CA IT Client Manager

DTSCLI Command Reference Guide

Release 12.8

CA Technologies Product References

This documentation set references to the following CA products:

- CA Advantage® Data Transport® (CA Data Transport)
- CA Asset Intelligence
- CA Asset Portfolio Management (CA APM)
- CA Common Services™
- CA Desktop Migration Manager (CA DMM)
- CA Embedded Entitlements Manager (CA EEM)
- CA Network and Systems Management (CA NSM)
- CA Patch Manager
- CA Process Automation
- CA Business Intelligence
- CA Service Desk Manager
- CA WorldView™
- CleverPath™ Reporter

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Data Transport Service Command Line Interface

Data Transport Service (DTS) provides a command line interface, the Data Transport Service Command Line Interface (DTSCLI), for network administrators to manage data transfers.

DTSCLI lets you schedule, initiate, monitor, and control data transfers. You can do the following from the command line:

- Create transfers, transfer jobs, and schedules
- Delete transfers, transfer jobs, and schedules
- Retrieve the status of transfers, transfer jobs, and schedules
- Add, insert, and remove transfers from transfer jobs
- Activate, suspend, resume, abort, and reset transfer jobs
- Adjust the priority of a transfer job
- Add, insert, and remove transfer jobs from schedules
- Enable and disable schedules
- Perform agent-to-agent transfers
- Perform managed transfers

Note: In CA ITCM, dtacli is superseded by dtscli. Although dtacli is still provided, we recommend that you migrate to dtscli.

This section contains the following topics:

Data Transfers

Data transfers that you specify using the dtscli command must be either managed transfers or agent-to-agent transfers. A single transfer must be one or the other; it cannot be both.

The following list of basic rules applies to all transfer operations:

- You *cannot* specify multiple -agent operands in the same command.
- You *cannot* specify multiple rpaths within the same -transfer operand.
- You *cannot* specify the -agent and -transfer operands together in the same command.
- If any operand or operand-parameter expression in a dtscli command includes an embedded space, enclose the entire expression in quotation marks. Always enclose the entire expression for the following in quotation marks: input and output path operands, filters, protocol_parameters, and method_parameters. This requirement also applies to other operands and parameters.
- For Data Transport Service r2 and r3 only, you can optionally use *macros* in data transfers to help systematically alter the names of input files (when they are sent) and output files (when they are received) when using the dtacli or dtscli command. For Data Transport Service, a macro is a set of characters that represents one or more variables or actions. You can make the output file name reflect the name of the sending computer, the date or time of the transfer, or other customized text.

  Note: For more information, see the Macros Policy Group topic in the Configuration Policy section of the *DSM Explorer Help*.

Important! DTS supports IPv4 broadcast and IPv4/IPv6 multicast addressing. The BCAST point-to-many protocol is only for use with IPv4 addresses. If you want to perform a broadcast-type transfer over an IPv6 network, use the MCAST protocol with the relevant IPv6 multicast address.

Managed Transfers

Managed transfers use the Transfer Object Server and optionally the Schedule Object Server to perform a transfer. To perform a managed transfer, you must specify the -transfer (-t) transfer parameters operand of the dtscli command. Under certain conditions, you may need to specify the Transfer Object Server (-tos operand). Additionally, the iuser and ruser parameters are required if the security mode of the DTS agents involved in the transfer are set to "fail." You also need to specify the input and output paths of the file or directory that is being transferred using the ipath and rpath parameters. All other managed transfer operands are optional.

More information:

Managed Transfers (TOS) (see page 65)
dtscli Command—Send Multiple Output Files (Managed) (see page 68)

Agent-to-Agent Transfers

Agent-to-agent transfers proceed from one agent to the other; no Transfer Object Server or Schedule Object Server is involved in the transfer. To specify an agent-to-agent transfer, you must specify the -agent parameters operand of the dtscli command. The iuser and ruser parameters are required if the security mode of the DTS agents involved in the transfer are set to "fail." You also need to specify the input and output paths of the file or directory that is being transferred using the ipath and rpath parameters. All other agent-to-agent transfer operands are optional.

More information:

Agent-to-Agent Transfers (see page 83)
dtscli Command—Send Multiple Output Files (Agent-to-Agent) (see page 87)

Transfer States

The following table lists the possible values that may be returned by Data Transport Service in response to your dtscli command:

Value	Description
INIT	Indicates that the transfer job was defined but not activated.
QUEUED	Indicates that the transfer job was activated, though currently being queued by the initiator.
READ_FILTERING	Indicates that initiator read filters are being applied to the transfer job.
CONNECTING	Indicates that the transfer job was activated, but no data has yet been transmitted.
RETRYING	Indicates that if a network connection has failed, the transfer job is performing session retry.
IN_PROGRESS	Indicates that the transfer job is active and in progress.
INTERRUPTED	Indicates that the transfer job is being interrupted.
WRITE_FILTERING	Indicates that responder write filters are being applied.

Value	Description
COMPLETE	Indicates that the transfer job has completed successfully.
ABORTED	Indicates that the transfer job was terminated before it completed.
ABORTING	Indicates that the transfer job is being aborted.
FAILED	Indicates that an error occurred during transfer job. The transfer job can be resumed after fixing the problem if checkpoint restart is enabled.
SUSPENDED	Indicates that data transmission for the transfer job was suspended, and can be resumed at a later stage.
RESETTING	Indicates that the transfer job is being reset.
RESUMING	Indicates that the transfer job was resumed, but no state was received from the initiating DTS agent.
CALCULATING	Indicates that the Transfer Object Server has requested a route between the initiator and responder from the Network Object Server and is waiting for a response.
WAITING_TO_CONNECT	Indicates that the transfer job is waiting for a connection to be established between the DTS agent and the Transfer job Object Server.

More information:

dtscli -transfer Parameter Group—Perform Managed Transfers (see page 18)
dtscli -job Parameter Group—Manage Transfer Jobs (see page 33)

Discreet Mode

Transfers performed in discreet mode are called *discreet transfers*. Discreet transfers are sent or received in the background when the sending and receiving computers are not heavily loaded. Based on the load, the DTS agent determines when to send the transfer and calculates the optimal transfer rate, so that the discreet transfer has minimal impact on the computer's performance and the user's productivity.

Discreet mode optimizes performance while minimizing load on all computers participating in the transfer. Therefore, discreet transfers are most often performed at a slower rate than transfers performed in normal mode. Discreet transfers are recommended for transfers that have a lower priority than normal transfers.

The rate at which a discreet transfer progresses is determined automatically and dynamically by the DTS agent. The DTS agent monitors the load (both CPU usage and the machine's network usage) on the sending and receiving computers. When a computer is loaded lightly, the DTS agent uses the spare resources available. However, if the load increases, the agent relinquishes resources, giving priority to other tasks being performed on the computer. In this way, discreet mode optimizes the computer's performance while transfers are being performed.

Restrictions

Discreet mode is not supported in Data Transport Service r1 and r1.1, Service Pack 1. For all transfers, any discreet mode specifications are ignored if either the initiating or responding DTS agent is running an earlier version of Data Transport Service that does not support discreet mode. Such transfers will still be successful, but are sent and received in normal mode.

Discreet mode applies only to point-to-point transfers; it does not apply to broadcasts, multicasts, or fanouts.

More information:

dtscli -job Parameter Group—Manage Transfer Jobs (see page 33)

Agent DiscreetMode and Transfer Priority Settings

For any DTS agent, you can specify that the agent uses discreet mode for initiating transfers, responding to transfers, or both. Additionally, for any transfer, you can specify discreet mode for the initiating computer, the responding computer, or both. Together, the agent DiscreetMode settings and the transfer priority settings determine which mode (normal or discreet) is used for sending the transfer and which mode is used for receiving the transfer.

Note: To specify the DiscreetMode setting (Y or N) for a DTS agent, see the Data Transport Agent Plugin Policy Group topic in the Configuration Policy section of the *DSM Explorer Help.*

More information:

dtscli -transfer Parameter Group—Perform Managed Transfers (see page 18)
dtscli -agent Parameter Group—Perform Agent-to-Agent Transfers (see page 48)

How Transfer Priority and Agent DiscreetMode Settings Work Together

The following table indicates how the transfer priority settings and the agent DiscreetMode settings work together to determine which mode is used for sending the transfer and which mode is used for receiving the transfer.

In this table, the first row indicates the transfer priority settings of the initiating and receiving computers. The From…To… headings for the remaining rows indicate the DTS agent DiscreetMode setting (Y or N), from the sending computer to the receiving computer. The remaining cells (the cells not in boldface) indicate whether discreet mode is used to send or receive the transfer.

Initiator_Priority (ipriority), Responder_Priority (rpriority)	**From N to N**	**From N to Y**	**From Y to N**	**From Y to Y**
Default, Default	Normal transfer	Discreet receive	Discreet send	Discreet transfer
Default, Discreet	Discreet receive	Discreet receive	Discreet transfer	Discreet transfer
Discreet, Default	Discreet send	Discreet transfer	Discreet send	Discreet transfer
Discreet, Discreet	Discreet transfer	Discreet transfer	Discreet transfer	Discreet transfer
Default, Urgent	Normal transfer	Normal transfer	Normal transfer	Normal transfer
Discreet, Urgent	Normal transfer	Normal transfer	Normal transfer	Normal transfer
Urgent, Default	Normal transfer	Normal transfer	Normal transfer	Normal transfer
Urgent, Discreet	Normal transfer	Normal transfer	Normal transfer	Normal transfer
Urgent, Urgent	Normal transfer	Normal transfer	Normal transfer	Normal transfer

This table reveals that a default transfer and an urgent transfer both mean the same kind of transfer: a normal transfer, in other words, not a discreet transfer. Similarly, a default or urgent send means a normal (not discreet) send, and a default or urgent receive means a normal (not discreet) receive.

For any transfer, any send, or any receive, the difference between a default setting and an urgent setting is that an urgent setting always overrides a discreet setting and makes it normal, while a discreet setting always overrides a normal setting and makes it discreet.

Note: The text and the table above assume that the initiator is the sending computer and the responder is the receiving computer, which is true for most transfers. However, if the initiator is the responding computer, replace the words “send,” “sending,” and so forth with the words “receive,” “receiving,” and so forth in the text above.

Chapter 2: dtscli Command—Manage Data Transfers

The dtscli command manages data transfers, transfer jobs, and schedules. The DTSCLI collects related arguments together in groups because of the large number of command line arguments. Arguments are specified left-to-right on the command line and are interpreted in terms of the most recently specified argument group.

This command has the following format:

```
dtscli [-tos parameters]
    [-sos parameters]
    [-transfer parameters]
    [-job parameters]
    [-schedule parameters]
    [-mode mode]
    [-log parameters]
    [-c comments-string]
    [@filename]
dtscli [-agent parameters]
    [-log parameters]
    [-c comments-string]
    [@filename]
dtscli -help [group[argument]]
dtscli -version
```

Note: The -a|t convention in the examples throughout this guide denote that the examples work either as a managed transfer or as an agent-to-agent transfer. Do not use -a|t literally; use -a (or-agent) and -t (or -transfer) appropriately.

-mode

(Optional) Specifies the object lifetime mode, controlling whether transfers created by the DTSCLI are activated after creation and, if so, whether they are deleted after the transfer completes.

Note: The -mode operand can be abbreviated as -m.

Valid values are as follows:

defer

Creates the transfer job but does not perform the transfer, allowing the transfer job to be attached to a schedule for deferred processing.

Example: The following example creates an agent-to-agent transfer with an object lifetime mode of defer. Defer mode creates the transfer and transfer job, but does not activate them, allowing the transfer job to be attached to a schedule for activation at a later time.

```
dtscli -tos host=SomewhereElse user=Administrator::password
        -t "ipath=branch2002::c:\daily_receipts\aug1000.xls"
        "rpath=hq050::c:\daily_receipts\aug1000.xls"
        -m defer
```

delete

Performs the transfer job and deletes the dynamically-created transfers and transfer jobs after they either complete successfully or fail.

deleteonsuccess

Performs the transfer job and deletes the dynamically-created transfers and transfer jobs only if they complete successfully.

keep

Performs the transfer job and keeps the dynamically-created transfer objects, so that the transfer job can be reactivated. Use this option when you want to use the dynamically-created transfer job or to copy it to create a new transfer job.

Example: The following example creates a managed transfer with an object lifetime mode of keep. You would specify keep mode when you want to reuse a transfer job, for example, for a file that is updated daily and sent to headquarters each evening for processing.

```
dtscli -tos host=SomewhereElse user=Administrator::password
        -t "ipath=branch2002::c:\daily_receipts\aug1000.xls"
        rpath=hq050::c:\daily_receipts\aug1000.xls"
        -m keep
```

Default: delete

-c

(Optional) Specifies a comment string.

Default: None

@filename

(Optional) Specifies the full path name of the command file to be read and used as input. You can specify both the file name and additional operands (such as -a, -mode, and so n) in the same dtscli command.

Default: None

Example: The following example executes the command lines specified in the dist_sales_rpt.txt file as input for an agent-to-agent transfer:

```
dtscli -a @dist_sales_rpt.txt
```

-help

Provides hierarchical help for each dtscli command parameter and its options.

Note: The -help operand can be abbreviated as -h.

Examples:

- To view top-level help, enter the following command:

  ```
  dtscli -h
  ```

- To view a list of the transfer parameters, enter the following command:

  ```
  dtscli -h transfer
  ```

- To view help for the transfer parameter named delivery, enter the following command:

  ```
  dtscli -h transfer delivery
  ```

-version

Displays version information for the dtscli command line interface.

Note: The -version operand can be abbreviated as -v.

Example:

```
dtscli -v
```

dtscli -tos Parameter Group—Specify the Location of the TOS

(Optional) The -tos parameter group specifies the location of the Transfer Object Server (TOS) and any user credentials needed to connect to it.

This syntax has the following format:

```
dtscli -tos [host=hostname] [user=username::password] [timeout=timeout]
    [port=port] [trustuser=username::password] [trustdomain=domain]
```

host=*hostname*

(Optional) Specifies the name of the host server to use for this transfer. If the server does not reside on the local computer, you must specify it explicitly using this parameter.

Default: *localmachine*

user=*username::password*

(Optional) Specifies the user name and password to connect to the server.

Default: None

timeout=*timeout*

(Optional) Specifies the timeout value in seconds for the transfer request to connect to the server. The timeout setting is used for all communications between the DTSCLI and the Transfer Object Server.

port=*port*

(Optional) Specifies the legacy communications port to use when connecting to the Transfer Object Server.

Default: 8198

trustuser=*username*::*password*

(Optional) Specifies the user name and password for the trusted security domain you want to access.

Note: Use the trustuser= and trustdomain= operands together to sign on to a trusted security domain and perform trusted transfers.

trustdomain=*domain*

(Optional) Specifies the name of the trusted security domain you want to access. When you have successfully signed on to the trusted security domain, you are able to perform trusted transfers in that domain. In a trusted security domain, a Data Transport Service administrator defines trusted users and the DTS agents to which they have trusted access. Trusted access means that these users can successfully activate transfers without specifying a user name and password for the Data Transport Agents in the transfer, even if the agents have their security mode set to "fail."

You can perform trusted transfers if your Data Transport Service administrator has assigned your user ID and DTS agent computer to a trusted security domain, and has completed other requirements. For more information about trusted transfers and to determine if you can perform trusted transfers, consult your Data Transport Service administrator.

Example: Perform Trusted Transfer

In the following example, the administrator has added the trusted user redlead1 to the trusted domain redblue1. The transfer can now be performed between machines yellow and green without specifying security credentials (that is, the iuser and ruser parameters) for the transfer.

```
dtscli -tos host=syd500 user=sesede::silles timeout=30 port=8898
    trustuser=redlead1::blueline trustdomain=redblue1
    -t "ipath=yellow::input_path" "rpath=green::output_path"
```

dtscli -sos Parameter Group—Specify the Location of the SOS

(Optional) The -sos parameter group specifies the location of the Schedule Object Server (SOS) and any user credentials needed to connect to it.

This syntax has the following format:

`dtscli -sos [host=`*`hostname`*`] [user=`*`username::password`*`] [timeout=`*`timeout`*`] [port=`*`port`*`]`

host=*hostname*

(Optional) Specifies the name of the host server to use for this transfer. If the server does not reside on the local computer, you must specify it explicitly using this parameter.

Default: *localmachine*

user=*username::password*

(Optional) Specifies the user name and password to connect to the server.

Note: If you specified a -default setting, then this setting overrides the default setting.

timeout=*timeout*

(Optional) Specifies the timeout value in seconds for the transfer request to connect to the server. The timeout setting is used for all communications between the DTSCLI and the Schedule Object Server.

port=*port*

(Optional) Specifies the legacy communications port to use when connecting to the Schedule Object Server.

Default: 8200

dtscli -transfer Parameter Group—Perform Managed Transfers

The -transfer parameter group creates, deletes, and retrieves the status of the transfer objects in a managed transfer.

This syntax has the following format:

```
dtscli -transfer [checkpoint=y|n] [commit=y|n] [cskip=nn|prefixtransfer]
    [fskip=nn|prefixtransfer] [data1=userdata1] [data2=userdata2]
    [delimiter_conversion=y|n] [delivery=discreet_delivery_period]
    [description=description] [direction=s|r] [expiry=discreet_expiry_period]
    [id=object_id] "ipath=host::path" [ipriority=urgent|default|discreet]
    [iuser=username::password] [iuuid=uuid] [ruuid=uuid] ["f_filters=filters"]
    ["p_filters=filters"] [label=label] [method=create|delete|status]
    ["method_parameters=method_parameters"] [output_mode=a|c|r|s|w]
    [retry_interval=session_retry_interval] [retry_limit=session_retry_limit]
    "rpath=host::path" [rpriority=urgent|default|discreet]
    [ruser=username::password]
```

Note: The -transfer operand can be abbreviated as -t.

checkpoint=y|n

(Optional) Specifies whether the checkpoint restart feature, which is an error recovery mechanism used by Data Transport Service, is enabled. Valid values are y (yes) and n (no). When checkpoint restart is enabled, if a failure occurs during a transfer, Data Transport Service attempts to restart the transfer from the point of failure. When checkpoint restart is disabled, the transfer is restarted from the beginning if a failure occurs during a transfer.

Default: None

commit=y|n

(Optional) Specifies whether the commit to disk feature is enabled. Valid values are y (yes) and n (no). If enabled, the DTS agent flushes its file output buffers to hard disk after every parcel of data has been received. If commit to disk is not specified (n), the DTS agent relies upon the operating system file I/O to flush buffers when necessary.

Default: None

cskip=*nn*|*prefixtransfer*

fskip=*nn*|*prefixtransfer*

Note: Applies to Data Transport Service r2 and r3 and CA ITCM only.

(Optional) Specifies the skip logic setting, that is, the number of transfers to skip if the transfer completes (cskip=) or fails (fskip=). Skip logic applies only to transfers in sequential transfer jobs. For each individual transfer in a sequential transfer job, skip logic allows you to skip the next *n* transfers or skip forward to another transfer when this transfer completes of fails.

nn

Specifies the number of transfers to skip, such as 1, 2, 3, or a negative number. Use a negative number to skip all remaining transfers, so that no more transfers in the job are activated.

prefixtransfer

Specifies the prefix (I, L, or N) followed by the ID, label, or name of the next transfer to be activated, as follows:

- I*transfer-ID*

  Indicates the object ID of the next transfer to be activated. For example, I657 specifies the transfer whose object ID is 657.

- L*transfer-label*

  Indicates the label of the next transfer to be activated. For example, Lrecpt_mon specifies the transfer whose label is recpt_mon.

- N*transfer-name*

  Indicates that the name of the next transfer to be activated. For example, Ntot251200 specifies the transfer whose name is tot251200.

Default: None (empty)

Note: Take special care when assigning skip logic values. No logic checking is performed: Whatever values you specify for *nn* in the cskip=*nn* and fskip=*nn* expressions are used for the skip logic values.

data1=*user_data1*

(Optional) Allows user-defined data to be associated with the transfer.

Default: None

data2=*user_data2*

(Optional) Allows user-defined data to be associated with the transfer.

Default: None

delimiter_conversion=y|n

(Optional) Specifies whether delimiter conversion is enabled. Valid values are y (yes) and n (no). Use delimiter conversion (y) only for instances when you specify the output file name in the same format as the input file name, even though the receiving computer is running on a different platform from the sending computer. In such cross-platform transfers, delimiter conversion helps to ensure that the output path name for the transferred data is correctly delimited for the receiving platform.

Delimiter conversion is intended for cross-platform transfers when the platforms involved use similar (but not exactly the same) delimiters and file naming conventions. The intended use most commonly includes one-to-many transfers between Windows and UNIX/Linux computers. Other kinds of cross-platform transfers are also supported.

If you specify the input and output path names for your transfer completely and accurately while creating or updating a transfer, do not use delimiter conversion. Use delimiter conversion only if you specify the output file name in the same format as the input file name in a cross-platform transfer.

Note: Use delimiter conversion with caution. It does not function as an error checking or validation mechanism for the file naming or delimiter conventions of the receiving platforms.

Default: None

Example: The following example depicts a fanout transfer to Windows and UNIX systems:

```
dtscli -t "ipath=c:\input.txt"
    "rpath=unixserver1::c:\tmp\output.txt"
    iuser=administrator::adminpass
    ruser=root::rootpass
    delimiter_conversion=y
    -t "ipath=c:\input.txt"
    "rpath=winserver1::c:\tmp\output.txt"
    iuser=administrator::adminpass
    ruser=root::rootpass
    delimiter_conversion=y
```

delivery=*discreet_delivery_period*

(Optional) Specifies the discreet delivery period in minutes. This option is applicable only if you specified a transfer priority of discreet for sending the transfer (using ipriority) or for receiving the transfer (using rpriority), or if you specified both. If you specify discreet mode for both sending and receiving the transfer, the same discreet delivery period applies to both.

The discreet delivery period specifies the amount of time during which the initiating or responding DTS agent tries to send or receive the transfer in discreet mode. If the discreet send or discreet receive is not completed within this period, the transfer priority automatically changes from discreet to urgent. (The transfer priority settings are explained in the ipriority and rpriority operand descriptions.)

The time begins to be counted when the transfer is activated (not created).

The discreet delivery period (delivery) is normally used with the discreet expiry period (expiry). The discreet delivery period must be less than or equal to the discreet expiry period. If you specify one of these options (delivery or expiry) but not the other, the default value is used for the missing option.

Default: -1 (unlimited)

Example: To specify a discreet transfer with a discreet delivery period of 360 minutes (6 hours) and a discreet expiry period (expiry) of 720 minutes (12 hours), enter:

```
dtscli -a|t "ipath=ukwin4q::c:\winnt\tngdts.ini"
    "rpath=philauk::c:\config\ukwinp4q\tngdts.ini"
    ipriority=discreet rpriority=discreet
    delivery=360 expiry=720
```

"description=*description*"

(Optional) Specifies a description for the transfer. If the description includes one or more spaces, enclose the entire expression in quotation marks.

Limits: 1–255 characters

Default: None

direction=s|r

(Optional) Specifies whether the initiating computer in the transfer will receive or send data. Valid values are as follows:

s

Sends data from the initiator to the responder.

r

Sends data from the responder to the initiator.

Note: When using the direction=r option, remember that the source file is on the responding computer, while the destination file is on the initiating computer. (Typically, the opposite is true.)

Default: s

Example: In the following managed transfer example, the initiating computer (NYC_HQ) receives data from the responder (STORE888):

```
dtscli -a|t "ipath=NYC_HQ::c:\receipts\store888\010398.rcp"
    "rpath=STORE888::PDEV.RRARI09.RECEIPTS" direction=r
```

expiry=*discreet_expiry_period*

(Optional) Specifies the discreet expiry period in minutes. This option is applicable only if you specified a transfer priority of discreet for sending the transfer (using ipriority) or for receiving the transfer (using rpriority), or if you specified both. If you specify discreet mode for both sending and receiving the transfer, the same discreet expiry period applies to both.

If the transfer expires, it is deleted, and no further attempts to send it occur. The expiry period (expiry) must be greater than or equal to the discreet delivery period (delivery). If you specify one of these options (expiry or delivery) but not the other, the default value is used for the missing option.

The time begins to be counted when the transfer is activated (not created).

Default: -1 (unlimited)

Example: This example specifies a discreet transfer with a discreet delivery period (delivery) of 180 minutes (3 hours) and a discreet expiry period of 360 minutes (6 hours):

```
dtscli -a|t "ipath=ukwin4q::c:\winnt\tngdts.ini"
    "rpath=philauk::c:\config\ukwinp4q\tngdts.ini"
    ipriority=discreet rpriority=discreet delivery=180 expiry=360
```

id=*object_id*

(Optional) Applies a transfer method, transfer job method, or schedule method appropriately to one, more, or all specified transfers, transfer jobs, or schedules.

Note: This operand is for existing transfers only. Do not specify an ID when creating a transfer.

Default: None

Examples:

- To apply a method (see method=create|delete|status) to a *single transfer*, use the following format:

  ```
  dtscli -t method=method id=id
  ```

- The add, insert, and remove methods let you apply a method to *multiple transfer*s, using either the id= operand or the label=*label* operand. For the id= operand, if you specify two or more transfer object IDs when applying a method, enclose the entire id= expression in quotation marks, as shown here:

  ```
  dtscli -t method=method "id=120 130 140"
  ```

- To apply a method to *all transfer objects*, specify id=all, as shown here:

  ```
  dtscli -t method=method id=all
  ```

"ipath=*host::path*"

Specifies the input file in the transfer enclosed in quotation marks. The host name is followed by two colons and the full path name of the transfer. For example:

- Windows

  ```
  "ipath=mmmww185::c:\receipts\010398.rcp"
  ```

- UNIX/Linux

  ```
  "ipath=machj17::/home/receipts/010398.rcp"
  ```

- HTTP Transfers (Internet Downloads):

  ```
  "ipath=DTA_host_name::destination_path"
  ```

Note: The ipath= operand is required.

host

Specifies the computer's name, IP address, or IPX address. For IP addresses, use periods (.) as separators; for example, 172.24.255.255. For IPX addresses, use either periods or colons (:) as separators; for example 00000005.000000000001 or 00000005:000000000001.

A DTS agent acting as an initiator receives transfer requests from this command line utility, establishes a connection to a target responder or group of responders, and sends the data. The initiator is the computer that is initiating the transfer. If the host name is omitted, the local host is used by default.

path

Specifies the full path name of the input data.

Examples:

- To send a managed transfer from a Windows computer named canoes to another Windows computer named tipper, enter the following command:

```
dtscli -a|t "ipath=canoes::C:\My Documents\Monthly Reports\apr2008.doc"
        "rpath=tipper::C:\Monthly Reports\David\apr2006.doc"
```

- To send a managed transfer between two UNIX/Linux computers named CHICAGO12 and NEWYORK20, enter the following command at the UNIX/Linux prompt:

```
dtscli -a|t "ipath=CHICAGO12::/home/ptfiles/ptcr1000.txt"
        "rpath=NEWYORK20::/home/ptfiles/ptcr1000.txt"
```

- To send a managed transfer from a Windows computer named Miami to a UNIX computer named NEWYORK20, enter the following command:

```
dtscli -a|t "ipath=Miami::C:\outgoing\file.txt"
        "rpath=NEWYORK20::/home/incoming/file.txt"
```

  Note: If running the dtscli command on UNIX/Linux, remember that the backslash character is the "escape" character. To specify a Windows path, the backslash characters must appear twice. For example:

```
dtscli -a|t "ipath=Miami::C:\\outgoing\\file.txt"
        "rpath=NEWYORK20::/home/incoming/file.txt"
```

ipriority=urgent|default|discreet

(Optional) Specifies the transfer priority on the initiating computer in the transfer: urgent, default, or discreet, as described following. The transfer priority you specify here for the initiating computer works with the transfer priority setting for the responding computer (rpriority) and the agent DiscreetMode settings (normal or discreet) for the initiating and responding computers in the transfer. (The agent DiscreetMode setting is one of the Data Transport Agent parameters, which can also be set using the DSM Explorer.)

Note: For more information, see the Data Transport Agent Plugin Policy Group topic in the Configuration Policy section of the *DSM Explorer Help.*

Together, the following transfer priority settings and agent DiscreetMode settings determine which mode (normal or discreet) is used to send the transfer and which mode is used to receive the transfer.

urgent

Sends the transfer in normal mode, even if the agent DiscreetMode setting is Y (yes) on the initiating computer.

default

Sends the transfer in normal mode if the agent DiscreetMode setting is N (no) on the initiating computer. However, if the agent DiscreetMode setting is Y (yes) on the initiating computer, then the transfer is sent in discreet mode.

discreet

Sends the transfer in discreet mode, even if the agent DiscreetMode setting is N (no) on the initiating computer.

Discreet transfers are sent and received in the background, using minimal bandwidth, when the sending and receiving computers are not heavily loaded. Based on the load, the DTS agent determines when to send the transfer and calculates the optimal transfer rate, so that the discreet transfer has a minimal impact on the computer's performance and the user's productivity.

Default: default

Example: To send and receive (rpriority) a transfer in discreet mode, enter the following command using the default values for the discreet delivery period (delivery) and the discreet expiry period (expiry):

```
dtscli -a|t "ipath=ukwin4q::c:\winnt\tngdts.ini"
    "rpath=philauk::c:\config\ukwinp4q\tngdts.ini"
    ipriority=discreet rpriority=discreet
```

iuser=*username::password*

(Optional) Specifies the user name and password for the initiating computer.

Note: This is a required parameter if the security mode is set to "fail" for the DTS agents involved in the transfer.

Default: None

Example: This example specifies the user name and password for both the initiating computer (ipath iuser) and the receiving computer (rpath ruser):

```
dtscli -a|t "ipath=NYC_HQ::c:\receipts\store888\010398.rcp"
iuser=Frank::furter
    "rpath=STORE888::PDEV.RRARI09.RECEIPTS" ruser=Molly::pitcher
```

iuuid=*uuid*

ruuid=*uuid*

(Optional) Specifies that the iuuid= and ruuid= parameters be used by the DTS agent on the responding computer in a transfer to ensure that the transfer request (request to initiate a transfer) it received was actually intended for it and not for another computer. Data Transport Service uses Dynamic Host Configuration Protocol (DHCP) more efficiently by checking the values of these parameters against the information stored for each DTS agent in the CA MDB.

Default: None

Important! Use these parameters only at the direction of your Management Database (MDB) administrator or Technical Support.

"f_filters=*filters*"

"p_filters=*filters*"

(Optional) Specifies the type of filter for a transfer. File filters (f_filters) process the data being transferred one file at a time. Parcel filters (p_filters) process the data being transferred one parcel at a time. (A parcel is a smaller unit of data within the file.)

Typically, you specify filter parameters for agent-to-agent transfers only, not for managed transfers. The only exception is any encryption filter. For all other filters that require parameters, managed transfers automatically take the values supplied by the Transfer Object Server.

To specify file filters, use the following format:

```
dtscli -a|t "ipath=ipath" "rpath=rpath" ...
    "f_filters=file_filters"
```

To specify parcel filters, use the following format:

```
dtscli -a|t "ipath=ipath" "rpath=rpath" ...
    "p_filters=parcel_filters"
```

You can specify a single read or write filter or a matching pair of read and write filters. Enclose the entire "f_filters=*file_filters*" or "p_filters=*parcel_filters*" expression in quotation marks. Separate multiple filters with a space.

Each type of filter specification requires a colon (:) in the position as shown here:

- To specify a read filter and a matching write filter, enter:

  ```
  "f|p_filters=read_filter_name[(read_filter_params)]
          :write_filter_name[(write_filter_params)]"
  ```

- To specify a read filter that has no matching write filter, enter:

  ```
  "f|p_filters=read_filter_name[(read_filter_params)]:"
  ```

- To specify a write filter that has no matching read filter, enter:

  ```
  "f|p_filters=:write_filter_name[(write_filter_params)]"
  ```

Filters are applied in the following order:

1. File read filters
2. Parcel read filters
3. Parcel write filters
4. File write filters

Read filters of the same type (file or parcel) are applied in the same order that they are specified on the command line. Write filters of the same type are applied in the reverse order that they are specified on the command line.

Default: Binary read and binary write filters

Example: The following command applies the specified filters in the following order:

- Binary read parcel filter
- Huffman compression parcel filter
- Huffman decompression parcel filter
- Binary write parcel filter

```
dtscli -a "ipath=ukwigp3n::c:\temp\getstart.pdf"
            "rpath=ukwigp3n::c:\temp\getstart2.pdf"
            "p_filters=BINARY_READ:BINARY_WRITE
    HUFFMAN_COMPRESS:HUFFMAN_UNCOMPRESS"
```

In this example, none of the filters have parameters.

Note: Each transfer must have either the text filters or the binary filters attached. If you specify any filters, the binary filters are not applied by default; therefore, you must specify either the text or binary filters explicitly in the same command.

label=*label*

(Optional) Specifies the label for the transfer you are creating. To assign a label, use the label=*label* operand and supply a non-numeric value, preferably a logical, simple name. You can specify a label only at object creation time; that is, when you enter the dtscli command that creates the object.

Using a label provides an *alias*, an alternate means besides the object ID for identifying one or more objects. If desired, you can give two or more objects the same label, so that you can apply the same method to all of those objects in a single command, by specifying a single label value instead of multiple object ID values.

A label used in this way is a two-step process:

- Create each object, using the label=*label* operand to specify the same label for each object. (As a result, each object has a unique object ID but the same label.)
- To apply the desired method to all of the objects, specify id=label instead of the "id=id1 id2 id3..." expression.

Default: None

Examples:

- This example creates three transfers that have the same label (stats):

```
dtscli -t "ipath=nnwigp4h::c:\pics\large1.bmp"
"rpath=sswigp55::c:\pics\large1.bmp" label=stats
        -t "ipath=nnwigp4h::c:\pics\large2.bmp"
"rpath=sswigp55::c:\pics\large2.bmp" label=stats
        -t "ipath=nnwigp4h::c:\pics\large3.bmp"
"rpath=sswigp55::c:\pics\large3.bmp"
                label=stats -mode defer
```

- This example deletes all three transfers:

```
dtscli -t id=stats method=delete
```

Note: This technique of specifying the same label and applying the same method to a group of objects applies to transfers, transfer jobs, and schedules.

method=create|delete|status

(Optional) Specifies the method to be applied to the transfer. Valid values are as follows:

create

Creates a transfer.

delete

Deletes one or more existing transfers. Specify the transfer(s) to be deleted, using the *id=object_id* operand, in a space-separated list enclosed in quotation marks. Use the following format:

```
method=delete "id=id1 id2 id3 ... "
```

status

Displays the status of one or more transfers. Specify the transfer(s) to be checked, using the id=*object_id* operand, in a space-separated list enclosed in quotation marks. Use the following format:

```
method=status "id=id1 id2 id3 ... "
```

The status, or *transfer state*, of each transfer you specify is returned after the command is executed.

Default: create

Examples:

- To delete three transfers whose object IDs are 442, 445, and 554, specify the following command:

  ```
  dtscli -t method=delete "id=442 445 554"
  ```

- To display the status of the same three transfers, specify the following command:

  ```
  dtscli -t method=status "id=442 445 554"
  ```

"method_parameters=*method_parameters*"

(Optional) Currently unused. The transfer method parameters are reserved for future use.

Default: None

output_mode=a|c|r|s|<u>w</u>

(Optional) Specifies the output mode. Valid values are as follows:

a

Appends data to an existing file. If the file does not exist, the transfer fails.

c

Creates a new output file. If the file already exists, the transfer fails.

r

Replaces an existing file. If the file does not exist already, the transfer fails.

s

Creates a new output file. If the file already exists, data is added onto the existing file.

<u>w</u>

Creates a new output file. If the file already exists, it is replaced.

Default: w

retry_interval=*session_retry_interval*

(Optional) Specifies the session retry interval, which is the number of seconds between retries if an attempt to connect to any computer involved in the transfer fails.

Default: 60

retry_limit=*session_retry_limit*

(Optional) Specifies the session retry limit, which is the number of retries that will be attempted if an attempt to connect to any computer involved in the transfer fails.

Default: 20

"rpath=*host::path*"

Specifies the output file in the transfer enclosed in quotation marks. The host name is followed by two colons and the full path name of the transfer. For example:

- Windows

  ```
  "rpath=mmmww188::c:\receipts\store888\010398.rcp"
  ```

- UNIX/Linux

  ```
  "rpath=uxmnuu18::/home/receipts/store888/010398.rcp"
  ```

- HTTP Transfers (Internet Downloads)

  ```
  "rpath=http_host_name::http_source_file_path_name"
  ```

Note: The rpath= operand is required.

host

Specifies the computer's name, IP address, or IPX address. For IP addresses, use periods (.) as separators; for example, 172.24.255.255. For IPX addresses, use either periods or colons (:) as separators; for example 00000005.000000000001 or 00000005:000000000001.

A DTS agent acting as a responder listens for and responds to incoming connections. These incoming connections (incoming data transfers) are initiated by the DTS agent.

path

Specifies the full path name of the output data used by the responder DTS agent.

Note: When using the dtscli command to transfer data from UNIX/Linux to Windows, you must enter two backslashes to delimit each directory and path if the output path is not enclosed in quotation marks.

Examples:

- This UNIX/Linux to Windows example encloses the rpath in quotation marks so only a single backslash delimiter is required:

```
dtscli -a|t "ipath=uxmnjj17::/home/receipts/010398.rcp"
        "rpath=mmmww188::c:\receipts\store888\010398.rcp"
```

- This example does not use quotation marks to enclose each input and output path expression, so the double backlashes are required.

```
dtscli -a|t ipath=iuxmnjj17::/home/receipts/010398.rcp
        rpath=mmmww188::c:\\receipts\\store888\\010398.rcp
```

Note: The requirement for double backslashes to delimit each directory and path applies only to the output path for transfers from UNIX/Linux to Windows.

rpriority=urgent|default|discreet

(Optional) Specifies the transfer priority on the responding computer in the transfer: urgent, default, or discreet, as described following. The transfer priority you specify here works with the transfer priority setting for the initiating computer (ipriority) and the agent DiscreetMode settings (normal or discreet) for the initiating and responding computers in the transfer. (The agent DiscreetMode setting is one of the Data Transport Agent parameters, which can also be set using the DSM Explorer.)

Note: For more information, see the Data Transport Agent Plugin Policy Group topic in the Configuration Policy section of the *DSM Explorer Help.*

Together, the following transfer priority settings and agent DiscreetMode settings determine which mode (normal or discreet) is used to send the transfer and which mode is used to receive the transfer.

urgent

Receives the transfer in normal mode, even if the agent DiscreetMode setting is Y (for yes) on the responding computer.

default

Receives the transfer in normal mode if the agent DiscreetMode setting is N (for no) on the responding computer. However, if the agent DiscreetMode setting is Y on the responding computer, then the transfer is received in discreet mode.

discreet

Receives the transfer in discreet mode, even if the agent DiscreetMode setting is N on the responding computer.

Discreet transfers are sent and received in the background, using minimal bandwidth, when the sending and receiving computers are not heavily loaded. Based on the load, the DTS agent determines when to send the transfer and calculates the optimal transfer rate, so that the discreet transfer has a minimal impact on the computer's performance and the user's productivity.

Default: default

Example: To send (ipriority) and receive a transfer in discreet mode, enter the following command using the default values for the discreet delivery period (delivery) and the discreet expiry period (expiry):

```
dtscli -a|t "ipath=ukwin4q::c:\winnt\tngdts.ini"
    "rpath=philauk::c:\config\ukwinp4q\tngdts.ini"
    ipriority=discreet rpriority=discreet
```

ruser=*username::password*

(Optional) Specifies the user name and password for the responding computer.

Note: This is a required parameter if the security mode is set to "fail" for the DTS agents involved in the transfer.

Default: None

Example: In the following example, the user specifies the user name and password for both the initiating computer (ipath) and the receiving computer (rpath):

```
dtacli -a|t "ipath=NYC_HQ::c:\receipts\store888\010398.rcp"
iuser=polly::nomial \
    "rpath=STORE888::PDEV.RRARI09.RECEIPTS" ruser=redbird::bdfeeder ... \
```

More information:

dtscli Command—Apply Filters (see page 73)
dtscli Command—Create Transfers (see page 66)
dtscli Command—Use Skip Logic (see page 71)
Transfer States (see page 9)
Agent DiscreetMode and Transfer Priority Settings (see page 11)

dtscli -job Parameter Group—Manage Transfer Jobs

(Optional) The -job parameter group creates, deletes, and invokes other methods on transfer job objects.

Note: In addition to the following operands, you may want to use the transfer (-t) skip logic operand cskip|fskip=. This operand is an optional transfer parameter that applies *only* to transfers in *sequential transfer jobs*. For each individual transfer in a sequential transfer job, skip logic allows you to skip the next *n* transfers or skip forward to another transfer when this transfer completes or fails.

This syntax has the following format:

```
dtscli -job [id=object_id] ["description=description"] [label=label]
    method=method "method_parameters=method_parameters" [sequential=y|n]
    [transfers=transfer_object_ids] [data1=userdata1] [data2=userdata2]
```

Note: The -job operand can be abbreviated as -j.

id=*object_id*

(Optional) Applies a transfer method, transfer job method, or schedule method appropriately to one, more, or all specified transfers, transfer jobs, or schedules.

Note: Required for existing transfer jobs only. Do not specify this operand when creating a transfer job (method=create).

Default: None

Examples:

- To apply a method (see method=*method*) to a single transfer job, use the following format:

  ```
  dtscli -j method=method id=id
  ```

- To specify multiple transfer jobs, use either the id= operand or the label=*label* operand. For the id= operand, if you specify two or more transfer job IDs when applying a method, enclose the entire id= expression in quotation marks, as shown here:

  ```
  dtscli -j method=method "id=120 130 140"
  ```

- For example, the following command adds two transfers (123 and 124) to a transfer job (789):

  ```
  dtscli -j id="789" method=add method_parameters="123 124"
  ```

- To apply a method to *all* transfer jobs, specify id=all, and enter the following command:

  ```
  dtscli -j method=method id=all
  ```

"description=*description*"

(Optional) Specifies a description for the transfer job. If the description includes one or more spaces, enclose the entire expression in quotation marks.

Limits: 1–255 characters

Default: None

label=*label*

(Optional) Specifies the label for the transfer job you are creating. To assign a label, use the label=*label* operand and supply a non-numeric value, preferably a logical, simple name. You can specify a label only at object creation time; that is, when you enter the dtscli command that creates the object.

Using a label provides an *alias*, an alternate means besides the object ID for identifying one or more objects. If desired, you can give two or more objects the same label, so that you can apply the same method to all of those objects in a single command, by specifying a single label value instead of multiple object ID values.

A label used in this way is a two-step process:

- Create each object, using the label=*label* operand to specify the same label for each object. (As a result, each object has a unique object ID but the same label.)
- To apply the desired method to all of the objects, specify id=label instead of the "id=id1 id2 id3..." expression.

Default: None

Examples: This example creates three jobs whose label is "red":

```
dtscli -j label=red -t "ipath=ipath" "rpath=rpath" -mode defer
    dtscli -j label=red -t "ipath=ipath" "rpath=rpath" -t "ipath=ipath"
    "rpath=rpath" -mode defer
    dtscli -j label=red -t "ipath=ipath" "rpath=rpath" -t "ipath=ipath"
    "rpath=rpath" -mode defer
```

- To activate all three, specify the following command:

  ```
  dtscli -j id=red method=activate
  ```

- To apply the same method to all transfer jobs that have a label, use the following format. The method is applied to all transfers that have a label, but not to transfers that have no label.

  ```
  dtscli -j method=method label=all
  ```

Note: This technique of specifying the same label and applying the same method to a group of objects applies to transfers, transfer jobs, and schedules.

method=*method*

Specifies the method to be applied to the transfer job you specify or the transfers you specify within a specific transfer job. The transfer job methods are:

Note: When invoking any method other than create, the only valid operands are id= (mandatory), method= (mandatory), and, if the method takes parameters, method_parameters.

abort

Halts a transfer job that is already in progress.

activate

Causes all of the transfers in a transfer job to be performed.

add

Adds the specified transfer(s) to the transfer job, at the end of the list of transfers in the job. This method takes parameters; for details, see *"method_parameters=method_parameters*".

create

Creates a new transfer job.

delete

Deletes the specified transfer job.

insert

Adds the specified transfer(s) to the transfer job, at the position you specify in the list of transfers in the job. This method takes parameters; for details, see *"method_parameters=method_parameters*".

remove

Deletes the specified transfer(s) from the transfer job. This method takes parameters; for details, see "method_parameters=*method_parameters*".

reset

Resets all transfers in the transfer job. This action enables the application using Data Transport Service to reuse and, therefore, to reactivate predefined transfers.

resume

Restarts the transfers in the transfer job that were either interrupted by a suspend method or failed, but can be restarted.

suspend

Halts the specified transfers in the transfer job. However, the transfers are retained in a state under which they can be resumed later.

status

Reports the status, or *transfer state*, of one or more transfer jobs.

urgent

Applies to discreet and default transfer jobs only. Forces the specified discreet or default transfer to become an urgent transfer. This method overrides any and all transfer priority settings and DTS agent discreet_mode settings that specify discreet mode. Consequently, the transfer is sent and received as a normal transfer.

discreet

Applies to any transfer in the job that you specify. Forces a transfer to be sent and received as discreet transfer. This method overrides any and all transfer priority settings and DTS agent discreet_mode settings that specify normal mode or urgent mode.

The transfer is processed at the time, using the transfer rate for discreet transfers determined by Data Transport Service during the lifetime of the transfer. The lifetime of a discreet transfer is determined by the delivery and expiry transfer parameters.

default

Forces the transfer priority settings to be the default. As a result, the DTS agent discreet_mode settings for the computers in the transfer determine which mode (normal or discreet) is used for sending the transfer and which mode is used for receiving the transfer.

Note: The urgent, discreet, and default methods can be applied to active transfer jobs only. Therefore, enter the following commands to activate the transfer job and make it urgent, default, or discreet:

```
dtscli -j id=nnn method=activate
```

```
dtscli -j id=nnn method=urgent|discreet|default
```

"method_parameters=*method_parameters*"

Specifies the parameters for transfer job methods. The method_parameters= parameter applies only to the add, insert, and remove transfer job methods.

Separate parameter values with spaces and enclose the entire method_parameters expression in quotation marks, as shown in the examples.

add

Specifies a list of transfer object IDs to be added.

Example: This example adds transfers 123–126 into transfer job 220.

```
dtscli -j id=220 method=add "method_parameters=123 124 125 126"
```

remove

Specifies a list of transfer object IDs to be removed.

Example: This example removes transfers 123–126 from transfer job 220.

```
dtscli -j id=220 method=remove "method_parameters=123 124 125 126"
```

insert

Inserts new transfer jobs in an existing transfer job. Used with the position parameter, which must be the first method_parameter specified; the remaining method_parameters are the object IDs of the transfers to be inserted.

Keep in mind that the position of a transfer in a transfer job is not important unless the transfers are performed sequentially (see sequential=y|n). By default, the transfers in a transfer job are performed simultaneously, not sequentially.

Examples:

- The following command inserts transfers 123–126 into transfer job 388, starting at position 1 (in front of all previously existing transfers).

```
dtscli -j id=388 method=insert "method_parameters=1 123 124 125 126"
```

- The following command inserts transfers 123–126 into transfer job 388, starting at position 0. Position 0 means that these objects are appended to the list after all previously existing transfers.

```
dtscli -j id=388 method=insert "method_parameters=0 123 124 125 126"
```

position

Specifies the position of a transfer to be added to a list of transfers in a sequential transfer job. The position number specifies when the transfer is performed. For example, if you specify position=3 for a transfer to be added to a list of sequential transfers, the new transfer will be the third transfer performed when the transfer job is executed. Specify position=0 to add the transfer at the end of the list of transfers.

Note: The position parameter is specified as the first parameter in the method_parameters argument. This parameter applies only to the insert method.

As mentioned previously, the position of a transfer in a transfer job is not important unless the transfers are performed sequentially (see sequential=y|n). By default, the transfers in a transfer job are performed simultaneously, not sequentially.

status

Reports the status, or *transfer state*, of transfer jobs. To display the status of one or more transfer jobs, use the following format:

```
dtscli -j "id=id1 id2 id3 ..." method=status
```

The string "id=id1 id2 id3 ..." specifies the object ID of the transfer jobs whose status you want to display.

Examples:

- To display only the transfer jobs that have a specific status, use the following format:

```
dtscli -j id="id1 id2 id3 ..." method=status
method_parameters=transfer_state
```

- To display all transfer job(s) whose status is COMPLETE, enter the following command:

```
dtscli -j id=all method=status "method_parameters=COMPLETE"
```

  In the resulting output, only the jobs with status=COMPLETE are reported.

Note: The method_parameters=*transfer_state* expression is case-sensitive. Specify the transfer state completely in UPPERCASE.

Default: None

sequential=y|n

(Optional) Indicates whether all transfers in the transfer job should be activated and performed sequentially. Specify sequential=y if you want all transfers in the transfer job to be activated and performed sequentially (one after the other, in the order you specify). See "method_parameters=*method_parameters*".

Specify sequential=n if you want all transfers in the transfer job to be activated and performed simultaneously (at the same time).

Default: n

transfers=*transfer_object_ids*

(Optional) Specifies the list of transfers in a transfer job. When the transfer job is activated, each transfer will be activated and performed. The transfers must already be created and you must know the object ID of each transfer. Enclose the list of transfers in a space-separated list enclosed in quotation marks.

Note: Required only for the method=add, insert, or remove parameters.

Example: The following command creates a simple transfer job containing transfers 110, 114, and 118:

```
dtscli -j transfers="110 114 118"
```

Note: The default transfer job method is create, so the create method does not need to be specified explicitly.

data1=*user_data1*

(Optional) Allows user-defined data to be associated with the transfer job.

Default: None

data2=*user_data2*

(Optional) Allows user-defined data to be associated with the transfer job.

Default: None

More information:

dtscli Command—Manage Transfer Jobs (see page 69)
dtscli Command—Use Skip Logic (see page 71)
Discreet Mode (see page 10)
Transfer States (see page 9)

dtscli -schedule Parameter Group—Manage Schedules

(Optional) The -schedule parameter group creates, deletes, and invokes other methods for schedule objects.

This syntax has the following format:

```
dtscli -schedule [id=object_id] [calendar=y|n] ["calendars=calendars"]
    ["calendar_check_times=calendar_check_times"] [description=description]
    [enable=y|n] [label=label] method=method
    ["method_parameters=method_parameters"] schedule_time=schedule_time
    ["jobs=transfer_job_object_ids"] [valid_from=valid_from] [valid_to=valid_to]
    [data1=user_data1] [data2=user_data2]
```

Note: The -schedule operand can be abbreviated as -s.

id=*object_id*

(Optional) Applies a transfer method, transfer job method, or schedule method appropriately to one, more, or all specified transfers, transfer jobs, or schedules.

Note: Required for existing schedules only. Do not specify this operand when creating a schedule (method=create).

Default: None

Examples:

- To apply a method (see method=add|insert|remove) to a single schedule, use the following format:

  ```
  dtscli -s method=method id=id
  ```

- To specify multiple IDs, use either the id= operand or the label=*label* operand. For the id= operand, if you specify two or more object IDs when applying a method, enclose the entire id= expression in quotation marks, as shown here:

  ```
  dtscli -s method=method "id=520 530 540"
  ```

- The following example adds the same transfer job (330) to two different transfer schedules (520 and 530):

  ```
  dtscli -s id="520 530" method=add method_parameters="330"
  ```

- To apply a method to *all* schedules specify id=all, enter the following command:

  ```
  dtscli -s method=method id=all
  ```

calendar=y|n

(Optional) Indicates whether the schedule is based on calendar definitions or is a one-time schedule activation. If the value is y, the schedule is based upon calendar definitions; therefore, the associated calendar properties must also be set (calendars, valid_from, valid_to, and calendar_check_times). If the value is n, the schedule is based upon a one-time activation; therefore, you must also set the schedule_time parameter.

Default: n

"calendars=*calendars*"

(Optional) Defines complex scheduling. Associating a calendar with a schedule allows repeated and complex activation times and dates to be specified, including quarterly, monthly, weekly, daily, every first Friday, and so on.

Specify a space-separated list of valid calendar names, and enclose the entire expression in quotation marks.

The calendars parameter must be used in conjunction with the valid_from and valid_to parameters.

Default: None

"calendar_check_times=*calendar_check_times*"

(Optional) Indicates the times of day when the calendars associated with the schedule are checked.

This property should contain a space-separated list of times in chronological order in the format HHMMSS, with the entire expression enclosed in quotation marks. A value indicates the times of day after 12:00 AM that the calendars associated with the schedule are checked.

Note: If this property is empty, then the calendars associated with the schedule are not checked.

Default: None

"description=*description*"

(Optional) Specifies a description for a schedule. If the description includes one or more spaces, enclose the entire expression in quotation marks.

Limits: 1–255 characters

Default: None

enable=y|n

(Optional) Indicates whether a schedule should be enabled or not at creation time. Specify enable=y to enable the schedule at creation time. Specify enable=n if you do not want to enable the schedule at creation time.

Once the schedule is created, you can change its state, using the enable and disable schedule methods.

Default: y

label=*label*

(Optional) Specifies the label for the schedule you are creating. To assign a label, use the label=*label* operand and supply a non-numeric value, preferably a logical, simple name. You can specify a label only at object creation time; that is, when you enter the dtscli command that creates the object.

Using a label provides an *alias*, an alternate means besides the object ID for identifying one or more objects. If desired, you can give two or more objects the same label, so that you can apply the same method to all of those objects in a single command, by specifying a single label value instead of multiple object ID values.

A label used in this way is a two-step process:

- Create each object, using the label=*label* operand to specify the same label for each object. (As a result, each object has a unique object ID but the same label.)
- To apply the desired method to all of the objects, specify id=label instead of the "id=id1 id2 id3..." expression.

Default: None

Example:

- To create two schedules that have the same label (june), specify the following command:

```
dtscli -s "jobs=110 120" label=june method=create
```

```
dtscli -s "jobs=130 140"label=june method=create
```

- The following command applies the same method (enable) to both schedules:

```
dtscli -s id=june method=enable
```

Note: This technique of specifying the same label and applying the same method to a group of objects applies to transfers, transfer jobs, and schedules.

method=*method*

Specifies the methods to be applied to a schedule.

Note: When invoking any method other than create, the only valid operands are id= (mandatory), method= (mandatory), and, if the method takes parameters, method_parameters.

add

Adds the specified calendars to the schedule, at the end of the list of calendars in the schedule. This method takes parameters; see method_parameters.

create

Creates a new schedule.

delete

Deletes one or more schedules.

disable

Disables one or more schedules.

enable

Enables one or more schedules.

insert

Adds the specified calendars to the schedule at the position you specify in the list of calendars in the schedule. This method takes parameters; see method_parameters.

remove

Deletes the specified calendars from the schedule. This method takes parameters; see method_parameters.

status

Reports the status of one or more schedules. This method indicates the status of the schedule, not the transfer jobs associated with the schedule. The system response is Initialized, Success, or Failure. This method takes parameters; see method_parameters.

This property is reset to Initialized when a schedule is enabled or disabled.

Default: None

"method_parameters=*method_parameters*"

(Optional) Specifies the method parameters you can apply to a schedule. Separate parameters with spaces and enclose the entire method_parameters expression in quotation marks, as shown in the examples.

add

Specifies a list of transfer job IDs to be added to an existing schedule.

Example: This example adds transfer jobs 123–126 into schedule 220:

```
dtscli -s id=220 method=add "method_parameters=123 124 125 126"
```

remove

Specifies a list of transfer job IDs to be removed from an existing schedule.

Example: This example removes transfer jobs 123–126 from schedule 220:

```
dtscli -s id=220 method=remove "method_parameters=123 124 125 126"
```

insert

Inserts new transfer jobs in an existing schedule. Used with the position parameter.

Examples:

- This example inserts transfer jobs 123–126 into schedule 388, starting at position 1 (in front of all previously existing transfer jobs).

  ```
  dtscli -s id=388 method=insert "method_parameters=1 123 124 125 126"
  ```

- This example inserts transfer jobs 123–126 into schedule 388, starting at position 0. Position 0 means that these objects are appended to the list after all previously existing transfer jobs.

  ```
  dtscli -s id=388 method=insert "method_parameters=0 123 124 125 126"
  ```

position

Specifies the position in an existing schedule where one or more new transfer jobs will be inserted. The position parameter must be the first method_parameter specified; the remaining method_parameters are the object IDs of the transfer jobs to be inserted. The position parameter applies only to the insert method.

status

Reports the status of a schedule. You can use the method=status expression alone to report the status of a schedule. Additionally, to report only a specific status, you can add the method_parameters=I|F|S expression. Valid values are I=Initialized, S=Succeeded, and F=Failed. Use the following format:

```
dtscli -s id=schedule_id method=status "method_parameters=I|F|S"
```

Data Transport Service displays the status of the schedule if it matches the method parameter (status) you specified.

Default: None

schedule_time=*schedule_time*

Specifies the time and date the schedule is to be activated. The schedule_time parameter is used when a schedule is required to be activated just once. If schedule_time is set, then the calendar parameter should be set to N. If a schedule is required to be activated periodically, the calendars parameter should be used and the calendar parameter should be set to Y.

Default: None

"jobs=*transfer_job_object_ids*"

(Optional) Identifies the transfer job objects in the schedule object. When the schedule is activated at the time specified by the calendars or schedule_time associated with the schedule, these transfer jobs are activated.

Specify a space-separated list of transfer job object identifiers, and enclose the entire expression in quotation marks, using the following format:

```
dtscli ... "jobs=132 154 185"
```

Note: Before you can add transfer jobs to a schedule, you must have already created the transfer jobs and know their object IDs.

Default: None

valid_from=*valid_from*

(Optional) Specifies the start of the time period in which the calendar specifications of the schedule object are examined.

The valid_from parameter must be used in conjunction with the valid_to calendar and calendars parameters. The valid_from parameter should specify a time earlier than the valid_to parameter.

Default: None

valid_to=*valid_to*

(Optional) Specifies the end of the time period in which the calendar specifications of the schedule object are examined.

The valid_to parameter must be used in conjunction with the valid_from calendar and calendars parameters. The valid_to parameter should specify a time later than the valid_from parameter.

Default: None

data1=*user_data1*

(Optional) Allows user-defined data to be associated with the schedule.

Default: None

data2=*user_data2*

(Optional) Allows user-defined data to be associated with the schedule.

Default: None

dtscli -log Parameter Group—Specify Logging Level and Mode

The -log parameter group specifies whether transfers are to be activated synchronously or asynchronously, and what level of logging is required.

This syntax has the following format:

```
dtscli -log interval=interval level=level mode=append|replace monitor=y|n path=path
```

Note: The -log operand can be abbreviated as -l.

interval=*interval*

Specifies the percentage interval at which progress messages are logged when monitoring a transfer.

Default: 10

Example: To specify that a message is written to the console each time 10 percent of the transfer completes, enter the following command:

```
dtscli ... -l interval=10
```

After the transfer is activated, the cumulative system response after the transfer is 30 percent complete appears:

```
transfer 457 activated ...
    10% complete ...
    20% complete ...
    30% complete ...
```

level=*level*

Specifies the logging level. Valid values are as follows:

0 = Silent

1 = Minimum

Specifies that object creation and deletion messages are not logged at this level.

2 = Medium

3 = Maximum

Default: 1

mode=append|replace

Specifies the logging mode. Valid values are as follows:

append

Appends logging messages to the existing file.

replace

Replaces the existing log file.

Default: append

monitor=y|n

Specifies whether the progress of the transfers should be monitored or not. Valid values are y (yes) and n (no). If y, a transfer is monitored and the progress of the transfer is written to the console. Consequently, no commands can be entered on the command line until the transfer completes or fails.

Default: y

path=*path*

Specifies the location of the file to which log messages should be written. If you enter a path name, enclose it in quotation marks, for example:

```
dtscli -l path="c:\logs\080100.log"
```

If you enter the standard output path (stdout), the output is displayed on the console unless you have specified another location in your shell.

Default: stdout

dtscli -agent Parameter Group—Perform Agent-to-Agent Transfers

(Optional) The -agent parameter group performs agent-to-agent transfers.

This syntax has the following format:

```
dtscli -agent [broadcast_address=broadcast_address] [checkpoint=y|n] [commit=y|n]
    [delimiter_conversion=y|n] [delivery=discreet_delivery_period]
    [direction=s|r] [expiry=discreet_expiry_period] ["f_filters=filters"]
    ["p_filters=filters"] [hop_hosts=hop_hosts] "ipath=host::path"
    [ipriority=urgent|default|discreet] [iuser=username::password]
    [multicast_address=multicast_address] [output_mode=a|c|r|s|w]
    [parcel_size=parcel_size] [protocol=protocol]
    [protocol_parameters=protocol_parameters]
    [retry_interval=session_retry_interval] [retry_limit=session_retry_limit]
    "rpath=host::path" [rpriority=urgent|default|discreet]
    [ruser=username::password] [throttle=throttle] [transfer_mode=transfer_mode]
```

Note: The -agent operand can be abbreviated as -a.

Important! DTS supports IPv4 broadcast and IPv4/IPv6 multicast addressing. The BCAST point-to-many protocol is only for use with IPv4 addresses. If you want to perform a broadcast-type transfer over an IPv6 network, use the MCAST protocol with the relevant IPv6 multicast address.

broadcast_address=*broadcast_address*

(Optional) Specifies an IPv4 broadcast (bcast) address. Valid only for broadcast transfers. When you specify this parameter, you must also specify transfer_mode=bcast and protocol=BCAST.

Note: The type of transfer performed depends on the transfer mode: A transfer mode of bcast or mcast implies a UDP broadcast or multicast transfer.

Note: For instructions for calculating broadcast and multicast addresses, see the *Data Transport Service Administration Guide*.

Default: None

Example: To send a broadcast to IP address 172.24.255.255, enter the following command, specifying a transfer mode of broadcast (bcast):

```
dtscli -a "ipath=steve::c:\steve\memo2.doc"
    "rpath=katie::c:\katie\memo2.doc"
    broadcast_address=172.24.255.255
    transfer_mode=bcast protocol=bcast
```

A broadcast requires only one send of the data to the broadcast address. The packets are broadcast to all computers on the subnet (172.24.0.0) identified by the broadcast address (172.24.255.255). If any are listening for the broadcast, they each receive the packet.

checkpoint=y|n

(Optional) Specifies whether the checkpoint restart feature, which is an error recovery mechanism used by Data Transport Service, is enabled. Valid values are y (yes) and n (no). When checkpoint restart is enabled, if a failure occurs during a transfer, Data Transport Service attempts to restart the transfer from the point of failure. When checkpoint restart is disabled, the transfer is restarted from the beginning if a failure occurs during a transfer.

Default: None

commit=y|n

(Optional) Specifies whether the commit to disk feature is enabled. Valid values are y (yes) and n (no). If enabled, the DTS agent flushes its file output buffers to hard disk after every parcel of data has been received. If commit to disk is not specified (n), the DTS agent relies upon the operating system file I/O to flush buffers when necessary.

Default: None

delimiter_conversion=y|n

(Optional) Specifies whether delimiter conversion is enabled. Valid values are y (yes) and n (no). Use delimiter conversion (y) only for instances when you specify the output file name in the same format as the input file name, even though the receiving computer is running on a different platform from the sending computer. In such cross-platform transfers, delimiter conversion helps to ensure that the output path name for the transferred data is correctly delimited for the receiving platform.

Delimiter conversion is intended for cross-platform transfers when the platforms involved use similar (but not exactly the same) delimiters and file naming conventions. The intended use most commonly includes one-to-many transfers between Windows and UNIX/Linux computers. Other kinds of cross-platform transfers are also supported.

If you specify the input and output path names for your transfer completely and accurately while creating or updating a transfer, do not use delimiter conversion. Use delimiter conversion only if you specify the output file name in the same format as the input file name in a cross-platform transfer.

Note: Use delimiter conversion with caution. It does not function as an error checking or validation mechanism for the file naming or delimiter conventions of the receiving platforms.

Default: None

Example: The following example depicts a fanout transfer to Windows and UNIX systems:

```
dtscli -a "ipath=c:\input.txt"
    "rpath=unixserver1::c:\tmp\output.txt"
    "rpath=winserver1::c:\tmp\output.txt"
    delimiter_conversion=y
    iuser=administrator::adminpass
    ruser=root::rootpass
    transfer_mode=fanout
```

delivery=*discreet_delivery_period*

(Optional) Specifies the discreet delivery period in minutes. This option is applicable only if you specified a transfer priority of discreet for sending the transfer (using ipriority) or for receiving the transfer (using rpriority), or if you specified both. If you specify discreet mode for both sending and receiving the transfer, the same discreet delivery period applies to both.

The discreet delivery period specifies the amount of time during which the initiating or responding DTS agent tries to send or receive the transfer in discreet mode. If the discreet send or discreet receive is not completed within this period, the transfer priority automatically changes from discreet to urgent. (The transfer priority settings are explained in the ipriority and rpriority operand descriptions.)

The time begins to be counted when the transfer is activated (not created).

The discreet delivery period (delivery) is normally used with the discreet expiry period (expiry). The discreet delivery period must be less than or equal to the discreet expiry period. If you specify one of these options (delivery or expiry) but not the other, the default value is used for the missing option.

Default: -1 (unlimited)

Example: To specify a discreet transfer with a discreet delivery period of 360 minutes (6 hours) and a discreet expiry period (expiry) of 720 minutes (12 hours), enter:

```
dtscli -a|t "ipath=ukwin4q::c:\winnt\tngdts.ini"
    "rpath=philauk::c:\config\ukwinp4q\tngdts.ini"
    ipriority=discreet rpriority=discreet
    delivery=360 expiry=720
```

direction=s|r

(Optional) Specifies whether the initiating computer in the transfer will receive or send data. Valid values are as follows:

s

Sends data from the initiator to the responder.

r

Sends data from the responder to the initiator.

Note: When using the direction=r option, remember that the source file is on the responding computer, while the destination file is on the initiating computer. (Typically, the opposite is true.)

Default: s

Example: In the following managed transfer example, the initiating computer (NYC_HQ) receives data from the responder (STORE888):

```
dtscli -a|t "ipath=NYC_HQ::c:\receipts\store888\010398.rcp"
"rpath=STORE888::PDEV.RRARI09.RECEIPTS" direction=r
```

expiry=*discreet_expiry_period*

(Optional) Specifies the discreet expiry period in minutes. This option is applicable only if you specified a transfer priority of discreet for sending the transfer (using ipriority) or for receiving the transfer (using rpriority), or if you specified both. If you specify discreet mode for both sending and receiving the transfer, the same discreet expiry period applies to both.

If the transfer expires, it is deleted, and no further attempts to send it occur. The expiry period (expiry) must be greater than or equal to the discreet delivery period (delivery). If you specify one of these options (expiry or delivery) but not the other, the default value is used for the missing option.

The time begins to be counted when the transfer is activated (not created).

Default: -1 (unlimited)

Example: This example specifies a discreet transfer with a discreet delivery period (delivery) of 180 minutes (3 hours) and a discreet expiry period of 360 minutes (6 hours):

```
dtscli -a|t "ipath=ukwin4q::c:\winnt\tngdts.ini"
    "rpath=philauk::c:\config\ukwinp4q\tngdts.ini"
ipriority=discreet rpriority=discreet delivery=180 expiry=360
```

"f_filters=*filters*"

"p_filters=*filters*"

(Optional) Specifies the type of filter for a transfer. File filters (f_filters) process the data being transferred one file at a time. Parcel filters (p_filters) process the data being transferred one parcel at a time. (A parcel is a smaller unit of data within the file.)

Typically, you specify filter parameters for agent-to-agent transfers only, not for managed transfers. The only exception is any encryption filter. For all other filters that require parameters, managed transfers automatically take the values supplied by the Transfer Object Server.

To specify file filters, use the following format:

```
dtscli -a|t "ipath=ipath" "rpath=rpath" ...
    "f_filters=file_filters"
```

To specify parcel filters, use the following format:

```
dtscli -a|t "ipath=ipath" "rpath=rpath" ...
    "p_filters=parcel_filters"
```

You can specify a single read or write filter or a matching pair of read and write filters. Enclose the entire "f_filters=*file_filters*" or "p_filters=*parcel_filters*" expression in quotation marks. Separate multiple filters with a space.

Each type of filter specification requires a colon (:) in the position as shown here:

- To specify a read filter and a matching write filter, enter:

  ```
  "f|p_filters=read_filter_name[(read_filter_params)]
          :write_filter_name[(write_filter_params)]"
  ```

- To specify a read filter that has no matching write filter, enter:

  ```
  "f|p_filters=read_filter_name[(read_filter_params)]:"
  ```

- To specify a write filter that has no matching read filter, enter:

  ```
  "f|p_filters=:write_filter_name[(write_filter_params)]"
  ```

Filters are applied in the following order:

1. File read filters
2. Parcel read filters
3. Parcel write filters
4. File write filters

Read filters of the same type (file or parcel) are applied in the same order that they are specified on the command line. Write filters of the same type are applied in the reverse order that they are specified on the command line.

Default: Binary read and binary write filters

Example: The following command applies the specified filters in the following order:

- Binary read parcel filter
- Huffman compression parcel filter
- Huffman decompression parcel filter
- Binary write parcel filter

```
dtscli -a "ipath=ukwigp3n::c:\temp\getstart.pdf"
            "rpath=ukwigp3n::c:\temp\getstart2.pdf"
            "p_filters=BINARY_READ:BINARY_WRITE
    HUFFMAN_COMPRESS:HUFFMAN_UNCOMPRESS"
```

In this example, none of the filters have parameters.

Note: Each transfer must have either the text filters or the binary filters attached. If you specify any filters, the binary filters are not applied by default; therefore, you must specify either the text or binary filters explicitly in the same command.

hop_hosts=*hop_hosts*

(Optional) Performs a hop through the host computers for all output files. You would specify a hop, for example, if the target receiving computer or computers (end nodes) were not directly accessible to the initiator but were accessible through the hop computer.

If you must specify one or more additional hop computers, enter each computer name. Separate computer names with a space. Make sure each hop computer has enough disk space to accommodate the transfer; otherwise, the transfer fails.

When executed, the transfer travels through the hop computers in the order you specify them on the command line. You can specify a maximum of 100 output files for a single transfer, and a maximum of 25 hop computers per transfer.

Note: If you specify a hop to a different platform (other than the one the initiator is running), and the transfer protocol you specify is not supported by the platform on which the hop computer is running, the transfer fails. For information about the protocols supported on all platforms, see protocol=.

Default: None

Example: The following example specifies a transfer_mode of point-to-point (p2p) and specifies a hop through the computer 172.24.213.47 for two output files:

```
dtscli -a "ipath=ddilgp78::c:\simpex.txt" hop_hosts=172.24.213.47
    "rpath=GGilyphy::c:\qa\in\simpex.txt"
    "rpath=GGilyphx::c:\qa\in\simpex.txt"
    transfer_mode=p2p
```

"ipath=*host::path*"

Specifies the input file in the transfer enclosed in quotation marks. The host name is followed by two colons and the full path name of the transfer. For example:

- Windows

  ```
  "ipath=mmmww185::c:\receipts\010398.rcp"
  ```

- UNIX/Linux

  ```
  "ipath=machj17::/home/receipts/010398.rcp"
  ```

- HTTP Transfers (Internet Downloads):

  ```
  "ipath=DTA_host_name::destination_path"
  ```

Note: The ipath= operand is required.

host

Specifies the computer's name, IP address, or IPX address. For IP addresses, use periods (.) as separators; for example, 172.24.255.255. For IPX addresses, use either periods or colons (:) as separators; for example 00000005.000000000001 or 00000005:000000000001.

A DTS agent acting as an initiator receives transfer requests from this command line utility, establishes a connection to a target responder or group of responders, and sends the data. The initiator is the computer that is initiating the transfer. If the host name is omitted, the local host is used by default.

path

Specifies the full path name of the input data.

Examples:

- To send a managed transfer from a Windows computer named canoes to another Windows computer named tipper, enter the following command:

  ```
  dtscli -a|t "ipath=canoes::C:\My Documents\Monthly Reports\apr2008.doc"
          "rpath=tipper::C:\Monthly Reports\David\apr2006.doc"
  ```

- To send a managed transfer between two UNIX/Linux computers named CHICAGO12 and NEWYORK20, enter the following command at the UNIX/Linux prompt:

  ```
  dtscli -a|t "ipath=CHICAGO12::/home/ptfiles/ptcr1000.txt"
          "rpath=NEWYORK20::/home/ptfiles/ptcr1000.txt"
  ```

- To send a managed transfer from a Windows computer named Miami to a UNIX computer named NEWYORK20, enter the following command:

```
dtscli -a|t "ipath=Miami::C:\outgoing\file.txt"
        "rpath=NEWYORK20::/home/incoming/file.txt"
```

  Note: If running the dtscli command on UNIX/Linux, remember that the backslash character is the "escape" character. To specify a Windows path, the backslash characters must appear twice. For example:

```
dtscli -a|t "ipath=Miami::C:\\outgoing\\file.txt"
        "rpath=NEWYORK20::/home/incoming/file.txt"
```

ipriority=urgent|<u>default</u>|discreet

(Optional) Specifies the transfer priority on the initiating computer in the transfer: urgent, default, or discreet, as described following. The transfer priority you specify here for the initiating computer works with the transfer priority setting for the responding computer (rpriority) and the agent DiscreetMode settings (normal or discreet) for the initiating and responding computers in the transfer. (The agent DiscreetMode setting is one of the Data Transport Agent parameters, which can also be set using the DSM Explorer.)

Note: For more information, see the Data Transport Agent Plugin Policy Group topic in the Configuration Policy section of the *DSM Explorer Help.*

Together, the following transfer priority settings and agent DiscreetMode settings determine which mode (normal or discreet) is used to send the transfer and which mode is used to receive the transfer.

urgent

Sends the transfer in normal mode, even if the agent DiscreetMode setting is Y (yes) on the initiating computer.

<u>default</u>

Sends the transfer in normal mode if the agent DiscreetMode setting is N (no) on the initiating computer. However, if the agent DiscreetMode setting is Y (yes) on the initiating computer, then the transfer is sent in discreet mode.

discreet

Sends the transfer in discreet mode, even if the agent DiscreetMode setting is N (no) on the initiating computer.

Discreet transfers are sent and received in the background, using minimal bandwidth, when the sending and receiving computers are not heavily loaded. Based on the load, the DTS agent determines when to send the transfer and calculates the optimal transfer rate, so that the discreet transfer has a minimal impact on the computer's performance and the user's productivity.

Default: default

Example: To send and receive (rpriority) a transfer in discreet mode, enter the following command using the default values for the discreet delivery period (delivery) and the discreet expiry period (expiry):

```
dtscli -a|t "ipath=ukwin4q::c:\winnt\tngdts.ini"
    "rpath=philauk::c:\config\ukwinp4q\tngdts.ini"
    ipriority=discreet rpriority=discreet
```

iuser=*username::password*

(Optional) Specifies the user name and password for the initiating computer.

Note: This is a required parameter if the security mode is set to "fail" for the DTS agents involved in the transfer.

Default: None

Example: This example specifies the user name and password for both the initiating computer (ipath iuser) and the receiving computer (rpath ruser):

```
dtscli -a|t "ipath=NYC_HQ::c:\receipts\store888\010398.rcp"
iuser=Frank::furter
    "rpath=STORE888::PDEV.RRARI09.RECEIPTS" ruser=Molly::pitcher
```

multicast_address=*multicast_address*

(Optional) Specifies an IPv4 or IPv6 multicast (mcast) address. Valid multicast addresses for use with Data Transport Service fall in the range 224.0.1.0 to 238.255.255.255 (the "Global Scope Addresses"). Valid only for UDP multicast transfers. When you specify this parameter, you must also specify transfer_mode=mcast and protocol=MCAST.

Note: For instructions for calculating broadcast and multicast addresses, see the *Data Transport Service Administration Guide*.

Default: None

Examples:

- To send a multicast to IP address 224.1.2.3, enter the following command, specifying a transfer mode of mcast. (This IP address is invalid and is used for illustration only.)

  ```
  dtscli -a "ipath=steve::c:\steve\memo2.doc"
          "rpath=katie::c:\katie\memo2.doc"
          multicast_address=224.1.2.3
          transfer_mode=mcast protocol=mcast
  ```

- To send a broadcast-type transfer over an IPv6 network to FF15:0:1:1:1::45, enter the following command, specifying a transfer mode of mcast:

  ```
  dtscli -a "ipath=steve::c:\steve\memo2.doc"
          "rpath=katie::c:\katie\memo2.doc"
          multicast_address=FF15:0:1:1:1::45
          transfer_mode=mcast protocol=mcast
  ```

output_mode=a|c|r|s|<u>w</u>

(Optional) Specifies the output mode. Valid values are as follows:

a

Appends data to an existing file. If the file does not exist, the transfer fails.

c

Creates a new output file. If the file already exists, the transfer fails.

r

Replaces an existing file. If the file does not exist already, the transfer fails.

s

Creates a new output file. If the file already exists, data is added onto the existing file.

<u>w</u>

Creates a new output file. If the file already exists, it is replaced.

Default: w

parcel_size=*parcel_size*

(Optional) Specifies the maximum parcel size, in bytes, that can be sent by the initiating DTS agent in this transfer. The minimum parcel size is 128 bytes and, theoretically, the maximum parcel size is limited only by the amount of virtual memory available.

Default: 524,288 bytes

Example: This example sends a point-to-point transfer from a Windows computer named benjamin to one named joseph, using a maximum parcel size of 50,000 bytes:

```
dtscli -a "ipath=benjamin::c:\nhmh.c" parcel_size=50000
    "rpath=joseph::c:\nhmh.c"
```

protocol=*protocol*

(Optional) Specifies the protocol for agent-to-agent transfers. Protocols require parameters under some circumstances. For details, see the protocol_parameters operand.

Valid values are as follows:

TCP

Specifies Transmission Control Protocol (see page 89).

UDP

Specifies User Datagram Protocol (see page 92).

SPX

Specifies Sequenced Packet Exchange (see page 94).

CPIC

Specifies Common Programming Interface for Communications (see page 93).

PPP

Specifies Point to Point Protocol (see page 90). Valid only for point-to-point transfers over standard telephone lines, ISDN, and other high-speed connections.

HTTP

Specifies Hypertext Transfer Protocol (see page 80). HTTP transfers (Internet downloads) require a specific syntax format.

BCAST

Specifies broadcast. When using this protocol, you must also specify broadcast_address=broadcast_address and transfer_mode=bcast. (For use with IPv4 only.)

MCAST

Specifies multicast. When using this protocol, you must also specify multicast_address=multicast_address and transfer_mode=mcast. (For use with IPv4 and IPv6.)

Default: TCP

Note: If desired, you can use the protocol= and protocol_parameters operands of the dtscli command to set the protocol and protocol parameters used for a specific transfer only. However, to set the default protocol and default protocol parameters for *all* transfers involving a specific computer, you must use either WorldView or the DSM Explorer to modify the computer's protocol-related settings. For instructions for modifying these settings using WorldView, see the *Data Transport Service WorldView Administration Client Help*, and for instructions using the DSM Explorer, see the Configuration Policy section of the *DSM Explorer Help*.

protocol_parameters=*protocol_parameters*

(Optional) Specifies protocol-specific parameters. Typically, the following requirements and restrictions are applicable:

- Address-related parameters are required.
- You should check the default port numbers used by Data Transport Service and make sure they do not conflict with any port numbers already being used. If you find a conflict, change the Data Transport Service port number to one that is not already being used.
- If a computer can connect to other computers only by using the CPI-C protocol, then it can function only as a responder in a data transfer using the dtscli command.

For further details, including which parameters are required, choose the desired protocol to view its parameters:

HTTP (see page 80)

PPP (Windows) (see page 90)

PPP (Solaris Solstice) (see page 90)

CPIC (see page 93)

TCP (see page 89)

UDP (see page 92)

SPX (see page 94)

retry_interval=*session_retry_interval*

(Optional) Specifies the session retry interval, which is the number of seconds between retries if an attempt to connect to any computer involved in the transfer fails.

Default: 60

retry_limit=*session_retry_limit*

(Optional) Specifies the session retry limit, which is the number of retries that will be attempted if an attempt to connect to any computer involved in the transfer fails.

Default: 20

"rpath=*host::path*"

Specifies the output file in the transfer enclosed in quotation marks. The host name is followed by two colons and the full path name of the transfer. For example:

- Windows

  ```
  "rpath=mmmww188::c:\receipts\store888\010398.rcp"
  ```

- UNIX/Linux

  ```
  "rpath=uxmnuu18::/home/receipts/store888/010398.rcp"
  ```

- HTTP Transfers (Internet Downloads)

  ```
  "rpath=http_host_name::http_source_file_path_name"
  ```

Note: The rpath= operand is required.

host

Specifies the computer's name, IP address, or IPX address. For IP addresses, use periods (.) as separators; for example, 172.24.255.255. For IPX addresses, use either periods or colons (:) as separators; for example 00000005.000000000001 or 00000005:000000000001.

A DTS agent acting as a responder listens for and responds to incoming connections. These incoming connections (incoming data transfers) are initiated by the DTS agent.

path

Specifies the full path name of the output data used by the responder DTS agent.

Note: When using the dtscli command to transfer data from UNIX/Linux to Windows, you must enter two backslashes to delimit each directory and path if the output path is not enclosed in quotation marks.

Examples:

- This UNIX/Linux to Windows example encloses the rpath in quotation marks so only a single backslash delimiter is required:

  ```
  dtscli -a|t "ipath=uxmnjj17::/home/receipts/010398.rcp"
          "rpath=mmmww188::c:\receipts\store888\010398.rcp"
  ```

- This example does not use quotation marks to enclose each input and output path expression, so the double backlashes are required.

  ```
  dtscli -a|t ipath=iuxmnjj17::/home/receipts/010398.rcp
          rpath=mmmww188::c:\\receipts\\store888\\010398.rcp
  ```

Note: The requirement for double backslashes to delimit each directory and path applies only to the output path for transfers from UNIX/Linux to Windows.

rpriority=urgent|default|discreet

(Optional) Specifies the transfer priority on the responding computer in the transfer: urgent, default, or discreet, as described following. The transfer priority you specify here works with the transfer priority setting for the initiating computer (ipriority) and the agent DiscreetMode settings (normal or discreet) for the initiating and responding computers in the transfer. (The agent DiscreetMode setting is one of the Data Transport Agent parameters, which can also be set using the DSM Explorer.)

Note: For more information, see the Data Transport Agent Plugin Policy Group topic in the Configuration Policy section of the *DSM Explorer Help.*

Together, the following transfer priority settings and agent DiscreetMode settings determine which mode (normal or discreet) is used to send the transfer and which mode is used to receive the transfer.

urgent

Receives the transfer in normal mode, even if the agent DiscreetMode setting is Y (for yes) on the responding computer.

default

Receives the transfer in normal mode if the agent DiscreetMode setting is N (for no) on the responding computer. However, if the agent DiscreetMode setting is Y on the responding computer, then the transfer is received in discreet mode.

discreet

Receives the transfer in discreet mode, even if the agent DiscreetMode setting is N on the responding computer.

Discreet transfers are sent and received in the background, using minimal bandwidth, when the sending and receiving computers are not heavily loaded. Based on the load, the DTS agent determines when to send the transfer and calculates the optimal transfer rate, so that the discreet transfer has a minimal impact on the computer's performance and the user's productivity.

Default: default

Example: To send (ipriority) and receive a transfer in discreet mode, enter the following command using the default values for the discreet delivery period (delivery) and the discreet expiry period (expiry):

```
dtscli -a|t "ipath=ukwin4q::c:\winnt\tngdts.ini"
    "rpath=philauk::c:\config\ukwinp4q\tngdts.ini"
    ipriority=discreet rpriority=discreet
```

ruser=*username::password*

(Optional) Specifies the user name and password for the responding computer.

Note: This is a required parameter if the security mode is set to "fail" for the DTS agents involved in the transfer.

Default: None

Example: In the following example, the user specifies the user name and password for both the initiating computer (ipath) and the receiving computer (rpath):

```
dtacli -a|t "ipath=NYC_HQ::c:\receipts\store888\010398.rcp"
iuser=polly::nomial \
    "rpath=STORE888::PDEV.RRARI09.RECEIPTS" ruser=redbird::bdfeeder ... \
```

throttle=*throttle*

(Optional) Specifies the throttle factor, the amount of network bandwidth used when initiating data transfers from this computer.

Valid values are 0 (no throttling, that is, no forced delay between parcel sends) to 100. Each increment increases the delay interval between parcel send operations by 50 milliseconds. For example, if you specify 20, a 1-second delay occurs between parcel sends. If zero is specified, no forced delay occurs between parcel sends.

Default: 0

Example: To specify a throttle factor of one-half second, enter:

```
dtscli -a "ipath=temo33::c:\jpegs\cows.jpeg" protocol=spx
    throttle=10 "rpath=kyto202::c:\jpegs\cows.jpeg"
```

transfer_mode=*transfer_mode*

(Optional) Specifies the transfer mode. A transfer mode of bcast or mcast implies a UDP broadcast or multicast transfer. Valid values are as follows:

p2p

Specifies point-to-point mode.

PPP

Specifies Point to Point Protocol (PPP) mode. Valid only for point-to-point transfers over standard telephone lines, ISDN, and other high-speed connections.

fanout

Specifies fanout mode.

bcast

Specifies broadcast mode. Valid only for IPv4 broadcast transfers. When using this transfer mode, you must also specify broadcast_address=broadcast_address and protocol=bcast.

mcast

Specifies multicast mode. Valid for IPv4 and IPv6 multicast transfers. When using this transfer mode, you must also specify multicast_address=multicast_address and protocol=mcast.

Default: p2p

Example: This example sends a fanout from a computer named LondonHQ to satellite computers named Local100 through Local110:

```
dtscli -a "ipath=LondonHQ::c:\mandates\memo5005.doc"
    "rpath=Local100::c:\mandates\memo5005.doc"
    "rpath=Local101::c:\mandates\memo5005.doc"
    "rpath=Local102::c:\mandates\memo5005.doc"
    .
    .
    .
    "rpath=Local110::c:\mandates\memo5005.doc"
    transfer_mode=fanout
```

More information:

dtscli Command—Apply Filters (see page 73)
Agent DiscreetMode and Transfer Priority Settings (see page 11)

Chapter 3: Managed Transfers (TOS)

Data transfers can be created in and managed by the Transfer Object Server (TOS) using the -transfer (or -t) parameter.

The following examples illustrate basic managed transfer operations.

Example: Perform a Simple Managed Transfer

The following example is a simple managed transfer for two Windows computers. Note the use of the required -transfer parameters.

```
dtscli -tos host=hostname user=username::password
-t "ipath=wstabp3j::C:\My Documents\Monthly Reports\apr2008.doc"
    "rpath=wstabp3n::C:\Monthly Reports\David\apr2008.doc"
```

Example: Perform Multiple Transfers

To perform multiple managed transfers, specify multiple -transfer parameters. The following example sends two files from the same computer to the same recipient. You can also send the same files multiple times to different recipients, or you can send different files from different computers.

```
dtscli -tos host=hostname user=username::password
    -t "ipath=jones201::c:\mandates\memo5005.doc"
    "rpath=smith110::c:\mandates\memo5005.doc"
    -t "ipath=jones201::c:\mandates\memo5010.doc"
    "rpath=smith110::c:\mandates\memo5010.doc"
```

This section contains the following topics:

dtscli Command—Create Transfers

The dtscli -transfer (or -t) command creates a transfer.

This following syntax is used for simple transfers:

```
dtscli -transfer "ipath=InitiatorMachine::SourceFile"
    "rpath=Respondermachine::DestinationFile"
    iuser=InitiatorUser::Password
    ruser=ResponderUser::Password
    Release 12.5.00
```

Example: Create and Activate a Transfer

This example first creates the transfer (ID 270), places it in a transfer job (ID 271), and activates the transfer job.

```
dtscli -transfer "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    iuser=dts::dts
    ruser=dts::dts
```

Sample Output Messages

```
Transfer 270 created.
Transfer job 271 created.
Transfer job 271 activated.
Transfer job 271 state: STARTING
Transfer job 271 state: IN_PROGRESS
```

Note: Your transfer and job IDs may differ from those in this example.

Example: Create and Keep a Transfer in the TOS

The default behavior of dtscli -transfer is to immediately activate the transfer and delete it after it has successfully completed. This may not be the desired behavior.

The -mode parameter group lets you specify a mode of control: whether the transfers are executed immediately and, if they are, whether the underlying objects are deleted once the transfers are complete.

This example keeps a transfer after it has been activated.

```
dtscli -transfer "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    iuser=dts::dts
    ruser=dts::dts
    -mode keep
```

Sample Output Messages

```
Transfer 292 created.
Transfer job 293 created.        <- Transfer job ID
Transfer job 293 activated.
Transfer job 293 state: STARTING
Transfer job 293 state: IN_PROGRESS
```

The output tells you the ID of the transfer and transfer job. (Making a note of the transfer job's ID is useful.)

The transfer is created, activated, but not deleted. If you have the ADT Transfer Client installed, open your transfer client; you will be able to see the transfer and the transfer job that have been created.

Example: Create but Do Not Activate a Transfer

This example creates a new transfer, but does not add the transfer to a job and does not activate the transfer.

```
dtscli -transfer "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    iuser=dts::dts
    ruser=dts::dts
    -mode defer
```

Sample Output Messages

```
Transfer 291 created.
```

The output tells you the ID of the transfer that has been created. Make a note of the transfer's ID, as it is used in another example.

The transfer is created but activation is deferred. Again, use the transfer client and view the transfer that has been created. You will find it in "unassigned transfers". You can also use dtscli to view the transfer:

```
dtscli -transfer method=status id=all
```

More information:

dtscli -transfer Parameter Group—Perform Managed Transfers (see page 18)

dtscli Command—Send Multiple Output Files (Managed)

The dtscli -transfer (or -t) command can be used to send multiple output files using a managed transfer. You must enter the -transfer operand once for each input and output file, separated by a space. Using a single dtscli command, you can specify a maximum of 16 transfers.

Example: Windows Managed Transfer

This example specifies only one output file.

```
dtscli -t "ipath=steve::c:\misc\secrets.doc"
    "rpath=katie::c:\misc\secrets.doc"
```

Important! When creating multiple managed transfers, specify the -tos, -sos, -job, and -schedule operands one time only, no matter how many transfers are specified. However, you must specify the -transfer operand and all of its parameters (such as ipath=, rpath=, direction=, and so on) once for each transfer (each -transfer expression).

Examples: Managed Transfers for Every Supported Protocol Except CPI-C

After you create a new transfer (using -transfer), dtscli does not carry over values from the previous transfer. The following example specifies three transfers: the first transfer uses delimiter conversion (delimiter_conversion=y), but the other two transfers do not specify this parameter, so they use the default (delimiter_conversion=n). This example is valid for Windows, and it sends the same input file to three different computers. A remote Transfer Object Server is used for all transfers.

```
dtscli -tos host=wsiwig
    -t "ipath=steve::c:\misc\secrets.doc" "rpath=margaret::c:\misc\secrets.doc"
    delimiter_conversion=y
    -t "ipath=steve::c:\misc\secrets.doc" "rpath=katie::c:\misc\secrets.doc"
    -t "ipath=steve::c:\misc\secrets.doc" "rpath=roshni::c:\misc\secrets.doc" ...
```

In the example above, the same input file (with the same path name) is sent to each receiving computer. If desired, you could send two or more different input files, change the path name of the output files, or both.

This example sends three different input files, and a new file name is specified for each output file.

```
dtscli -t "ipath=steve::c:\receipts\monday.doc"
"rpath=margaret::c:\monday\mon_fs.doc"
    -t "ipath=steve::c:\receipts\tuesday.doc"
    "rpath=margaret::c:\tuesday\tues_fs.doc"
    -t "ipath=steve::c:\receipts\wednesday.doc"
    "rpath=margaret::c:\wednesday\wed_fs.doc" ...
```

Note: If any computer in the transfer has its security mode set to "fail," make sure before sending the transfer that you specify the correct user names and passwords for all receiving computers. Alternatively, make sure that the security mode is quiet or warn on any receiving computer for which you have not specified a valid user ID and password. Otherwise, the security features of the DTS agent on the receiving computer cause the transfer to that computer to fail.

More information:

Managed Transfers (see page 8)

dtscli Command—Manage Transfer Jobs

The dtscli -job (or -j) command manages transfer jobs. There are different methods that can be applied to transfer jobs. For a list of methods available, use *one* of the following help commands:

```
dtscli -help job
dtscli -help job method
```

Example: Activate a Transfer Job

This example reactivates a transfer job that was created and kept earlier:

```
dtscli -job method=activate id=transfer_job_id
```

Example: Reset a Transfer Job

This example resets the transfer job using the reset method:

```
dtscli -job method=reset id=transfer_job_id
```

Example: Abort a Transfer Job

This example uses the abort method to abort a job that is in progress:

```
dtscli -job method=abort id=transfer_job_id
```

Alternatively, you can press CTRL-C to abort the current operation.

Sample Output Messages

```
Transfer job 293 activated.
Transfer job 293 state: STARTING
Transfer job 293 state: IN_PROGRESS
Transfer job 293 state: ABORTED
```

Example: Create a New Transfer Job

This example creates a new transfer job:

```
dtscli -job method=create
```

Sample Output Message

```
Transfer job 305 created.
```

Instead of jotting down the job transfer ID, you can use a name alias, or *label*. By setting the label of this job, you can refer to the job by the label.

Example: Add Transfers to the Transfer Job

This example adds transfers to a transfer job that was created but not activated, and requires the transfer ID of the earlier example.

```
dtscli -job id=transfer_job_id method=add method_parameters=your_transfer_id
```

More information:

dtscli -job Parameter Group—Manage Transfer Jobs (see page 33)

dtscli Command—Use Skip Logic

Skip logic is most useful when the success or failure of one or more specific transfers determines which other transfers (if any) should be activated and which other transfers (if any) should be skipped. You can specify one or both skip logic settings (CSKIP= and FSKIP=) for a transfer.

The following table lists some representative skip logic settings and their results:

Values	If Transfer Completes	If Transfer Fails
cskip=1 fskip=2	Next transfer is skipped.	Next two transfers are skipped.
*fskip=0	Next transfer is activated.	Next transfer is activated.
*cskip=-1	No more transfers are activated.	No more transfers are activated.
fskip=I450	Next transfer is activated.	The next transfers until transfer 450 are skipped. Transfer 450 is activated.
fskip= LtransferOne	Next transfer is activated.	The next transfers until the transfer with the label “TransferOne” are skipped. This transfer is activated.
fskip= NtransferTwo	Next transfer is activated.	The next transfers until the transfer with the name “TransferTwo” are skipped. This transfer is then activated.

*In these cases, the 0 or -1 value specifies that there is no skip logic to perform.

More information:

dtscli -transfer Parameter Group—Perform Managed Transfers (see page 18)
dtscli -job Parameter Group—Manage Transfer Jobs (see page 33)

Use Skip Logic in a Sequential Transfer Job

Given the following example: If Transfer A succeeds, skip Transfers B-D and perform Transfers E-G; but if Transfer A fails, perform Transfers B-D and skip Transfers E-G. If any other transfer besides Transfer A fails, abort the entire job.

To create a transfer job for a sequential job

1. Create a sequential transfer job and add Transfers A-G to the job, in sequential order.
2. For Transfer A, code the following skip logic setting:

   ```
   dtscli "ipath=ipath" "rpath=rpath" ... -t cskip=3 fskip=0
   ```

3. For Transfer B, code the following skip logic setting:

   ```
   dtscli "ipath=ipath" "rpath=rpath" ... -t cskip=0 fskip=-1
   ```

4. For Transfer C, code the following skip logic setting:

   ```
   dtscli "ipath=ipath" "rpath=rpath" ... -t cskip=0 fskip=-1
   ```

   If Transfers B and C succeed, the next transfer is activated, but if either transfer fails, the entire job aborts according to the original requirements for this example.

5. For Transfer D, code the following skip logic setting:

   ```
   dtscli "ipath=ipath" "rpath=rpath" ... -t cskip=-1 fskip=-1
   ```

   Note that Transfer D specifies -1 for both the cskip and fskip values. As a result, if Transfers B-D are performed, no more transfers are performed after Transfer D succeeds or fails, according to the original requirements for this example.

6. For Transfer E, code the following skip logic setting:

   ```
   dtscli "ipath=ipath" "rpath=rpath" ... -t cskip=0 fskip=-1
   ```

7. For Transfer F, code the following skip logic setting:

   ```
   dtscli "ipath=ipath" "rpath=rpath" ... -t cskip=0 fskip=-1
   ```

   If Transfers E and F succeed, the next transfer is activated, but if either transfer fails, the entire job aborts according to the original requirements for this example.

Note: No skip logic settings are needed for Transfer G, because it is the final transfer in the job.

dtscli Command—Apply Filters

The dtscli command can be used to send filtered transfers. DTS agents have the ability to apply filters when a file is being transferred. Filters transform data in some way, such as encryption and decryption, compression and decompression, and so on.

There are two basic categories of filters:

File Filters (f_filters)

Process the data being transferred one file at a time. File filters are applied to the entire file before it is transferred.

Parcel Filters (p_filters)

Process the data being transferred one parcel at a time. (A parcel is a smaller unit of data within the file.) Parcel filters are applied as the file is being transferred.

Filters usually work in pairs. The write filter must reverse an action performed by the read filter. Filters are applied in the following order:

1. File read filters
2. Parcel read filters
3. Parcel write filters
4. File write filters

Read filters of the same type (file or parcel) are applied in the same order that they are specified on the command line. Write filters of the same type are applied in the reverse order that they are specified on the command line.

Typically, you specify filter parameters for agent-to-agent transfers only, not for managed transfers. The only exception is any encryption filter. For example, Data Transport Service supplies the PLAIN file and parcel encryption filters, each of which requires the ENDEC_KEY parameter. For both managed transfers and agent-to-agent transfers, you must specify the ENDEC_KEY when you use the PLAIN encryption filters. For all other filters that require parameters, managed transfers automatically take the values supplied by the Transfer Object Server.

dtscli Command—Add Parcel Filters

Valid DTSCLI parcel filters include the following types:

- Binary (default) or text
- Compress
- Encrypt

The two main parcel filters used are either binary or text. Use the text filter if transferring text files between two different types of computers (Windows XP and UNIX, for example). This ensures that the text format is preserved; otherwise, always use binary filters.

Note: Binary and text filters are mutually exclusive: A transfer can either be text or binary but not both.

Help for adding filters to agent transfers is given by the command:

```
dtscli -help agent p_filters
```

Example: Transfer a File as Text

- The following example transfers the file from Gold to Silver using the text filter and a short format:

```
dtscli -agent "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    "p_filters=text"
```

- This example performs the same operation but uses a long format:

```
dtscli -agent "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    "p_filters=TEXT_READ:TEXT_WRITE"
```

Note: Only use text filters for transferring text files.

Example: Compress a Text File

- The following example transfers the file from Gold to Silver using the compression filter and a short format:

```
dtscli -agent "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    "p_filters=text compress"
```

- This example performs the same operation but uses a long format:

```
dtscli -agent "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    "p_filters=TEXT_READ:TEXT_WRITE HUFFMAN_COMPRESS:HUFFMAN_UNCOMPRESS"
```

Example: Add Multiple Parcel Filters

- The following example adds more than one pair of parcel filters using a short format:

```
dtscli -agent "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    "p_filters=binary compress”
```

- This example performs the same operation but uses a long format:

```
dtscli -agent "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    "p_filters=BINARY_READ:BINARY_WRITE
            HUFFMAN_COMPRESS:HUFFMAN_UNCOMPRESS"
```

The order in which the filters are listed is important. The binary filter *must* be listed before the compression filter. For example, try swapping the filters around; call the compression filter first, then the binary filter. You will notice that the transfer fails.

Note: Always make sure that either the binary or text filter is first in the list.

dtscli Command—Add File Filters

Valid DTSCLI file filters include the following types:

- Directory filters
- Encryption filters
- File attribute filters

When adding file filters, parcel filters may also be needed. To transfer a directory, file filters must now be added to the dtscli command. If you try to transfer a directory without adding the directory filter, you will get the following error message:

```
"Failed to open input data <c:\dtstemp> error=<Permission denied>".
```

Help for file filters is given by the command:

```
dtscli -help agent f_filters
```

Example: Transfer a Directory

- The following example transfers a directory using a short format for the directory filter:

```
dtscli -agent ipath=Gold::c:\dtstemp
    "rpath=Silver::c:\dtstemp"
    "f_filters=dir"
```

- This example performs the same operation but uses a long format:

```
dtscli -agent "ipath=Gold::c:\dtstemp"
    "rpath=Silver::c:\dtstemp"
    "f_filters=DIRTREE_READ:DIRTREE_WRITE"
    "p_filters=binary"
```

Note: If using the longhand format, you must also apply parcel filters.

Example: Encrypt a File

The following example encrypts the file before it sending it to the responder:

```
dtscli -agent "ipath=Gold::c:\dtstemp"
    "rpath=Silver::c:\dtstemp"
    "f_filters=PLAIN_ENCRYPT_FILE(ENDEC_KEY=key):PLAIN_DECRYPT_FILE()"
    "p_filters=binary"
```

Example: Apply Attributes of Original File to Transferred File

The use of the file attribute filter forces the transferred file to have the same attributes as the original.

- This example gives file.src certain attributes, like read only, hidden, and so on, using a short format:

```
dtscli -agent "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    "f_filters=fattr"
    "p_filters=binary'
```

- This example performs the same operation but uses a long format:

```
dtscli -agent "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    "f_filters=FILE_ATTRIBUTE:"
    "p_filters=binary"
```

Predefined Filters

The following is a list of predefined DTS file and parcel filters:

Filter Name	Description	Type
BINARY_READ	Binary read	read_parcel
BINARY_WRITE	Binary write	write_parcel
TEXT_READ	Text read	read_parcel
TEXT_WRITE	Text write	write_parcel
STAGING	Use initiator staging file	read_file
STAGING	Use responder staging file	write_file
HUFFMAN_COMPRESS	Huffman compression	read_parcel
RLE_COMPRESS	RLE compression	read_parcel
ZLIB_COMPRESS	ZLIB compression	read_parcel
LZRW_COMPRESS	LZRW compression	read_parcel
HUFFMAN_UNCOMPRESS	Huffman decompression	write_parcel
RLE_UNCOMPRESS	RLE decompression	write_parcel
ZLIB_UNCOMPRESS	ZLIB decompression	write_parcel
LZRW_UNCOMPRESS	LZRW decompression	write_parcel
PLAIN_ENCRYPT	DTS Plain encryption	read_parcel
PLAIN_DECRYPT	DTS Plain decryption	write_parcel
PLAIN_ENCRYPT_FILE	DTS Plain encryption file filter	read_file
PLAIN_DECRYPT_FILE	DTS Plain decryption file filter	write_file
VIRUS_SCAN	Identify viruses	read_file
VIRUS_SCAN	Identify viruses	write_file
CREATE_PATH	Creates a path based on the input or the filter parameters	read_file
CREATE_PATH	Creates a path based on the output or the filter parameters	write_file
DIRTREE_READ	Creates an archive containing the contents of a directory tree	read_file
DIRTREE_WRITE	Creates a directory tree based on the contents of an archive	write_file
USD_RESOLVE_PATH	Reserved for future use	write_file
FILE_ATTRIBUTE	File attribute filter	read_file

Filter Name	Description	Type
BINARY_VARREC_READ	MVS variable record read	read_parcel
BINARY_VARREC_WRITE	MVS variable record write	write_parcel
PDS_UNLOAD	MVS PDS unload	read_file
PDS_UNLOADED_READ	MVS PDS unloaded file read	read_parcel
PDS_UNLOADED_WRITE	MVS PDS unloaded file write	write_parcel
PDS_LOAD	MVS PDS load	write_file
VSAM_UNLOAD	MVS VSAM unload	read_file
VSAM_UNLOADED_READ	MVS VSAM unloaded file read	read_parcel
VSAM_UNLOADED_WRITE	MVS VSAM unloaded file write	write_parcel
VSAM_LOAD	MVS VSAM load	write_file
SAVE_FILE_READ	AS/400 save file read	read_file
SAVE_FILE_WRITE	AS/400 save file write	write_file
DM_SCRIPT_READ	Read-stage Desktop Management script execution	read_file
DM_SCRIPT_WRITE	Write-stage Desktop Management script execution	write_file
AES256_ENCRYPT	CA-DSM AES-256 encryption	read_parcel
AES256_DECRYPT	CA-DSM AES-256 decryption	write_parcel
AES256_ENCRYPT_FILE	CA-DSM AES-256 encryption file filter	read_file
AES256_DECRYPT_FILE	CA-DSM AES-256 decryption file filter	write_file
AES192_ENCRYPT	CA-DSM AES-192 encryption	read_parcel
AES192_DECRYPT	CA-DSM AES-192 decryption	write_parcel
AES192_ENCRYPT_FILE	CA-DSM AES-192 encryption file filter	read_file
AES192_DECRYPT_FILE	CA-DSM AES-192 decryption file filter	write_file
AES128_ENCRYPT	CA-DSM AES-128 encryption	read_parcel
AES128_DECRYPT	CA-DSM AES-128 decryption	write_parcel
AES128_ENCRYPT_FILE	CA-DSM AES-128 encryption file filter	read_file
AES128_DECRYPT_FILE	CA-DSM AES-128 decryption file filter	write_file
3DES_ENCRYPT	CA-DSM 3DES encryption	read_parcel
3DES_DECRYPT	CA-DSM 3DES decryption	write_parcel
3DES_ENCRYPT_FILE	CA-DSM 3DES encryption file filter	read_file
3DES_DECRYPT_FILE	CA-DSM 3DES decryption file filter	write_file
EXT_COMPRESS	External Compress	read_file

Filter Name	Description	Type
EXT_UNCOMPRESS	External Decompress	write_file

More information:

dtscli Command—Apply Filters (see page 73)

dtscli Command—Use Transfer Aliases

Keeping track of all your transfer IDs is not easy. We remember names a lot better than we do numbers. DTSCLI supports name aliases, which means that you can use the transfer or transfer job's label to refer to it rather than the number.

Example: Create and Activate a Job Using Aliases

This example uses labels for both the transfer and transfer job.

To create and activate a job using name aliases

1. Create a transfer using an alias and defer activation:

```
dtscli -transfer "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    iuser=dts::dts
    ruser=dts::dts
    label=WATER
    -mode defer
```

   The transfer is created. Notice the label.

2. Create a transfer job using an alias:

```
dtscli -job method=create label=FIRE
```

   The transfer job is created. Notice the label.

3. Add the transfer to the transfer job:

```
dtscli -job id=FIRE method=add method_parameters=WATER
```

4. Activate the transfer:

```
dtscli -job method=activate id=FIRE
```

Using the ADT Transfer Client, if installed, you can view the new transfer and transfer job that you have created and activated. However, you cannot view the labels.

dtscli Command—View Transfer Status

You can use the status method to retrieve and view the status of transfers.

Example: View the Status of a Single Transfer

This example displays the status of a transfer with the label “WATER”:

```
dtscli -transfer method=status id=WATER
```

Example: View the Status of All Transfers

This example displays the status of all transfers by using the special keyword "id=all":

```
dtscli -transfer method=status id=all
```

HTTP Protocol Parameters

Data Transport Service r2 and r3 support the HyperText Transfer Protocol (HTTP) protocol. The HTTP protocol is the most common protocol for transferring data from World Wide Web servers to browsers.

You can create, maintain, and perform transfers to receive data from an HTTP server or its proxy server, even if the server does not have a DTS agent installed. Data Transport Service thus supports downloading files from the Internet. Using Data Transport Service, you can receive data from HTTP servers or their proxy servers, but you cannot send data to them.

The syntax for performing an HTTP agent-to-agent transfer (see page 48) follows. All operands are required.

```
dtscli -a|t
    protocol=http "ipath=DTA_machine_name::destination_path"
    "rpath=http_host_name::http_source_file_path_name"
    iuser=username::password
    direction=r
```

-a|t

Indicates the type of transfer, agent-to-agent (-a) or managed (-t).

protocol=http

Specifies HTTP as the protocol for any Internet download.

"ipath=*DTA_machine_name::destination_path*"

Specifies the name of the DTS agent computer that is receiving the HTTP download, followed by two colons, followed by the path name of the newly downloaded file on this computer.

"rpath=*http_host_name::http_source_file_path_name*"

Specifies the host name of the HTTP server (or its proxy server) followed by two colons, followed by the path name of the file you want to download from this server.

iuser=*username::password*

Specifies the user name and password for the initiating computer, that is, the DTS agent computer that is initiating the transfer.

direction=r

Specifies that the initiating computer receives the transfer.

Chapter 4: Agent-to-Agent Transfers

The parameter -agent (or -a) specifies that the transfer should be an agent-to-agent transfer, that is, one that does not involve a Transfer Object Server (TOS).

The following examples illustrate basic agent-to-agent transfer operations.

Example: Perform a Simple Agent-to-Agent Transfer

The following example is a simple agent-to-agent transfer for two UNIX or Linux computers. Note the use of the required -agent (-a) parameter.

```
dtscli -a iuser=username::password ruser=username::password
    "ipath=LondonHQ::/home/mandates/memo5005.txt"
    "rpath=Local100::/home/mandates/memo5005.txt"
```

Example: Send One File from a Computer to Multiple Recipients

When using an agent-to-agent transfer, you can send *one* file from one computer to multiple recipients, as shown in the following example.

```
dtscli -a "ipath=jones201::c:\mandates\memo5005.doc"
    "rpath=smith110::c:\mandates\memo5005.doc"
    "rpath=patel335::c:\mandates\memo5005.doc"
```

This section contains the following topics:

dtscli Command—Create Simple Point-to-Point Transfers

The dtscli -agent (or -a) command can be used to create simple point-to-point transfers.

The following is the syntax for a simple point-to-point transfer:

```
dtscli -agent "ipath=InitiatorMachine::SourceFile"
    "rpath=Respondermachine::DestinationFile"
    iuser=InitiatorUser::Password
    ruser=ResponderUser::Password
```

Example: Transfer a File

The following example transfers the file, c:\dtstemp\file.src, from computer Gold to c:\dtstemp\file.dest on computer Silver:

```
dtscli -agent "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    iuser=dts::dts
    ruser=dts::dts
```

dtscli Command—Create Multiple Point-to-Point Transfers

The dtscli -agent (or -a) command can be used to create multiple point-to-point transfers. The same input file can be transferred to multiple output locations by using the rpath and transfer_mode parameters.

Valid values for the transfer_mode parameter are as follows:

p2p

Specifies a point-to-point transfer.

PPP

Specifies Point to Point Protocol (PPP) mode. Valid only for point-to-point transfers over standard telephone lines, ISDN, and other high-speed connections.

Note: DTS does not support the latest version of PPP communication protocol, PPPv6.

f (or fanout)

Specifies a fanout transfer.

bcast

Specifies an IPv4 broadcast point transfer.

mcast

Specifies an IPv4 or IPv6 multicast transfer.

Default: p2p

Example: Transfer in Point-to-Point Mode

This example performs two point-to-point transfers: first, the transfer between Gold and Silver, and after that has finished, the transfer between Gold and Bronze takes place.

```
dtscli -agent "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    "rpath=Bronze::c:\dtstemp\file.dest"
```

Example: Transfer in Fanout Mode

Fanout transfers are those in which data is sent to multiple responding computers using a direct point-to-point communications protocol. Data is read once and then sent multiple times, once to each of the target responders.

This example sets the transfer method to fanout mode for our earlier point-to-point transfer example:

```
dtscli -agent "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    "rpath=Bronze::c:\dtstemp\file.dest"
    transfer_mode=fanout
```

Example: Transfer as a Broadcast

This example sets the transfer method to broadcast (bcast). Broadcasts require a broadcast address.

```
dtscli -agent "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    "rpath=Bronze::c:\dtstemp\file.dest"
    transfer_mode=bcast
    broadcast_address=your_broadcast_address
```

Example: Transfer as a Multicast

This example sets the transfer method to multicast (mcast). Multicasts are similar to broadcasts in that you need a multicast address:

```
dtscli -agent "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    "rpath=Bronze::c:\dtstemp\file.dest"
    transfer_mode=bcast
    multicast_address=your_multicast_address
```

Note: For instructions for calculating broadcast and multicast addresses, see the *Data Transport Service Administration Guide*.

dtscli Command—Set Up Hop Transfers

The dtscli -agent (or -a) command can be used to create hop transfers. This may be necessary when there is no direct route from the initiator computer to the responder computer. A hop transfer lets the transfer take place using a third computer.

The following is the syntax for a hop transfer:

```
dtscli -agent "ipath=InitiatorMachine::SourceFile"
    "rpath=ResponderMachine::DestinationFile"
    iuser=InitiatorUser::InitiatorPassword
    ruser=ResponderUser::ResponderPassword
    hop_hosts=HopMachine
```

Example: Transfer a File Using a Hop Computer

This example performs a point-to-point transfer using computer Bronze. First the file is transferred from Gold to Bronze and then from Bronze to Silver.

```
dtscli -agent "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    iuser=dts::dts
    ruser=dts::dts
    hop_hosts=bronze
```

Example: Transfer a File Using Multiple Hop Computers

It is possible to hop using multiple computers. To do this, add a space-separated list of hop computers to the hop_hosts parameter. In this example, the transfer goes from Gold to Bronze, Bronze to Tin, and then Tin to Silver.

```
dtscli -agent "ipath=Gold::c:\dtstemp\file.src"
    "rpath=Silver::c:\dtstemp\file.dest"
    iuser=dts::dts
    ruser=dts::dts
    "hop_hosts=bronze tin"
```

Note: It is important to enclose the hop_hosts parameter with quotation marks.

dtscli Command—Send Multiple Output Files (Agent-to-Agent)

The dtscli -agent (or -a) command can be used to send multiple output files using an agent-to-agent transfer. You must enter the -agent operand once for each input and output file, separated by a space. You can specify a maximum of 32 output files for a single transfer.

Example: Send the Same Output File to Multiple Computers

If you are creating an agent-to-agent transfer, and you want to send the same output file to two or more computers, you can enter the input path once and the output path as many times as needed to specify the responders:

```
dtscli -a iuser=username::password ruser=username::password
    "ipath=steve::c:\misc\secrets.doc"
    "rpath=margaret::c:\misc\secrets.doc"
    "rpath=katie::c:\misc\secrets.doc"
    "rpath=roshni::c:\misc\secrets.doc"
```

If any computers in the transfer have their security mode set to "fail," take special care in specifying the required user name and password. Because you may specify only one user ID and password, you must make sure before sending the transfer that the user ID and password you specify is valid for every receiving computer. Alternatively, make sure that the security mode is quiet or warn for any receiving computer on which the user ID and password combination you specified is not valid. Otherwise, the security features of the DTS agent on the receiving computer cause the transfer to that computer to fail.

Example: Send Multiple Input Files

If you want to send multiple input files using agent-to-agent transfers, you must use multiple dtscli commands, as illustrated in the following example:

```
dtscli -a "ipath=steve::c:\misc\secrets1.doc"
    "rpath=margaret::c:\misc\secrets1.doc"
dtscli -a "ipath=steve::c:\misc\secrets2.doc"
    "rpath=margaret::c:\misc\secrets2.doc"
dtscli -a "ipath=steve::c:\misc\secrets3.doc"
    "rpath=margaret::c:\misc\secrets3.doc"
```

More information:

Agent-to-Agent Transfers (see page 9)

dtscli Command—Change Direction

The direction of a transfer is determined by the direction parameter. Valid values for this parameter are as follows:

s

Sends the data from the initiator computer to the responder computer.

r

Sends the data from the responder computer to the initiator computer.

Default: s

Example: Transfer a File from the Responder to the Initiator

This example directs the transfer to pull or transfer file.src from Silver (the responder) to file.dest on Gold (the initiator).

```
dtscli -agent "ipath=Gold::c:\dtstemp\file.dest"
    "rpath=Silver::c:\dtstemp\file.src"
    direction=r
```

TCP Protocol Parameters

Valid TCP parameters for the dtscli command in agent-to-agent transfers (see page 48) are as follows:

IPORT=*nnnn*

Specifies the number of the legacy port on which the DTS agent is listening for TCP connections.

Default: 8222

NETADDR=*nnn.nnn.nnn.nnn*

Indicates the IP address to use in place of the responder address (if required). An example of an IPv6 address would be fe80::20d:56ff:fe18:5843.

HTTP Protocol Parameters

Data Transport Service r2 and r3 support the HyperText Transfer Protocol (HTTP) protocol. The HTTP protocol is the most common protocol for transferring data from World Wide Web servers to browsers.

You can create, maintain, and perform transfers to receive data from an HTTP server or its proxy server, even if the server does not have a DTS agent installed. Data Transport Service thus supports downloading files from the Internet. Using Data Transport Service, you can receive data from HTTP servers or their proxy servers, but you cannot send data to them.

The syntax for performing an HTTP agent-to-agent transfer (see page 48) follows. All operands are required.

```
dtscli -a|t
    protocol=http "ipath=DTA_machine_name::destination_path"
    "rpath=http_host_name::http_source_file_path_name"
    iuser=username::password
    direction=r
```

-a|t

Indicates the type of transfer, agent-to-agent (-a) or managed (-t).

protocol=http

Specifies HTTP as the protocol for any Internet download.

"ipath=*DTA_machine_name::destination_path*"

Specifies the name of the DTS agent computer that is receiving the HTTP download, followed by two colons, followed by the path name of the newly downloaded file on this computer.

"rpath=*http_host_name::http_source_file_path_name*"

Specifies the host name of the HTTP server (or its proxy server) followed by two colons, followed by the path name of the file you want to download from this server.

iuser=*username::password*

Specifies the user name and password for the initiating computer, that is, the DTS agent computer that is initiating the transfer.

direction=r

Specifies that the initiating computer receives the transfer.

Point to Point Protocol (PPP) Parameters

PPP is a communication protocol for computers that use TCP/IP over standard telephone lines, ISDN, and other high-speed connections.

Note: DTS does not support the latest version of PPP communication protocol, PPPv6.

Valid PPP parameters for the dtscli command in agent-to-agent transfers (see page 48) are as follows:

PPP for Windows

PPP_ENTRY=*entry_name*

Defines the name of the pre-configured phone book entry to be used by the PPP protocol when establishing a connection to the responder.

PPP_PHONENUMBER=*phone_number*

Specifies the telephone number required to dial in to the responder.

PPP_USERNAME=*login_name*

Specifies the name of the user account to be used for authentication with the responder. Enter the name of the user account that the initiating computer will use to dial in to the responder. This account must be granted dial-in permission.

PPP_PASSWORD=*login_password*

Specifies the password of the user account to be used for authentication with the responder. Enter the password of the user account specified in PPP_USERNAME.

PPP for Solaris Solstice

PPP_ENTRY=*entry_name*

Specifies the name used to identify the responder in link.conf.

PPP_USERNAME=*login_name*

Specifies the name of the user account to be used for authentication with the dial-in account on the responder.

PPP_PASSWORD=*login_password*

Specifies the password of the user account to be used for authentication with the dial-in account on the responder.

PPP_AUTHENTICATION=*PAP|CHAP|CLEARTEXT*

Identifies the authentication protocol used to authenticate with the responder. Password Authentication Protocol (PAP) is the most likely choice to succeed.

Note: PAP is prevalent but is weaker than Challenge-Handshake Authentication Protocol (CHAP). Refer to the responder's configuration to determine what authentication protocol to use. If no PPP authentication is required, set the value to CLEARTEXT.

PPP_DYNAMICIP=*YES|NO*

Indicates whether the server assigns dynamic IP addresses to its clients. Set to NO if the server assigns a static IP address to the client. Then use PPP_ASSIGNEDIP to specify the required IP address. Refer to the responder's configuration to determine the correct value.

PPP_ASSIGNEDIP=*client_IP_address*

Specifies the IP address for the connection to the responder when PPP_DYNAMICIP is set to NO. Use this parameter when the remote server assigns a static IP address to the client.

PPP_PHONENUMBER=*phone_number*

Specifies the telephone number of the responder. This number is used by the initiating computer to gain access to the responder.

Example: Point-to-Point Windows Transfer

To send a point-to-point transfer from a Windows computer named benjamin to one named reuben using the PPP protocol, enter the following command:

```
dtscli -a "ipath=benjamin::c:\nhmh.c" -protocol=ppp
    "protocol_parameters=PPP_ENTRY=williamc
    PPP_PHONENUMBER=16095551212
    PPP_USERNAME=bill0627 PPP_PASSWORD=whisocks"
    "rpath=reuben::c:\nhmh.c"
```

Example: Point-to-Point Solaris Solstice Transfer

To send a point-to-point transfer from a Solaris Solstice computer named HQ501 to one named Branch212 using the PPP protocol, enter the following command:

```
dtscli -a "ipath=HQ501::c:\nhmh.c" protocol=ppp
    "protocol_parameters=PPP_ENTRY=williamc
    PPP_USERNAME=bill062> PPP_PASSWORD=whisocks
    PPP_DYNAMICIP=YES PPP_PHONENUMBER=19085551212"
    "rpath=Branch212::c:\nhmh.c"
```

In this example, note that the responder has a dynamic IP address, therefore, you do not need to specify the PPP_ASSIGNEDIP=*client_IP_address* parameter. However, that parameter is required if the responder has a static IP address.

UDP Protocol Parameters

Valid UDP parameters for the dtscli command in agent-to-agent transfers (see page 48) are as follows:

IPORT=*nnnn*

Specifies the number of the port on which the DTS agent is listening for UDP connections.

Default: 8223

NETADDR=*nnn.nnn.nnn.nnn*

Specifies the IP address to use in place of the responder address (if required).

Example: Point-to-Point Transfer Using UDP

To send a point-to-point transfer from one Windows computer to another, using the UDP protocol and a different port number than the default one, enter the following command. Specify IPORT if the default listening port number of the responding DTS agent has been changed because of a conflict with another application that is already using it.

```
dtscli -a "ipath=ruth::c:\ruth\memo2.doc"
    protocol=udp "protocol_parameters=IPORT=3535"
    "rpath=naomi::c:\naomi\memo2.doc" ...
```

CPI-C Protocol Parameters

If a computer can connect to other computers only by using the CPI-C protocol, then it can function only as a responder in a data transfer using the dtscli command. However, users of Advantage Data Transport or another CA solution designed to enhance the functions of Data Transport Service can use such computers as both initiators and responders in data transfers by using the Data Transport Explorer GUI rather than the dtscli command to perform the transfer.

Valid CPIC parameters for the dtscli command in agent-to-agent transfers (see page 48) are as follows:

CPIC_LUNAME=*luname*

(Required) The logical unit (LU) name of the responder computer. The first character of an LU name must be alphabetic.

Example: Point-to-Point Transfer

To send a point-to-point transfer from a Windows computer named HQ501 to one named Branch212 using the CPI-C protocol, enter the following command:

```
dtscli -a "ipath=HQ501::c:\nhmh.c" protocol=cpic
    "protocol_parameters=CPIC_LUNAME=D1234567"
    "rpath=Branch212::c:\nhmh.c"
```

Send Multiple Output Files Using CPI-C

You can use CPI-C to send multiple output files if you follow these *special requirements* for each output file:

To send multiple output files

1. In the "rpath=*output_file*" operand, specify the CPIC_LUNAME of the responder instead of the computer name of the responder.

   No brackets are required around the luname.

2. In the protocol parameters operand, do not specify the CPIC_LUNAME.

Example: Agent-to-Agent Transfer

The following is an agent-to-agent transfer example for CPI-C users. It is valid for a Windows user to transfer a file across multiple platforms to four different computers.

```
dtscli -a "ipath=SWTZDAMY::DTO.DTS20.C(PWTCP)"
    "rpath=A44ISTD1::PUBLIC.AAKEHH.PWTCP.TSO94"
    "rpath=A04ISTD1::PUBLIC.AAKEHH.PWTCP.TSO84"
    "rpath=KWWIGP21::C:\TDSTEST\SNATEST\FANOUT.C"
    "rpath=KWWIGP51::/USR/AAKEHH/TDSTEST/SNATEST/FANOUT.C"
    direction=s protocol=CPIC
```

Example: Managed Transfer

An equivalent managed transfer for CPI-C users follows. This example is valid for a Windows user to transfer a file across multiple platforms to four different computers. Note that the -t operand and its parameters (protocol= and retry_interval=) must be specified in their entirety for each transfer.

```
dtscli -t "ipath=SWTZDAMY::DTO.DTS20.C(PWTCP)"
    "rpath=A44ISTD1::PUBLIC.AAKEHH.PWTCP.TSO94"
    protocol=CPIC retry_interval=30
    -t "ipath=SWTZDAMY::DTO.DTS20.C(PWTCP)"
    "rpath=A04ISTD1::PUBLIC.AAKEHH.PWTCP.TSO84"
    protocol=CPIC retry_interval=30
    -t "ipath=SWTZDAMY::DTO.DTS20.C(PWTCP)"
    "rpath=KWWIGP21::C:\TDSTEST\SNATEST\FANOUT.C"
    protocol=CPIC retry_interval=30
    -t "ipath=SWTZDAMY::DTO.DTS20.C(PWTCP)"
    "rpath=KWWIGP51::/USR/AAKEHH/TDSTEST/SNATEST/FANOUT.C"
    protocol=CPIC retry_interval=30
```

SPX Protocol Parameters

Valid SPX parameters for the dtscli command in agent-to-agent transfers (see page 48) are as follows:

SPX_IPORT=*nnn*

Specifies the number of the port on which the DTS agent is listening for SPX connections.

Default: 8226

NETADDR=*nnnnnnnn*

Specifies the network number portion of the IPX address to use in place of the responder address (if required).

NODEADDR=*nnnnnnnnnnnn*

Specifies the node number portion of the IPX address to use in place of the responder address (if required).

IPX addresses are formatted as follows: *network-number.node-number*, which is in 8.12 format; for example, 789EE001.00805F0EC8D8.

If the initiating DTS agent cannot recognize the responder address as being in 8.12 format address, it searches the protocol parameters for NETADDR and NODEADDR. If the responder address is in the 8.12 format, then there is no need to specify the SPX protocol parameters. However, in certain cases, the responder address may not be in IPX format; therefore, the SPX parameters are provided.

When using the SPX address in the protocol parameters, specify the address in 8.12 format, as follows:

"protocol_parameters=NETADDR=*network-number* NODEADDR=*node-number*"

For example:

"protocol_parameters=NETADDR=89EE001 NODEADDR=00805F0EC8D8"

Example: Point-to-Point Transfer

To send a point-to-point transfer from a computer named toro444 to one named toky0001 using the SPX protocol with its SPX NETADDR= and NODEADDR= parameters, enter the following command:

```
dtscli -a "ipath=toro444::c:\jpegs\cows\bessie.jpeg" protocol=spx
    "protocol_parameters=NETADDR=789EE001 NODEADDR=00805F0EC8D8"
    "rpath=toky0001::c:\jpegs\cows\bessie.jpeg"
```

Index