDI is specified, this keyword specifies the Alias Table to be used.
SENDXML=	Specifies the qualified filename of an XML file to be sent to the server.
TO_EDINAME=	Same as SENDUSERID. For AS1 and AS2 there must be an entry in the Trading Partner address book.
TRANSFER=	Used to specify an existing transfer, or create a new transfer, and add it to the list of transfers to be run. To specify an existing transfer, use the form: TRANSFER=name or TRANSFER=" name " (quotes are required if 'name' contains spaces). To create a new transfer, use the form: TRANSFER=(keyword=value keyword=value... keyword=value) where all the various keywords which define the transfer appear within the (required) parentheses.
TO_ARCHIVE=	For Dataguard only. Specifies directory and file name to receive the secured file.
USE_ORIGINAL_NAME	Specifies that the original filename is to be used as the filename for received files. Available only for IGN and EAFTP network styles. Note that the filename specified in the RECEIVE= or RECEIVEEDI= keyword must specify a directory only; if the RECEIVE or RECEIVEEDI keyword specifies a fully qualified filename then it will be used instead and the USE_ORIGINAL_NAME keyword will be ignored.
USE_SERVER_NAME	Specifies that the server's filename is to be used as the filename for received files. Note that the filename specified in the RECEIVE= or RECEIVEEDI= keyword must specify a directory only; if the RECEIVE or RECEIVEEDI keyword specifies a fully qualified filename then it will be used instead and the USE_SERVER_NAME keyword will be ignored.

Notes:

1. If you use the login_password keyword, your password is entered as clear text on the command-line or from the command-file. **YOU SHOULD CHECK WITH YOUR SECURITY MANAGER BEFORE USING THIS KEYWORD.**
2. If the network server supports the ability to change your password, this may be accomplished using either LOGINPASSWD=old-password/new-password, or

LOGINPASSWORD=old-password/new-password/new-password, depending on the syntax expected by the network server.

3.11 Overriding the Default Parameters Used to Send Files

The following keywords are used either at the command line or within a command file.

Keyword Usage	
Keyword	Usage
COMPRESS=Y or N	Tells TDAccess whether to compress a file before it is sent to the server.
COMPRESS_RATE=	(TImage product only) Applies only when JPEG2000_ENCODE=Y (or COMPRESS=Y) and SENDIMAGE=<filename> is in use. Allowed values include: L - specifies loss-less compression xx - where xx is an integer value greater than 0 and less than 100; specifies percent compression. For example, a value of 90 instructs TImage to reduce the file size by 90%. xxB - where xx is an integer value greater than 0 and less than the files current size in bytes; specifies compressed file size in bytes. For example, a value of 12345B instructs TImage to attempt to compress the image file down to 12345 bytes in size. Note that this size does not include an MIME headers or base64 encoding which may be added during message construction after the jpeg2000 encoding.
CRLF= Y or N	Tells TDAccess whether to use the CRLF option during compression.
DELETE_AFTER_SEND=Y or N	Tells TDAccess whether or not to delete the file after it has been successfully sent.
FILTER=Y or N	Tells TDAccess whether to use the FILTER option during compression.
JPEG2000_ENCODE=	(TImage product only) Y or N. Tells TImage whether or not to encode an image file using the jpeg2000 encoder, enabling image compression, (See COMPRESS_RATE= keyword), before sending the file to the server. Note that this keyword is synonymous with COMPRESS=Y, but specifically applies to SENDIMAGE= transfers.

OTHER_COMP_PARMS =	Specifies advanced Comm-Press compression parameters. Below is a list of supported keywords. The advanced parameters should be entered just as they would appear on the command-line invocation of the Comm-Press compression program. Example Usage: "TRANSFER=(NAME=mytransfer OTHER_COMP_PARMS='lrec1=72 delim=250 ')"
PERPETUAL_SEND=Y or N	Tells TDAccess whether or not to make the transfer repeat it's Send-cycle as specified by the RETRY=, MAX_RETRY=, and RETRY_DELAY= keywords.
SECURE=Y or N	Tells TDAccess whether to use the SECURE option during compression.
SENDASCII=Y or N	Tells TDAccess whether to treat the file being sent to the server as an ASCII file.

Note

The advanced parameters can be delimited using single or double quotes, parentheses, or square or curly braces.

3.12 Compression, Decompression and Security

The **COMPRESS** and **DECOMP** programs support several options that are used to invoke the programs' advanced features. Add options to the workstation command line or MVS and AS/400 **PARM** clause to invoke these features. Some features are common to all computer platforms supported by TDAccess and some features are platform specific. Tables listing the **COMPRESS** and **DECOMP** options and the computer operating platforms where they apply, along with keyword usage, are displayed in the following tables.

Keyword Usage					
Keyword	Usage	Windows	Unix/Linux	AS400	MVS
APPEND	<p>This option commands COMPRESS and DECOMP to append data to the end of existing output files. The output files are created, if they do not exist. Appended compressed files can be separated into individual files during decompression. Note, that an implicit APPEND operation takes place when multiple input files are specified (via a file mask) but a single output file is produced. For example, the following command causes all files in the INDIR directory to be compressed into the single output file COMP.OUT:</p> <p>Compress \indir*. * \compwork\comp.out</p>	X	X	X	X
ARCHIVE	<p>This option commands DECOMP to save specific information relating to an authenticated file to enable reprocessing. Related information is stored in a dynamically allocated file containing a copy of the input file as well as in the decomp.log. The syntax of this option is ARCHIVE—stores the dynamically allocated files into the current working directory or ARCHIVE=path—defines a directory path (or high-level qualifier) to be specified for file storage.</p>	X	X	X	X
ASCII	<p>This option causes COMPRESS to translate the data to ASCII or EBCDIC, if necessary, depending on the platform where the data is decompressed. The ASCII and CRLF options should always be used when compressing text files.</p>	X	X	X	X

AUTOEXT[=n]	<p>This option commands COMPRESS and DECOMP to automatically generate numeric extensions to name the output files. The optional value n specifies the number of digits to use in the extension. n may be in the range from one to nine. The default number of digits is three. The generated extensions start with two and continue to the highest number possible for the number of digits in the extension. All extensions are left-padded with zeros so they are n digits in length. The extensions are appended to the output name specified on the command line to create the full output file name. If the output name specified on the command line does not exist, then the name without an automatic extension, is used to name the first output file. Subsequent output files contain extensions. Existing output files are not overwritten. When decompressing, if the compressed files contain the original file name in the signature, the original names are used to name the output files unless the NOINFO option is specified. For example, the following command decompresses the input data into separate output files named DCMP.OUT, DCMP.002, DCMP.003, etc. The original file names are ignored.</p> <p>decomp \indir*. * \compwork\dcmp.out autoext noinfo</p>	X	X		
BISYNC	<p>This option invokes a proprietary filter algorithm that prevents bytes within the compressed data from being misinterpreted as BISYNC control characters. Use of this option is not recommended. If your BISYNC communications software does not support transparency mode, use the FILTER option described below.</p>	X	X	X	X
CCA	<p>Optional Special Feature support.</p>				X
CRLF	<p>This option commands COMPRESS to convert delimiter characters (for example, line feeds or carriage return/line feed pairs) into record separators. On AS/400 and MVS machines, COMPRESS inserts record separators at the end of each input record. During decompression, the delimiter characters that are appropriate for the target platform replace the record separators. On the PC and UNIX workstations, this option causes an x'1A' character to be treated as an end-of-file marker unless the IGNORE1A option is also chosen. The ASCII and CRLF options should always be used when compressing text files.</p>	X	X	X	X
DE3	<p>DE3 invokes the Triple DES encryption algorithm. Triple DES is simply the DES algorithm executed three times with three different keys. A 24-byte encryption key (actually three 8-byte keys) must be supplied and the same key must be used to decompress the data.</p>	X	X	X	X

DELETE	This option causes DECOMP to delete the input files after successful decompression.	X	X		
DELIMIT=nn	This option, in conjunction with the FILTER option, creates compressed data in delimited-text format. COMPRESS adds delimiters (for example, line feeds or carriage return/line feed pairs) after every nn characters of compressed output. The FILTER option is automatically invoked if this option is used.	X	X		
DES	This option invokes the Data Encryption Standard—U.S. government standard encryption algorithm. An 8-byte encryption key must be supplied and the same key must be used to decompress the data.	X	X	X	X
DIRNAME	For COMPRESS, this option stores the full path of the input file, including the directory, in the compressed output. Default action is to store only the file name. For DECOMP, this option uses the input file path names stored in the compressed data to build an output path for the decompressed files. The input file path names are concatenated to the output path specified on the command line, or to the current directory if no output path was specified, in order to generate the actual path name for the decompressed files. If the resulting output path does not exist, it is automatically created. This option is not valid in combination with the EDI parameter because directory information is not stored by COMPRESS processing.	X	X		
EDI	This option may be used when X12, EDIFACT, UN/EDIFACT, or UCS EDI data is compressed. The header and trailer records that mark the start and end of an EDI envelope are left uncompressed in the output file. Use of this parameter also ensures that the compressed data does not contain any characters that could be misinterpreted as EDI control characters. If the EDI parameter is used to compress the data, then the EDI parameter must also be used to decompress the data.	X	X	X	X
FILTER	This option invokes the FILTER algorithm described in RFC 1113 to convert the compressed data from binary to text format. Filtered data is always transmitted as text. Use this option when the data communication environment does not allow transparent data transmission. FILTER is also required when the output data contains a combination of compressed and uncompressed data, as is the case when the EDI or PF options are used, and the data is transmitted between unlike computer platforms. For example, EDI data on a PC that is sent to an AS/400 should be compressed with the EDI and FILTER options. The resulting compressed file is then sent to the AS/400 as a text file.	X	X	X	X

IGNORE1A	This option, when used with the CRLF option, causes COMPRESS to not treat any x'1A' characters as end-of-file markers. Default processing when the CRLF option is chosen is to stop reading input when an x'1A' character is read.	X	X		
IV=iv	This option, when used with one of the encryption options DES, DE3 or RC2, provides the initialization vector for the encryption algorithm.	X	X	X	X
KEEPSIGS	This option is only valid when decompressing secured data. Digital signatures present in the secured data are verified and then removed from the output files by default. This option keeps digital signatures in the output.	X	X	X	X
KEY=key	This option, when used with one of the encryption options DES, DE3 or RC2, provides the key for the encryption algorithm.	X	X	X	X
LIST	This parameter causes DECOMP to print a listing of the compressed input with each compressed segment categorized by its size and file information (if present). LIST also checks each compressed segment for data integrity. Decompression is not performed when LIST is specified. This option is not valid in combination with the EDI parameter because the information is not stored by COMPRESS processing.	X	X	X	X
LOGPATH=path	This option gives the path or AS/400 library where the compression and decompression log files are written. The default is to write the log files in the current directory or AS/400 library. The names of the log files are compress.log and decomp.log. Workstation users can also set the LOGPATH= environment variable to set the location globally.	X	X	X	
LOOKUP	This option causes COMPRESS to read the CPLOOKUP file to determine whether to compress EDI data.	X	X	X	X
LOWERCASE	This option causes DECOMP to convert all file names stored in the compressed data to lower case before naming the decompressed output files.		X		

NOINFO	For COMPRESS, this option overrides the default storage of the input file name and date in the compressed output. This information is normally used to name the output files, if they are decompressed on a PC or UNIX workstation. For DECOMP, this option causes any directory information stored in the compressed files to be ignored when naming the decompressed output files. This option is not valid in combination with the EDI parameter because the information is not stored by COMPRESS processing.	X	X	X	X
NOLOG	This option suppresses the creation of the compression and decompression log files.	X	X	X	X
NOUNCOMP	This option causes the MVS and AS/400 decompression programs to ignore uncompressed data in the input. The default is to write uncompressed data to the output file <i>as is</i> .	X	X	X	X
ODATE	This option causes decompressed output files to retain the creation date and time of the original compressed input files. ODATE is only valid in true DOS mode.	X			

PF=filename	<p>This option is used when certain input records are to be left uncompressed. On PCs and UNIX workstations, the input data must be delimited for this option. Replace filename with the fully qualified name of a parameter file that contains information about the records that are to be left uncompressed. Each record in the parameter file contains a character string, its starting position in the input records, and its length. COMPRESS examines each input record to see if one of the character strings occurs in the specified location. If a match is found, then the record is left uncompressed in the output file. The parameter file can contain as many records as necessary to provide all the character strings used to identify uncompressed records. The format of the parameter file records is:</p> <p><starting position> < length > <character string></p> <p>Each parameter file record must end with a record delimiter appropriate to the computer platform (for example, a line feed character on UNIX, or a carriage return/line feed pair on a PC). As an example, if records that contain the string HEADER starting in column 10 and records that contain the string TRAILER starting in column 1 are to be left uncompressed, then the parameter file will contain the following two records:</p> <p>10 6 HEADER 1 7 TRAILER</p> <p>This option is not valid in combination with the EDI parameter.</p>	X	X	X	X
QUIET	This option suppresses all output messages from COMPRESS and DECOMP.	X	X	X	X
RC2	This option invokes the RC2 encryption algorithm. This algorithm is exportable to foreign countries with no special authorization from the U.S. government. An 8-byte encryption key must be supplied because it must also be used to decompress the data.	X	X	X	X
RECEIVER=rcvr	This option may be used to provide the name of the security receiver when securing data. RECEIVER is only valid when the SECURE option is also specified.	X	X	X	X

RECURSE	This option causes COMPRESS to travel down the subdirectory tree and compress all files that match the input file mask. The DIRNAME option is automatically invoked when this option is specified.	X	X		
RUNTIMEPATH =path	This option may be used to provide the path where the security run-time files are located. RUNTIMEPATH is only valid when the SECURE option is also specified. Workstation users can also set the RUNTIMEPATH= environment variable to set the location globally.	X	X	X	
SAFE	This option causes DECOMP to create a reject file containing data that fails decompression or decryption. Only valid data is written to the output file(s).	X	X	X	X
SAVEMODE	This option causes UNIX and VMS file modes to be saved in the compression signature. The modes are restored on DECOMP.		X		
SECFILE=filename	This option provides the name of the security definition file that may be required when securing non-EDI data. This option is only valid when the SECURE option is specified.	X	X	X	X
SECURE	This option invokes the advanced security features of Comm-Press2000 Extended Security Option. This is the lenient form of invoking security. If no security relationship is defined (for example, no record found in the lookup run-time file), then the data is not secured, and no error is issued. Contrast this with the SECUREONLY option.	X	X	X	X
SECURECK	With this option DECOMP ensures that only decrypted data is written to the output file(s). Input data that is not encrypted is written to a reject file.	X	X	X	X
SECUREONLY	This option invokes the advanced security features of Comm-Press2000 Extended Security Option. This is the strict form of invoking security. If no security relationship is defined (for example, no record found in the lookup run-time file), then an error is issued and processing stops.	X	X	X	X

SELECT()	<p>This option allows selective decompression of compressed segments. The segments must contain embedded file name information (included by default when compressing on an AS/400 or workstation). Specify the exact file names of the segments to decompress in the parentheses that follow the SELECT parameter. Separate multiple file names with a comma. If multiple segments with the same file name are in the input file, then by default, the first segment is decompressed. Override this default action by adding a colon and relative sequence number to the file name. An example follows:</p> <pre>decomp infile \work select(file.txt.1)</pre> <p>The above example reads the compressed archive file named INFILE. It decompresses the first occurrence of the compressed file named FILE.TXT. \WORK is the directory where the decompressed file is written.</p> <p>This option is not valid in combination with the EDI parameter because the filename is not stored by COMPRESS processing.</p>	X	X	X	X
SENDER=sndr	<p>This option is used to provide the name of the security sender when securing data. SENDER is only valid when the SECURE option is specified.</p>	X	X	X	X
SIZE	<p>This option creates the smallest possible compressed output at the expense of processing time.</p>	X	X	X	X
SKIPWHITESPACE	<p>This option is valid only in conjunction with the EDI parameter. SKIPWHITESPACE causes invalid characters in EDI segments (for example, carriage return/line feed characters) to be skipped. This option is useful when the transmission software inserts the invalid characters and causes COMPRESS and DECOMP to process the EDI data incorrectly.</p>	X	X	X	X
SPEED	<p>This option decreases processing time at the expense of compressed output size. In many cases the savings in processing time make up for the slightly larger compressed output.</p>	X	X	X	X

SQL(ssn,plan)	This option can be used when securing data in an MVS-DB2 environment. If MVS-DB2 is the TDManager repository, then COMPRESS and DECOMP can directly access these tables for run-time information. This option specifies the DB2 subsystem name and plan used to access the TDManager database tables. Run-time files are not used when this option is specified. Default: ssn=DSN plan=COMPSQL				X
STDIN	This option causes COMPRESS to read its input from STDIN rather than from disk files.	X	X		
STDOUT	This option causes DECOMP to write its output to STDOUT rather than to disk files. Specify the QUIET parameter in addition to STDOUT to keep messages from appearing in the data file.	X	X		
TRANSID=trid	This option is used to provide the name of the security transaction ID when securing data. TRANSID is only valid when the SECURE option is specified.	X	X	X	X
TRANTBL	This option provides the name of an ASCII/EBCDIC translation table to use when compressing and securing data. TRANTBL when used on an ASCII workstation does not translate data being compressed. It would only affect hash processing for authentication. Yes or No overrides the CPLOOKUP table. Filename gives the name of an actual translation table. Transupdate is supported on MVS.	X	X	X	X
TRLBLK	This option, when used with the CRLF option, causes trailing blanks in MVS and AS/400 records to be removed from the compressed data. Default CRLF processing keeps trailing blanks.			X	X
UNCOMP	This option signals DECOMP that the input files contain valid uncompressed data in addition to compressed data. DECOMP will copy the uncompressed data to the output files as it decompresses. If this option is not specified, then any data that occurs between the end of a compressed segment and the beginning of the next compressed segment are assumed to be pad characters and are ignored. This option is not valid in combination with the EDI parameter, since all data outside the EDI envelope is ignored and passed as-is by default.	X	X	X	X
UNCOMPSIG	This option tells DECOMP to expect a signature at the end of the file during a session.	X	X	X	X

UNWRAP	This option is valid only in conjunction with the EDI option. When both the EDI and UNWRAP options are specified, COMPRESS and DECOMP split EDI segments so that each segment begins on a new record.	X	X	X	X
USEGS	This option is only valid when the SECURE and EDI options are also specified. USEGS causes the GS02 and GS03 elements of the X12 GS segment to be used as the security sender and receiver.	X	X	X	X
USEINFO	This option is used to name decompressed members on an AS/400. DECOMP does not normally create members when decompressing. All decompressed data is appended into the output file specified on the command line. USEINFO uses the file names stored in the compressed data to create members in the output file.			X	

3.13 Overriding the Default Parameters Used to Receive Files

These keywords allow for the specification of how received files are processed during a Receive or Receive EDI transfer.

Keyword Usage	
Keyword	Usage
APPEND=Y or N	Tells TDClient to append all downloaded files into the file specified by the RECEIVE or RECEIVEEDI keyword.
AUTOEXT=Y or N	Tells TDAccess to "auto-extend" the filename specified by the RECEIVE or RECEIVEEDI keyword. As each file is received, it is given a filename with a unique numeric extension. Either APPEND or AUTOEXT must be Y, but not both.
RECEIVEASCII=Y or N	Tells TDAccess to treat the file being downloaded from the server is in ASCII file format.
UNCOMP=Y or N	Tells TDAccess whether to use the UNCOMP option during decompression. If a file received is not compressed, this parameter must be Y in order to correctly process the file.
PERPETUAL_RECEIVE=Y or N	Tells TDAccess whether or not to make the transfer repeat its Receive-cycle as specified by the RETRY=, MAX_RETRY=, and RETRY_DELAY= keywords.
OTHER_DECOMP_PARMS=	Specifies additional Comm-Press decompression parameters, please see the Compress Options in the previous section of supported keywords. The advanced parameters should be entered just as they would appear on the command-line invocation of the Comm-Press decompression program. Example Usage: "TRANSFER=(NAME=mytransfer OTHER_DECOMP_PARMS='unwrap delim=250')"

Note:

The advanced parameters can be delimited using single or double quotes, parentheses, or square or curly braces.

3.14 Send/Receive Pre/Post Processing While Creating a Transfer

These keywords allow you to tell TDAccess to execute a program before or after you have it send or receive files, and, you must use these keywords inside a transfer definition.

Keyword Usage	
Keyword	Usage
POST_RECEIVE=	Specifies a program or command-file is to be run after to the Receive cycle of a transfer. See below for the syntax and related keywords.
POST_SEND=	Specifies a program or command-file is to be run after to the Send cycle of a transfer. See below for the syntax and related keywords.
PRE_RECEIVE=	Specifies a program or command-file is to be run prior to the Receive cycle of a transfer. See below for the syntax and related keywords.
PRE_SEND=	Specifies a program or command-file is to be run prior to the Send cycle of a transfer. See below for the syntax and related keywords.
RECEIVE_VERIFY = Y or N	(DATAGUARD only) Tells TDAccess to check for the existence of the file(s) you are telling it to Unsecure at the time the transfer is being created. The default is to check that the file(s) exist, to catch typing errors and so on. However, if you are executing an Unsecure (Receive) Pre-processing program, which will create the files to be unsecured, then you will want to disable this checking, since the files may not exist until the transfer is executed.
SEND_VERIFY=Y or N	Tells TDAccess whether it should check for the existence of the file(s) you are telling it to Send at the time the transfer is being created. The default is to check that the files exist, to catch typing errors and so on. However, if you are executing a Send Pre-processing program, which will create the files to be sent, then you will want to disable this checking, since the files may not exist until the transfer is executed.

3.14.1 PRE_SEND, POST_SEND, PRE_RECEIVE, and POST_RECEIVE Keywords

These keywords all use the following syntax to specify a program or command-line to be executed and how to tell TDAccess how to check if the program or command-line executed successfully.

Keyword Usage	
Keyword	Usage
CMDLINE=	Specifies the directory and filename of the program, batch file, script file, or operating system command to be executed, along with any arguments to be passed to the program/script/command.
EACH_FILE= Y or N	(Post Processing Only) If Y, then TDAccess will execute the post-processing after each file has been received. If N, then TDAccess will execute the post-processing after all files have been received to the program/script/command.
FAILS_ALWAYS	Tells TDAccess that the program/script/command always fails, regardless of its return code.
FAILS_IF_EQ	Tells TDAccess that the program/script/command failed if the return code is equal to the value specified by the RETCODE= keyword.
FAILS_IF_GT	Tells TDAccess that the program/script/command failed if the return code is greater than the value specified by the RETCODE= keyword.
FAILS_IF_LT	Tells TDAccess that the program/script/command failed if the return code is less than the value specified by the RETCODE= keyword.
RETCODE=	Numeric value that specifies a return code value to be used in determining if the program/script/command executed successfully or not.
SUCCEEDS_ALWAYS	Tells TDAccess that the program/script/command always succeeds, regardless of its return code.
SUCCEEDS_IF_EQ	Tells TDAccess that the program/script/command succeeded if the return code is equal to the value specified by the RETCODE= keyword.
SUCCEEDS_IF_GT	Tells TDAccess that the program/script/command succeeded if the return code is greater than the value specified by the RETCODE= keyword.
SUCCEEDS_IF_LT	Tells TDAccess that the program/script/command succeeded if the return code is less than the value specified by the RETCODE= keyword.

Note:

This is only useful in testing, since the pre- or post-processing will fail, causing TDAccess to consider the transfer to have failed.

The syntax used to specify pre- and/or post-processing is shown in the following example:

```
TRANSFER=(name=...sendclass=.....
```

UNIX example:

```
PRE_SEND=[ CMDLINE='sh -x myScript.ksh 2>err.out' RETCODE=127  
SUCCEEDS_IF_LT]
```

Windows example:

```
POST_SEND=[ CMDLINE='C:\MyPrograms\cleanup.exe /log' RETCODE=0  
SUCCEEDS_IF_EQ]
```

This transfer will execute a UNIX shell script (myScript.ksh) before the Send step and will execute a Windows program (cleanup.exe) after the Send step.

3.15 Handling Dial Connections (Windows platforms only)

These keywords specify how a Dial Up Networking connection is to be established automatically whenever needed when performing file transfers, Query Mailbox operations, and so on.

Keyword Usage	
Keyword	Usage
AUTODIAL=Y or N	Tells TDAccess whether it should "auto dial" the DialUp Networking entry if no current connection is in progress.
AUTODISCONNECT=Y or N	Tells TDAccess whether it should "auto-disconnect" the current connection when it exits.
BACKUP_DIAL=	Text name of a backup DialUp Networking entry you have previously set up on your computer. The backup Dial entry is used if the primary DialUp Networking entry or Dialer program fails to connect.
BACKUP_DIAL_PROGRAM=	Text filespec (fully qualified filename) of a Dialer program that you want launched (instead of DialUp Networking) as your backup. For example, the AT&T Global Network Services Dialer program IDIALER.EXE (plus directory path) may be specified. The backup Dialer program is used if the primary DialUp Networking entry or Dialer program fails to connect. If both BACKUP_DIAL= and BACKUP_DIAL_PROGRAM= are specified, then the Dialer program will be used as the backup.
DIAL=	Text name of one of the DialUp Networking entries you have previously set up on your computer.
DIAL_PROGRAM=	Text filespec (fully qualified filename) of a Dialer program that you want launched (instead of DialUp Networking). For example, the AT&T Global Network Services Dialer program IDIALER.EXE (plus directory path) may be specified. If both DIAL and DIAL_PROGRAM are specified, then the Dialer program will be used.
TIMEOUT=	Numeric value that tells TDAccess how long, in seconds, it should wait for a dial attempt to connect before it decides the attempt has failed.

3.16 Specifying Auto-Retry Options

These keywords control whether or not a failed task is to be automatically retried, and the details for any retries to be attempted.

Keyword Usage	
Keyword	Usage
MAX_RETRY=	Numeric value that tells TDAccess how many times to “Retry” a failed transfer.
RETRY=Y or N	Enables or disables Auto-Retry.
RETRY_DELAY=	Numeric value that tells TDAccess how long, in seconds, it should wait between retry attempts.

3.17 Adding/Modifying Entries in a Trading Partner Address Book

These keywords allow for the creation of Trading Partner Address Book entries.

Keyword Usage	
Keyword	Usage
AS2NAME1=	Specifies the "AS2 Name" for this trading partner. If NETWORK1 is an AS2-style network, then this value is used in the "AS2-From:" field. If left blank, the TPNAME value is used for the "AS2-From" field.
AS2NAME2=	Specifies the "AS2 Name" for this trading partner. If NETWORK1 is an AS2-style network, then this value is used in the "AS2-From:" field. If left blank, the TPNAME value is used for the "AS2-From" field.
HTTP_PASSWD1=	(AS2 primary network only) Specifies the HTTP password to be used to log on to the AS2 server. (Not required for most AS2 servers).
HTTP_PASSWD2=	(AS2 secondary network only) Specifies the HTTP password to be used to log on to the AS2 server. (Not required for most AS2 servers).
HTTP_USERID1=	(AS2 primary network only) Specifies the HTTP User ID to be used to log on to the AS2 server. (Not required for most AS2 servers).
HTTP_USERID2=	(AS2 secondary network only) Specifies the HTTP User ID to be used to log on to the AS2 server. (Not required for most AS2 servers).
MAILBOX1=	Specifies the Primary mailbox, user ID, or login name for the Trading Partner on the Primary Network.
MAILBOX2	(Optional) Specifies the Backup Mailbox, or user ID, or login name for the Trading Partner on the Backup Network.

NETWORK1=	Specifies the Primary Network to be used when sending to or receiving from this Trading Partner.
NETWORK2=	(Optional) Specifies the Backup Network to be used when sending to or receiving from this Trading Partner. The Backup Network is used only if a transfer fails using the Primary Network and Mailbox (if Auto-Retry is enabled). The Backup Network will be used for the last half of the specified retries. That is, if Auto-Retry is enabled and MAX_RETRY is 2, then when a transfer fails, the first retry will use the Primary Network and the second retry will use the Backup Network.
SERVER_EDINAME1=	(AS1 or AS2 primary network only) Specifies that the certificate associated with the specified EDI Name to use to encrypt the message, overriding the default trading partner certificate.
SERVER_EDINAME2=	(AS1 or AS2 secondary network only) Specifies that the certificate associated with the specified EDI Name is to be used to encrypt the message, overriding the default trading partner certificate.
TPBOOK=(.....)	Specifies an entry in the Trading Partner Address Book is to be added or edited. Here is an example command-file entry to create/modify a Trading Partner: TPBOOK=(TPNAME=MyPartner NETWORK1=bTrade MAILBOX1=MyPartnersMailbox NETWORK2='IGN-I/E SSL' MAILBOX2=CMAP.MyPartnersIGNAccount MAILBOX2=CMAP.MyPartnersIGNAccount)
TPNAME=	Specifies the Trading Partner name to be added/modified.

Notes:

1. You can use single or double quotes, square brackets, curly braces, or parentheses for transfer creation, and for pre/post processing specification, and for TPBOOK usage.
2. The network2 and mailbox2 entries are optional for TPBOOK.
3. You can use square brackets or parentheses for transfer creation, and for pre/post processing specification, and for TPBOOK= usage too
4. A value-substitution capability for AS2 http URL addressing is provided to accommodate the requirements imposed by some AS2 servers. The available substitutions include:

- %filename% - replaced with the base filename of file being transmitted
- %fromEdiName% - replaced with the EDI Name of sender
- %fromEdiQual% - replaced with the EDI Qualifier of sender
- %fromAS2Name% - replaced with the AS2 Name of the sender
- %toEdiName% - replaced with the EDI Name of receiver

%toEdiQual% - replaced with the EDI Qualifier of receiver

%toAS2Name% - replaced with the AS2 Name of the receiver

For example, if in the Trading Partner Address book the AS2 URI for a trading partner is https://MyPartner.com:28080/%fromAS2Name%_file_is_%fileName%, then when you send to this partner, the %fromAS2Name% and %filename% values will be replaced with the corresponding values in effect for the transfer being executed. That is, the current AS2 Name of the sender and the actual filename being packaged and sent will appear in the http address sent to the receiving AS2 server.

4 Canceling a Running Transfer (Command Line Only)

Sometimes it is convenient to cancel a running transfer, and thus you will want to do the following from time to time. To cancel an in-progress transfer, create the file 'cancel.fil' in the <TDAccess Installation> 'temp' subdirectory. The TDAccess program will terminate the transfer if it can. If it is able to respond, TDAccess will terminate with a return code of 2.

5 API Interfaces

bTrade TDAccess includes 2 Application Programming Interfaces (APIs) for developers to use to write programs to automate and interact with bTrade TDAccess or bTrade TDCompress.

The TDAccess API allows developers to programmatically alter the TDAccess application while running and sending/receiving information. There is also an API to TDCompress, a subcomponent of TDAccess. TDCompress is the component that performs compression, decompression and information security. The TDCompress API can be invoked without have TDAccess running. Thus, one will use the TDCompress API when the developer only wants to leverage the compression, decompression and security features of TDAccess. Using just TDCompress in this case provides for less overhead, being lightweight and efficient.

5.1 TDAccess API

The following explains how to set up and use the TDAccess API.

5.1.1 Extracting the API

To set up the bTrade TDAccess API, one needs to execute the `tdclient_api.exe` file located in the directory where bTrade TDAccess was installed. Sample command files and C programs, referenced below, are thus extracted and placed in the directory.

5.1.2 Using the API

The `tdclients.dll` provides a callable 'C' API for bTrade TDAccess client functionality. The API consists of a single entry point:

```
int tdClientCmdMain(char* programName, ...);
```

This takes a variable length argument, taking as arguments all of the keywords supported by the bTrade TDAccess client command-line program.

The first argument, `ProgramName`, can be any null-terminated character array. It takes the place of the normal `argv[0]` provided to any 'C' program when it is executed, which is the name of the program being executed.

All the subsequent arguments are keywords from the bTrade TDAccess command-line keyword set, using the same syntax as if the command-line program was being invoked. Along with referencing the Keyword section of this document, you can also review a

command-line documentation example file, `tdclient.cmd`, for further descriptions and examples of all the supported keywords.

There is also a working program example, `eatest.c`, for referencing how to use the `tdclients.dll`. This program invokes the `eaCmdMain` entry point several times to perform several tasks. You may need to modify this example program to reference one of the Network instances in your `tdclient.ini` file.

If you are using a different type of network style than that demonstrated, you might have to tailor the examples to the capabilities of the server to which you are communicating, since different network styles (styles of server) support different features. For example, not all network-styles support the `RECEIVE_AUDIT_LOGS` keyword.

The `tdClientCmdMain` function returns 0 if it successfully executes the specified tasks; otherwise it returns a nonzero error code.

The `tdclients.dll` expects to run in a directory containing the files used by the regular TDAccess client programs, which are listed in the table below.

Files	
File	Usage
<code>tdclient.ini</code>	Network definitions, current tasks, program settings
<code>exfer.ini</code>	Stored transfers
<code>tpaddrss.ini</code>	Trading Partner Address book
<code>commpr32.dll</code>	Compression
<code>tdclientd.dll</code>	Communications objects
<code>btftpw95.dll</code>	FTP transport (with optional SSL)
<code>btpop3.dll</code>	POP3 transport
<code>btsmtp.dll</code>	SMTP transport
<code>rimport.dll</code>	Imports SecureManager-generated 'runtimes'
<code>bthttp.dll</code>	HTTP/HTTPS transport (TDAccess version 2.0 and above)
<code>ssockapi.dll</code>	Socket API (TDAccess version 2.0 and above)

In addition, the following sub-directories are expected:

Sub-Directories	
Sub-Directory	Usage
Temp	Directory for temp files and several optional log files

Runtime	Directory for certificates, private keys, security tables
Security	Directory for certificates, private keys, security tables
Maint	Directory for receiving maintenance releases
Incoming	(Optional) directory for incoming files
Outgoing	(Optional) directory for outgoing files
Error	Directory for storing duplicate messages
Mdn	(Optional) for storing Message Disposition Notification messages

Notes:

1. The TDAccess client programs and the `tdclients.dll` 'own' the `tdclient.ini` and other files in the directory tree when they are executing. This means that multiple instances of the application cannot execute from the same directory at the same time. The DLLs are all thread-safe, multiple instances can execute from the same machine provided the application has been installed at least once and copied to different directories. In each directory tree three `.ini` files and the five required sub-directories are required. A single copy of all the DLLs and programs in a separate directory and execute using the `'INIPATH='` keyword may be used when invoking the programs or `tdclients.dll` to point to the directory to use for any given run.
 2. API interface will be modified for strategic clients based on design specifications jointly developed (such as an AQ wrapper, MQ wrapper, Vitria, etc).
-

5.2 TDCompress API

Application programmers can call the TDCompress utilities to compress, decompress, and secure data within their applications. A platform-specific static library, DLL, or shared object is provided that contains the TDCompress utilities as function calls. The libraries contain separate functions for compressing and securing data, and for decompressing and unsecuring data. All the features of the standalone utilities are available via the CALL interface.

Data is compressed and secured by calling either the `compress` function or the `compprog` function. Data is decompressed and unsecured by calling either the `decomp` function or the `dcmpprog` function. The functions differ in the specific interface between them, the caller and in how input and output is handled.

The `compress` and `decomp` functions expect to receive a parameter list like the parameter list built by the operating system when using the standalone utilities. The parameter list contains the input and output file names and any keyword options required. The called `compress` and `decomp` routines operate like the standalone utilities, reading and writing

files, and performing the requested compression and security. The compress and decompress entry points are referred to as the command-line API because they operate as if they had been executed from the command line.

The compress and decompress functions require the calling program to set up a parameter block used to specify options and pass status information between the caller and the functions. The calling program is responsible for all input and output operations; compress and decompress do not perform file handling. Instead, the calling program passes and receives data from the functions in memory buffers. The compress and decompress entry points are referred to as the interactive API because they interact with the calling program to process data.

5.2.1 Using the Command-Line API (COMPRESS and DECOMP)

When a C program is executed from the command line, the operating system passes two parameters to the program's main function. The first parameter is an integer that is the number of values entered on the command line. The second parameter is the address of an array of pointers to the values entered on the command line. The first value pointed to in the array is the name of the executed program itself, so there is always at least one value present. For example, to compress and secure a file named filein to the output file fileout, the command line might look like this:

```
compress filein fileout ascii crlf secure
```

When the compress program executes, it receives two parameters from the operating system — an integer, traditionally given the variable name argc, and the address of an array of pointers, traditionally given the variable name argv[]. For the example above, argc has a value of 6 and argv[] contains six pointers — a pointer to each value entered on the command line.

Calling programs can emulate the operating system by building the argv[] array and then calling compress or decompress with the argc and argv[] parameters. For a program to emulate the operating system using the command-line example above, it needs to allocate and initialize the six values as string variables. The argv array must be initialized to contain six pointers to the string variables. The program calls compress passing two parameters — the integer 6 and the address of the argv array. The following code fragment demonstrates this:

```
#include "comprss.h"
int CallCompress( )
{
    // declare and initialize local variables
    //
    int status;
```

```
char *newArgv[6] = {
    "compress",
    "filein",
    "fileout",
    "ascii",
    "crlf",
    "secure"
};

// call compression routine and return completion code
//
status = compress( 6, newArgv );
return status;
}
```

5.2.2 Using the Interactive API (compprog and dcmpprog)

A program that handles its own files, or that processes data that is not file oriented, calls the compprog and dcmpprog functions to add or remove compression and security. The caller sets up a parameter block that specifies the desired compression and decompression options. Most of the options correspond to one of the TDCompress command-line options, for example, **EDI**, **SECURE**, etc. Data is passed to and from the TDCompress functions in memory buffers. Next, a conversation takes place between the calling program and the TDCompress functions, with the calling program providing additional input when requested, and TDCompress giving output as needed until all data is processed.

The name of the parameter block structure is CALLPBLK. It is defined in the parmblok.h header file and is not reprinted here. Other definitions, including the TDCompress function prototypes and calling conventions are in the commprss.h and cpapi.h header files. Because sample programs are distributed with the TDCompress software to demonstrate the compprog and dcmpprog functions, only a brief overview is given below.

5.2.2.1 The Interactive API Conversation

The calling program must initialize the parameter block to NULLs and then put the size of (CALLPBLK) in the first field of the block. This indicates the version of CALLPBLK to the TDCompress functions.

The caller must allocate input and output buffers and set the maximum length of the output buffer in the **outbufLength** field in CALLPBLK. The maximum buffer size is 4,294,967,295 bytes (4G -1). When securing X12 or EDIFACT EDI data, the output buffer must be large enough to hold an entire, secured EDI envelope (for example, from

the ISA through the IEA segments of X12 data and from the UNA/UNB through the UNZ segments of EDIFACT data).

Desired options are requested by setting the various option flags to 1. All TDCompress command-line options are available with the exception of specific file and directory handling options. Certain text options, such as the path where the security run-time files are located, must be placed in the appropriate CALLPBLK string fields.

To begin a conversation, the caller fills the first input buffer and sets its length in the **inbufLength** field in CALLPBLK. The caller then invokes `compprog` or `dcmpprog` passing the addresses of the buffers and the address of the parameter block as parameters. Upon return, the caller must check the **rc**, **inrq** and **otrq** fields in CALLPBLK.

If **rc** is non-zero, then an error occurred and corrective action is needed. Use the value in the **rc** field to look up the error in the Error and Return guide.

If **inrq** is non-zero, then `compprog` or `dcmpprog` is making an input request for a new input buffer. The caller must fill the input buffer and set its length in the **inbufLength** field in CALLPBLK. Next, the caller must re-invoke `compprog` or `dcmpprog`.

When calling `compprog` on a record-based operating system, such as MVS and OS/400, if the **CRLF** option is specified, then the caller must supply input records one at a time. This allows `compprog` to correctly delimit multiple records.

If **otrq** is non-zero, then `compprog` or `dcmpprog` has filled the output buffer, and is making an output request. The actual length of the output buffer filled by the TDCompress function is set in the **usedOutbufLength** field in CALLPBLK. The caller must dispose of the filled output buffer and re-invoke `compprog` or `dcmpprog`.

When calling `dcmpprog` on a record-based operating system, such as MVS and OS/400, if the data is compressed with the **CRLF** option then output records are returned one at a time. This allows the caller to distinguish between multiple records.

This conversation between the calling program and the TDCompress function continues until either the caller wishes to stop compressing/securing input, or until input has been completely decompressed/unsecured.

5.2.2.2 Ending the Conversation

The beginning of the end of a conversation with the `compprog` function happens when the caller sets the **eod** field in CALLPBLK to a non-zero value. The caller sets **eod** in response to an input request from `compprog`. This is the beginning of the end of the conversation because `compprog` will more than likely need to return at least one output buffer. The conversation is actually ended when `compprog` returns with both the **inrq** field and the **otrq** field set to zero. The caller can begin a new conversation by once again

filling the input buffer, setting its length in the **inbufLength** field and invoking **compprog**.

When calling **dcmpprog**, the caller is not responsible for ending the conversation. Under normal circumstances, **dcmpprog** detects the end of compressed data automatically and, after returning the last output buffer, returns with both the **inrq** field and the **otrq** field set to zero. The **inbufLength** field is set to the length of the unprocessed data remaining in the last input buffer.

Appendix A. COMMAND FILE EXAMPLE

```
# Example of an TDClient command-file:
# (Remove leading '#' character to activate any given line)

NETWORK=bTrade.com           # Select the network to use (do only once)
USERID=myUserId             # Override the server login userid
PASSWD=myPassWord          # Override the server password

# Create some transfers, invoke other transfers already Stored
# Note use of quotes for transfer name:
TRANSFER="MY RECEIVE TEST"  # Execute existing 'MY RECEIVE TEST' transfer

# Note you can use single or double quotes, square brackets, curly braces, or
# parentheses for transfer creation,
# and for pre/post processing specification, and for TPBOOK= usage too...
transfer=(                  # Create a new transfer and then execute it
    name="MY SEND TEST"    # Name is REQUIRED!
    send=c:\autoexec.bat
    senduserid=CPINC03     # This server is case sensitive:userid in caps!
    sendclass=DOMINV      # Receive all files currently in DOMINV class
                          # COMPRESS=Y SENDASCII=Y CRLF=Y FILTER=Y
                          # SECURE=N
    pre_send= [cmdline='dir *.*' retcode=0 succeeds_always]
    post_send= [cmdline='sh -x /home/user/cleanup.ksh -h -l=60'
                each_file=y
                retcode=0
                succeeds_if_eq]
)                            # End transfer creation

Transfer= (Name=SendInvoice send=c:\inv\invoices.txt SendUserId=CPINC03
          SendAPRF=INV)

transfer= (name=ReceiveINV receive=c:\inv\new_inv.txt receiveuserid=CPINC03
          receiveclass=INV autoext=y receiveascii=y append=n)
#
# Examples of a transfer to an EDS*ELIT FTP server:
#
transfer= (name=MySend send=c:\billing\myBill.txt senduserid=MyEdsElitTP
          elit_send_filename=Billing
          elit_send_refno=BillingReferenceNumber1
          elit_send_options='FB=MyFeedback,LRECL=80,KEEP LF'
          sendascii=y )
```

```
transfer= (name=MyReceive receive=c:\inv\new_inv.txt
          receiveuserid=MyEdsElitTP
          elit_receive_filename=*billing*
          elit_receive_refno=*Number1
          elit_receive_options='CTLNO=*1*'
          autoext=y receiveascii=y append=n)
#
#
# Examples of the RECEIVE_SERVER_FILE= and DELETE_SERVER_FILE= keywords
# the specified filename 'ABC...' is the server's name for the file
#
transfer= (name=ReceiveOneFile receive=c:\myfile.txt
          receive_server_file=ABCDEFG12345)
# Note - still have to specify RECEIVE= keyword even though it is not used...
transfer= (name>DeleteOneFile receive=c:\myfile.txt
          delete_server_file=ABCDEFG12345)

# Example of transfer which receives only System Error messages from the
# IBM Information Exchange Ftp server:
transfer= (name=receiveIBMerrmsg receiveedi=incoming/err.msg
          receiveuserid="*SYSTEM*.ERRMSG")

# Tell TDCClient to get a list of available files from the server.
# The file list is written to file myaudit.log. If the auditFile keyword
# is not specified, then the file list is written to the default file, list.fil
# in the TDCClient 'temp' directory.
# Note that no other transfers will be executed during the program run if
# the "queryList" keyword is specified -- it supersedes the execution
# of all send/receive transfers.
queryList queryFile=myFiles.lst

# Tell TDCClient to get an audit report from the server showing all files
# sent and received from/to the current login
# during the period from 11/1/1999 thru 12/1/1999
# The audit report is written to the file audit.log in the TDCClient "temp"
# directory.
# Note that no other transfers will be executed during the program run if
# the "receiveAuditLogs" keyword is specified -- it supersedes the execution
# of all send/receive transfers.
receiveAuditLogs
  auditStartDate=19991101
  auditEndDate=19991201
  auditFile=myaudit.log
```

```
# Tell TDClient to create a public/private key pair, generate a
# certificate request, and send the request to the configured Certificate
# Authority for approval (as specified in the SECURITY section of the
# tdclient.ini file). In addition, specify a pass-phrase location to store
# the key-encrypting key, used to provide security-token capabilities.
# Note that no other transfers will be executed during the program run if
# the "genkeys" keyword is specified -- it supersedes the execution
# of all send/receive transfers.
genkeys passloc=[a:/mytoken.txt]

# Tell TDClient to create and execute a transfer to receive the users
# runtime files (previously generated by the configured Certificate Authority
# as specified in the SECURITY section of the tdclient.ini file).
# Once the runtime files are received, they are automatically installed.
# Note that no other transfers will be executed during the program run if
# the "receive_runtimes" keyword is specified -- it supersedes the execution
# of all send/receive transfers.
receive_runtimes

# Create some Trading Partner Address Book entries, and use one in a transfer
# Note that the network2 and mailbox2 entries are optional...
TPBOOK=( name=myPartner network1=bTrade.com mailbox1=MyPartnersMailbox )

# Note you can use square brackets, or parentheses for transfer creation,
# and for pre/post processing specification, and for TPBOOK= usage too...
TPBOOK=[ name=myOtherPartner
        network1=bTrade.com
        mailbox1=MyOtherPartnersMailbox
        network2="IGN-I/E SSL"
        mailbox2=CMAP.MyOtherPartnersIgnMailbox
      ]

# This transfer will try to send to myOtherPartner on the bTrade.com network
# and if this fails, will switch to the IGN-I/E SSL network (since auto-retry
# is enabled below...)
Transfer= (Name>ShowOff send=c:\inv\invoices.txt SendUserId=myOtherPartner
          SendAPRF>ShowOff)

# Show use of dial-up networking (Windows only)
DIAL="Dial Fedex"           # Name of existing DialUp Networking entry
# If the "Dial Fedex" DialUp Networking entry fails to connect, then
# launch the following Dialer program to try to connect:
BACKUP_DIAL_PROGRAM='C:\Program Files\AT_T_GlobalDialer\IDIALER.EXE'
AUTODIAL=Y                 # Do the dial before trying to connect!
```

```
AUTODISCONNECT=Y           # Hang up when program is done
TIMEOUT=300                 # If no connection in 3 minutes, then failed

# Show use of auto-retry
RETRY=Y                     # Enable auto-retry
MAX_RETRY=2                 # Retry twice after initial failure
RETRY_DELAY=10             # Delay 10 seconds between retries
```