Teleport Exec™
Teleport Exec/VLX™

Interface and Specifications

TENNYSON MAXWELL INFORMATION SYSTEMS, INC.
www.tenmax.com

Teleport Exec™

Interface and Specifications

Teleport Exec is a multithreaded Win32 executable program that performs all the file-retrieving, Internet-exploring, HTML parsing and relinking functions of the Teleport Pro™ commercial webspider—except that Teleport Exec is run and configured entirely from the command-line, and optionally by response files. Teleport Exec provides a complete programmatic interface to all of the functions and capabilities of Teleport Pro, as well as extra reporting functions not available through Teleport Pro. Communication with the calling process is handled via the command-line, response files, system-wide progress messages broadcast via Windows™, and end-of-session report and log files generated by Teleport Exec.

Teleport Exec has a database capacity of 260,000 URLs per project. Teleport Exec/VLX™ (Very Large eXploration) is a modified version of Teleport Exec that is capable of handling up to 40 million addresses in a single project database. The API for Teleport Exec/VLX is identical to that of Teleport Exec; the only difference between the programs is the larger database capacity of the VLX spider engine. For further information regarding memory requirements and usage of Teleport Exec/VLX, please consult the readme document accompanying that program.

Contents   **Page**

Terms and Lexical Conventions Used in this Document

Project is the combination of starting addresses, and exploration, retrieval, and relinking parameters that define where and how Teleport Exec will retrieve, modify, and save files from the Internet.

Session indicates a complete execution, by Teleport Exec, of a given project.

Arguments are strings that appear on the command-line.

Switches are arguments that control project creation, modification, running, relinking, and other Teleport Exec functions. Switches must be preceded by either the / (forward slash) or - (minus) characters.

Parameters are strings or numbers that follow a switch.

`monotype` denotes command-line arguments, switches, or parameters.

`[square brackets]` denote optional arguments or parameters.

`<angle brackets>` denote required arguments or parameters.

`|` denotes alternative parameters; for example, `/r[r|u|c]` indicates that the `/r` switch may be followed by any one of the letters r, u, or c. In some cases, more than one of the listed parameters may be given in a single switch.

Command-line Arguments and Response Files

When Teleport Exec is executed from the command-line or batch file, or from another process, command-line arguments are used to configure the program's operation, its exploration parameters, and the addresses it will search. In effect, the command-line for Teleport Exec generates a complete Teleport Pro project—or opens an existing one and, optionally, modifies it—and then runs it.

To circumvent the command-line length limit under Windows (1024 characters), Teleport Exec allows the use of a "response file," a text file that contains, on one or more lines, command-line arguments. The arguments in the response file are processed in addition to, not in lieu of, other arguments on the command-line.

The Teleport Exec command-line format is—

```
exec <projectfile> [switches] [@responsefile]
```

The Teleport Exec/VLX command-line format is the same, but use `execvlx` instead of `exec`—

```
execvlx <projectfile> [switches] [@responsefile]
```

The arguments may appear in any order, except that the `<projectfile>` argument must precede all arguments that would affect the project. The command-line and response file together may contain any number of switches and response files. Multiple response files are allowed, each processed in the order listed. Response files may be nested; a response file may call another response file.

The complete command-line (after all response files have been processed) may contain only one project filename. If the project already exists, Teleport Exec will open it and modify and/or run it as the command-line dictates. If the project does not exist, Teleport Exec will save the project it creates under the given project filename.

Teleport Exec processes command-line arguments from left to right, and response file arguments from top to bottom. Successive arguments that conflict with preceding ones will override the preceding arguments. Project creation and modification arguments that precede the project filename will be ignored.

Arguments must be separated by one or more spaces. Arguments that contain space characters must be enclosed in double quotes. Note that in such a case the *complete* argument must be contained in quotes. For example, to specify autoconnection using the defined connection "Concentric Boston," the complete switch is—

```
"/aConcentric Boston"
```

a. The Response File Format

When Teleport Exec encounters a `@responsefile` argument (an argument preceded by the `@` symbol), it reads that file from disk and processes the arguments listed in it, in top-to-bottom order, as if the file were one long command-line. The response file arguments are processed in addition to, not in lieu of, other command-line and response file arguments.

Multiple response files are allowed, each processed in the order listed. Response files may be nested; a response file may call another response file.

Response files must be ASCII text files. They may be of any length, up to the operating system filesize limit of 2 Gb.

Each argument in the response file must begin and end on the same line. All arguments are treated as if quoted, and thus quotes are unnecessary. Thus, the switch for autoconnecting via the connection named "Concentric Boston" may be given on a single line in a response file, without quotation marks, as—

```
/aConcentric Boston
```

Response files may have comments. Comments are lines preceded by a semicolon (`;`), for example—

```
;This causes the program to dial Concentric Boston
/aConcentric Boston
;other lines here...
```

b.   Switches for Adding Addresses and Setting Address Properties

`/nu<address>` *New URL Address:* Adds a new starting address to the project. If the given address is already a starting address in the project, this switch references that address for purposes of the exploration parameter switches which follow it (see below). If the address already exists in the project, but is not a starting address, Teleport Exec promotes the address to starting address status.

New starting addresses are given the following default exploration parameters, which subsequent switches on the command-line may modify: Inner exploration depth=0, Outer exploration depth=0, boundary condition=1, and no user account or password.

The address must be specified as either a complete URL, partial URL, or local file path. Partial URLs will be considered HTTP addresses and parsed accordingly. For example—

`/nuhttp://www.tenmax.com`	adds http://www.tenmax.com/
`/nuwww.tenmax.com`	adds http://www.tenmax.com/
`/nuwww.tenmax.com/index.html`	adds http://www.tenmax.com/index.html
`/nuc:\temp\index.htm`	adds c:\temp\index.htm
`/nuftp://tenmax.com/test.doc`	adds ftp://tenmax.com/test.doc

Addresses may contain standard URL port specifications, for example—

`/nuberne.ch:100/home.htm`	adds http://berne.ch/home.htm on port 100

Addresses may contain hexadecimal escape codes, ISO-8859-1 entities, and SGML entities, for example—

`/nutest.com/Test%20file.htm`	adds http://test.com/Test file.htm
`/nutest.com/My file.htm`	adds http://test.com/My file.htm
`/nutest.com/Rålf.htm`	adds http://test.com/Rålf.htm

Addresses may contain any ANSI character, for example—

`/nutest.com/Rålf.htm`	adds http://test.com/Rålf.htm

Note that valid Internet addresses may contains spaces and quotation marks. To avoid confusing the command-line, you should specify such addresses using any of the special notations given above. For example, to add the address http://search.com/search?param="bin", *specify the following—*

```
/nusearch.com/search?param=%22bin%22
```

`/nid<depth>` *New Inner Depth:* Modifies the immediately preceding address's inner exploration link depth. This option also sets the default inner depth for any addresses added later with the `/nu` parameter.

`/nod<depth>` *New Outer Depth:* Modifies the immediately preceding address's outer exploration link depth. This option also sets the default outer depth for any addresses added later with the `/nu` parameter.

`/nb<1-9>` *New Boundary Condition:* Modifies the immediately preceding address's boundary condition. This option also sets the default boundary condition for any addresses added later with the `/nu` parameter. Note that the Inner Depth property is *always* obeyed; if you want "unlimited" inner depth scanning of a site, set the inner depth to a very high number, such as 1000.

Boundary conditions correspond to those listed in the Teleport Pro Starting Address dialog box, but Teleport Exec has some additional boundary types not in Teleport Pro. The boundary conditions are—

1   *Stay within path:* Directs Teleport Exec to explore only addresses that have the same domain and path stub as the starting address.

2   *Stay within domain:* Directs Teleport Exec to explore only addresses that have the same domain as the starting address.

3   *Stay within domain, or within path of external links:* Directs Teleport Exec to explore only addresses that have the same domain as the starting address, except that it may also explore addresses that have the same domain and path stub as any external link on the starting domain.

4   *Stay within domain, or within the domain of external links (and outer depth):* Directs Teleport Exec to explore only address that have the same domain as the starting address, except that it may also explore addresses that have the same domain as any external link on the starting domain (but it must also obey the outer depth setting).

5   *No boundaries, but stay within the "inner depth" from the starting address.* Directs Teleport Exec to explore all addresses leading away from the starting address, up to *n* addresses away from the starting address, where *n* is given by the previous `/nid` setting. When this boundary is selected, Teleport Exec does not distinguish between in-domain and out-of-domain addresses; any address may be explored, so long as it is within *n* links from the starting address.

6   *Stay within unqualified domain:* Directs Teleport Exec to explore only addresses that have the same unqualified domain as that of the starting address. For example, starting from the address "www.yahoo.com," addresses on the "info.yahoo.com" would be considered in-bounds. The "unqualified domain" is the **last two** elements of the full domain name. For example, "home.mysite.com," "shop.mysite.com," and "info.france.mysite.com" all have the same unqualified domain name.

7   *Stay within unqualified domain, or within path of external links:* Directs Teleport Exec to explore only addresses that have the same unqualified domain as that of the starting address, except that it may also explore addresses that have the same domain and path stub as any external link on the starting domain.

8   *Stay within unqualified domain, or within the domain of external links (and outer depth):* Directs Teleport Exec to explore only address that have the same unqualified domain as that of the starting address, except that it may also explore addresses that have the same domain as any external link on the starting domain (but it must also obey the outer depth setting).

9   *Null boundaries, except for inclusions:* With this setting, no URL is in-bounds unless it is a starting address or it matches at least one of the Inclusions set for the preceding starting address.

Note *that none of the above boundary conditions apply to embedded files, such as graphics, sounds, and applets, that are required to compose a retrieved web page. These files are always retrieved, so long as they meet all other retrieval specifications.*

`/na<account>` *New Account:* Modifies the immediately preceding address's user account. This option also sets the default account name for any addresses added later with the `/nu` parameter.

`/np<password>` *New Password:* Modifies the immediately preceding address's password. This option also sets the default password for any addresses added later with the `/nu` parameter.

`/ne[url]` *Enable Address:* Enables the URL, if specified; if no URL is specified as part of the command, then it enables any starting address specified most previously with the /nu command.

`/nd` *Disable Address:* Disables the URL, if specified; if no URL is specified as part of the command, then it enables any starting address specified most previously with the /nu command. If the URL is specified that does not yet exist in the project, the URL will be added to the project as a new starting address and disabled, in order to give effect to the command.

`/nt` *Set Title:* Specifies the title of the immediately preceding address. Please note that the title is cosmetic only, and will be visible only in the GUI versions of Teleport, or in a response file generated with the /m command.

`/nc` *Set Inclusions:* Sets the list of regular expressions that define additional "included" areas for the immediately preceding address. Inclusions are set using grep-style regular expressions. Multiple inclusions can be specified, if they are separated with semicolons. For example—

`/nc//*.server.com/`	includes all URLs ending in .server.com
`/nc.gif;.jpg`	includes all URLs to gif and jpg files
`/ncDocId=[5-9][0-9]+`	includes all URLs having DocId >= 50

`/ns` *Set Aliases:* Sets the list of aliases by which the server for the starting address may be known. Aliases are better than inclusions where the webserver actually responds to two or more names, but serves identical content, because URLs to files using the aliased server names are de-duplicated and removed. Aliases can also be specified using grep-style regular expressions. For example, if the starting address is "www.server.com", then—

`/nsserver.com`	says that "server.com" is another name for www.server.com
`/nswww2.server.com;www3.server.com`	says that "www2.server.com" and "www3.server.com" are other names for www.server.com
`/nswww[1-9].server.com`	says that "www1.server.com" through "www9.server.com" are other names for www.server.com

c. Switches for Setting Project Properties

`/pra[autosaverate]` *Autosave Rate:* Specifies the project autosave rate in seconds. Default: 600 seconds. If zero or no value is given, autosaving is disabled. When autosaving is enabled, Teleport Exec will always save the project file just prior to terminating.

`/prf[maximumsize]` *Retrieve All Files [less than maximum size]:* Disables the project's file retrieval specifications, if any, and directs Teleport Exec to retrieve all files it encounters. You may optionally set a maximum filesize limit, given in kilobytes. Default: On (with no maximum size restriction).

`/prs[patternlist[;minimumsize[;maximumsize][;title]]]` *New Retrieval Specification:* Adds a new file retrieval specification to the project. The specification must be given as a semicolon-separated list of filename patterns, optionally using DOS wildcards such as * (match zero or more characters) and ? (match one character). You may optionally specify minimum and maximum filesize limits for the specification, each limit given in kilobytes. You can also optionally specify a short descriptive title for the type. The descriptive title is purely cosmetic, and will appear only in the GUI or in response files generated with the /m command.

For example—

`/prs*.gif;*.jpg;*.jpeg`	specifies all gif, jpg, and jpeg files, of any size
`/prs*.gif;*.jpg;16;32`	specifies all gif and jpg files 16 to 32 kb
`/prsindex*.*;8`	specifies all files named index*.* 8 kb or larger
`/prspdf*.*;0;0;Acrobat`	specifies PDF files of any size, called "Acrobat"

Filename patterns may also contain **grep-style regular expressions,** except that the regular expression characters * . ? are given DOS-style meanings instead of their usual grep interpretations. For example—

`/prsfile[0-9].htm`	specifies files named f0.htm, f1.htm, f2.htm, …

If no pattern follows this switch, all existing file retrieval specifications are cleared. Default: No retrieval specifications.

`/prm[b|e|j|n]` *Set Retrieval Mode(s):* Sets or clears one or more retrieval mode types. These modes correspond to the last four options on Teleport Pro's Project Properties, Retrieval page. Unspecified modes are cleared. They are—

b   *Retrieve background graphics:* Unless this mode is set, Teleport Exec will not retrieve background graphics files, even if they otherwise meet all other retrieval specifications. Default: On.

e   *Retrieve embedded files (graphics, sounds, or animations):* Unless this mode is set, Teleport Exec will not retrieve embedded files (those files that appear along with the text of a web page), even if they otherwise meet all other retrieval specifications. Default: On.

j   *Retrieve Java applets:* Unless this mode is set, Teleport Exec will not retrieve Java applets, even if they otherwise meet all other retrieval specifications. This option also enables Teleport Exec's Java parser, which will scan retrieved applets and attempt to locate and retrieve any base classes the applet may require. Default: On.

n   *Retrieve only file names:* This option simply prevents Teleport Exec from saving retrieved files. Default: Off.

`/prb[a|d|m|h|i|l|s|e|f]` *Set Browsing Parameter(s):* Sets or clears one or more browsing, mirroring, and linking parameters . These modes correspond to the options on Teleport Pro's Project Properties, Browsing/Mirroring page. Unspecified parameters are cleared. They are—

a   *Always save HTML:* This option ensures that Teleport Exec always saves HTML files, regardless of the file's actual extension. Default: On.

d   *Preserve directory structure of remote server(s):* Directs Teleport Exec to save all files in a directory structure that mimics that of the remote server(s) the files are drawn from. Default: Off.

m   *Use browser-compatible filenames:* Directs Teleport Exec to append a ".htm" extension to HTML files that do not already have that extension. This ensures that the files will be treated as HTML files when opened offline in the user's browser. Default: On.

h *Add HTML and charset tags if necessary:* Directs Teleport Exec to add the <html> tag to HTML files that lack it; this is necessary for some older browsers to display content correctly offline. In addition, Teleport Exec will add a meta tag containing the charset specification for the page, if the page uses a non-default character set but lacks a charset specification. This is necessary for the page to display correctly, on all browsers. Default: On.

i *Add meta tags for the original URL and retrieved date/time:* Directs Teleport Exec to add meta tags to the document header, containing the page's original URL and the date/time it was retrieved. If this option is enabled, the option to add HTML and charset tags to the document is also automatically enabled. Default: Off.

l *Localize HTML links:* Enables Teleport Exec's dynamic and end-of-session relinking systems, which will relink all HTML files so that the links point to other retrieved files within the same project. Teleport Exec will automatically relink any files that become eligible for relinking during the retrieval session. At the session's end, Teleport Exec will then relink all remaining HTML files, as necessary. Default: On.

Teleport Exec's dynamic relinker is multithreaded and operates on a queued, as-available basis, giving execution priority to the retrieval threads. The end-of-session relinker is single-threaded, and simply sequences through all of the remaining HTML files, after the retrieval system has shut down.

HTML link localization **must** be enabled at the start of a project in order for Teleport Exec's relinking system ever to work. An HTML file not processed by the relinker before it is saved to disk, cannot be relinked later.

s *Localize links using short filename notation:* The same as option l, but relinks HTML files using the operating system's generated short name for the retrieved files.

This option does **not** cause Teleport Exec to change the names of the files it writes. Instead, it only relinks the HTML files using the generated (8.3) short filename. This ensures that the relinked website will work equally well on both long filename and short filename volumes—and that on long filename volumes, the original filenames are kept intact.

To export a website copy generated using this option to a short filename volume, teleport the file first to a long filename volume, then copy the files to the short filename volume using the generated short filename. (You may use the DOS xcopy /n command for this purpose.) Default: Off.

e *Accept HTML error pages as actual content:* This option tells Teleport to accept HTML error pages — that is, the content of pages sent with error codes 400 through 599 — as actual content to be read and stored. By default, Teleport ignores the content of error pages, as it rarely contains valid or useful information, but in some rare cases it may be useful to keep this content in the project. Default: Off.

f *Set file date to match server-reported file date*: This option tells Teleport to set the local file modification date to match the file modification date reported by the server (where available). Not all servers report file modification dates, and many report them only for some files (usually actual files, and not for dynamic web pages and other generated responses). Default: Off.

/prl<1|2|3|4|5> *Set Linkage Mode:* Sets the type of links that Teleport Exec will write for links to files it does not retrieve. Linkage modes only apply when Teleport Exec is commanded to localize links (using the /prbl switch). The first three modes correspond to the three modes listed on Teleport Pro's Project Properties, Browsing/Mirroring page, and apply only to *clickable* links which would point to files that are not retrieved in the project. The fourth mode applies to all unretrieved links in the project, whether clickable or embedded. The fifth mode applies to all links. The linkage modes are—

1   *Boundary link:* Causes Teleport Exec to point these links to a "boundary" location, which is typically a javascript pop-up message box that briefly describes why the file was not retrieved, and lets the user select whether to get the file from the original server. Other types of boundary links can be specified using the /b option. This is the default linkage mode. ***Note** that this is a change from previous versions of Teleport Exec, in which this linkage mode created a link to an HTML page containing the explanatory message.*

2   *Link to Internet:* Causes Teleport Exec to "externalize" these links so that they point to the absolute (usually Internet) location at which the file was originally addressed.

3   *Link to predicted location:* Causes Teleport Exec to write the link to the location where it would store the file, if the file were retrieved. This mode imposes the least burden on the relinking system. Links to missing files will not work—they may generate a short message box in the browser when clicked, or they will do nothing. However, if the missing files are eventually retrieved, relinking of the originating HTML page is not required.

4   *Link to Internet for all unretrieved links:* Causes Teleport Exec to rewrite *all unretrieved* links as external (absolute Internet) addresses. (Note that this option differs from linkage mode 2 in that mode 2 will cause Teleport Exec to relink only *clickable* unretrieved links.)

5   *Link to Internet for all links:* Causes Teleport Exec to rewrite *all* links, whether retrieved or unretrieved, as external (absolute Internet) addresses.

***Note** that Teleport Exec will always relink un-explorable elements, such as forms containing data that must be entered by the user, using mode 2 (link to Internet), so that such elements will still operate when browsed offline. Note also that Teleport Exec will, except when using linkage modes 4 or 5, relink unretrieved embedded elements to a dead local address.*

`/b[address|filename]` *Set Boundary Link Type:* Sets the type of link that Teleport Exec will write to replace links to files it does not retrieve, when the Linkage Mode (see above) is set to 1. Linkage modes only apply when Teleport Exec is commanded to localize links (using the `/prbl` switch). The types of "unretrieved" links Teleport can write are—

`unspecified` If no parameter is given after the `/b` command, then Teleport Exec will point unretrieved links to a javascript pop-up message box that briefly describes why the file was not retrieved, and lets the user select whether to get the file from the original server. This is the default option.

`address`   If a file address is given (any parameter that contains a slash (/) or backslash (\) character is considered to be a file address), then Teleport Exec will write unretrieved links so that they point to that address. Typically this should be an absolute address, such as "http://myserver.com/boundary.htm" or a similar absolute location. You should then create a file at that location, so that clicking on the link will retrieve that file.

`filename`   If a file name is given (it cannot contain any slashes or backslashes), then Teleport Exec will write unretrieved links so that they point to a file having that name, in the root of the project folder. You should then create a file having that name in the root of the project folder, so that clicking on the link will retrieve that file. ***Note** that if you clear the project, your boundary page will also be deleted, so you must re-create the boundary page and any other files it needs in the project folder, if you delete or otherwise clear the project folder.*

`/pre[f|o|j|c|s[gridsize]]` *Set Advanced Exploration Parameter(s):* Sets or clears one or more advanced exploration parameters. The exploration parameters correspond to the first three options listed on Teleport Pro's Project Properties, Exploration page. Unspecified parameters are cleared. They are—

`f`   *Explore frames:* Unless this mode is set, Teleport Exec will not retrieve frame source files, even if they otherwise meet all other retrieval specifications. Default: On.

o   *Explore forms:* Causes Teleport Exec to explore form links, when possible. Explorable forms are those that require no data or user input, or have only hidden data fields. Default: On.

j   *Process script and event code:* When this option is enabled, Teleport Exec will parse javascript commands contained in embedded scripts and events, in an attempt to find commonly used commands such as window.open(*url*) and location.href="*url*" which can create new links. Only where *url* is a single, identifiable URL requiring no interpretation (*i.e.*, it is a single string and is not, for example, a concatenation of strings), will Teleport Exec treat the command like a link. Default: On.

c   *Accept and return cookies:* Tells Teleport Exec that it is permitted to accept and return "cookies," which are small data tags often given by the server to a client in order to track the client's progress through the site, or for authentication purposes. If this option is not enabled, Teleport will not remember cookies sent to it and will not return those cookies to the server. Not returning cookies can sometimes prevent Teleport (or any other client) from getting data from a server. Default: On.

s   *Explore server-side image map [with optional grid size]:* Enables Teleport Exec's server-side image map pinging system. Only where necessary, Teleport Exec will "ping" a server-side image map across a grid of points, to locate any links. Upon relinking (if enabled; see above), Teleport Exec will install a parallel client-side image map in the calling HTML file, so that the image map may be browsed offline. Default: Off.

Teleport Exec will ping a server-side map only when necessary—if a client-side map is available, it will use the client-side map instead.

If no grid size is specified, the grid size defaults to 30 pixels.

/prt<threadcount> *Set Thread Count:* Sets the maximum number of retrieval threads Teleport Exec may launch. Range = 1–10. Default: 10, or the maximum number of open sockets that the operating system will allow, whichever is smaller.

/prz<timeout> *Set Thread Timeout:* Sets the thread timeout period, in seconds. Threads that wait longer than this period for a response from the remote server will abort. Note that the operating system's underlying sockets layer may impose a shorter timeout period; this value cannot be predicted. Default: 360 seconds.

/prrd<retrytimes> *Set Retries for Denied Requests:* Sets the number of times Teleport Exec will request a file, if the server reports that the file is temporarily unavailable. Teleport Exec will add retry requests as low-priority items to the normal retrieval queue, to avoid badgering an unresponsive server. Nevertheless, values in excess of 10 retries may be considered bad netiquette. Default: 5.

/prri<retrytimes> *Set Retries for Incomplete Files:* Sets the number of times Teleport Exec will attempt to retrieve a file, if the file is incompletely sent or corrupted during the transfer (when such a state is detectable). Teleport Exec will add retry requests as low-priority items to the normal retrieval queue, to avoid badgering the server, which may simply be overloaded. Nevertheless, values in excess of 10 retries may be considered bad netiquette. Default: 5.

/pru[g|b|h|e|s|a|x] *Set Update Parameters(s):* Sets or clears one or more of the project's update parameters. These parameters correspond to those listed in the bottom section of Teleport Pro's Project Properties, Exploration page. Unspecified parameters are cleared. They are—

g   *Update good files:* Unless this mode is set, Teleport Exec will not update retrieved files, even if they otherwise meet all other retrieval and update specifications. Default: On.

b   *Update bad files:* Unless this mode is set, Teleport Exec will not attempt to retrieve unretrieved files for which the remote server reported an error (such as file not found), even if they otherwise meet all other retrieval specifications. Default: On.

h   *Update HTML files:* Unless this mode is set, Teleport Exec will not update HTML files, even if they otherwise meet all other retrieval and update specifications. Default: On.

e   *Update embedded files:* Unless this mode is set, Teleport Exec will not update embedded files—background graphics, sounds, regular graphics, animations, and Java applets—even if they otherwise meet all other retrieval and update specifications. Default: On.

Note that the term "embedded" is here more broadly used than in the Retrieve Embedded Files mode.

s   *Update server-side image maps:* Unless this mode is set, Teleport Exec will not update (re-ping) server side image maps. Default: Off.

a   *Update all other files:* Unless this mode is set, Teleport Exec will not update non-HTML, non-embedded files, even if they otherwise meet all other retrieval and update specifications. Default: On.

x   *Update as explored:* Unless this mode is set, Teleport Exec will update all files in its database, regardless of whether they meet other project retrieval specifications (such as project boundaries, exclusions, etc.), so long as the meet the other update requirements in this section. With this mode set, Teleport will update only those files that meet project retrieval specifications *and* meet the other update requirements in this section. Therefore, if you use this option, please remember to set at least one of the other updating options as well (such as update HTML), . Default: Off.

`/prk[keywordlist]` *Set Keyword Exclusions:* Sets the project's keyword list. The keyword list may contain one or more keywords or phrases, separated by semicolons. If this switch is followed by no keywords, any existing project keywords are cleared. Default: No keywords.

When a project has keyword exclusions, retrieved HTML pages will not be saved unless they contain one or more keywords; and embedded files that belong to non-matching pages will be skipped altogether. Teleport Exec's keyword search is word-bounded and case insensitive.

`/prq[c|w]` *Set Keyword Properties:* Sets or clears one or more of the project's keyword matching options. Unspecified parameters are cleared. The options are—

c   *Match case:* Unless set, Teleport will match keywords regardless of case. Default: Off.

w   *Match whole words:* If set, Teleport will match only whole words. Default: On.

`/prx[patternlist]` *Set Filename Exclusions:* Sets the project's filename exclusion list. The list may contain one or more filename patterns, of the style normally used for file retrieval specifications (described above). This feature, like file retrieval specifications, supports both DOS- and grep-style wildcards and pattern-matching symbols. If no pattern list follows this switch, any existing pattern list is cleared. Default: No filename exclusions.

`/prp[pathstub]` *Set Path Exclusion:* Adds a path stub to the project's path exclusions list. Teleport Exec will not explore or retrieve any files whose addresses **begin with** the given path stub. If no path stub follows this switch, all existing path exclusions are cleared. Default: No path exclusions.

Path stubs must be specified using **complete URL notation**—in other words, the http://, ftp://, and c:\ prefixes are required. For example—

`http://www.microsoft.com/`   will exclude all addresses on www.microsoft.com

`http://www.test.com/home/`   will exclude all addresses in /home/ on www.test.com

Teleport Exec supports **DOS- and grep-style wildcards and pattern-matching symbols** for this feature. For example—

`http://*.ca/`   will exclude all addresses on domains ending in .ca

`http://*/test/`	will exclude all addresses with /test/ in the path
`http://*.c[ah]/`	will exclude all addresses on domains ending in .ca or .ch

`/prn[d|s|r|f|w[delaytime]]` *Set Netiquette Parameters:* Sets or clears one or more of the project's netiquette parameters. These parameters correspond to those listed at the top of Teleport Pro's Project Properties, Netiquette page. Unspecified options are cleared. They are—

- d   *Domain Dispersed Querying:* Enables Teleport Exec's Domain Dispersed Querying™ algorithm, which dynamically shuffles the retrieval queue to distribute file and update requests across different servers, whenever possible. Distributing simultaneous file requests relieves transmission bottlenecks that could be caused by a single slow or deadlocked server. Default: On.

- s   *Server Overload Protection:* Enables Teleport Exec's Server Overload Protection algorithm. "Server overload" is a misnomer; in reality, the client's connection (via modem) is far more likely to become overloaded than the remote server. This algorithm moderates the number of active retrieval threads so that the amount of requested data does not exceed the network connection's ability to send it. Default: On.

- r   *Robot Exclusions System:* Enables Teleport Exec's Robot Exclusions System. This system provides compliance with the voluntary RES standard for excluding automated agents from parts of websites. Teleport Exec's RES system attempts to locate a robots.txt file on each domain it visits. If such a file is found, Teleport Exec parses it, then locates and obeys all "disallow" restrictions in the file. Default: On.

  The RES system will match all robots.txt directives that specify a user-agent of "*" (the global wildcard), or which contain the word "teleport" (case insensitive).

- w   *Wait Between Requests:* Causes Teleport Exec to wait the specified number of seconds between sequential file or update requests to the same server, when the server is reacting slowly (unless the *Force Wait* option is specified, see below). If zero or no delay time is given, waiting is disabled. Default: 1 second.

  Teleport Exec does not wait between requests to different servers. Also, Teleport Exec automatically disables waiting when pinging a server-side image map (see above).

- f   *Force Wait:* Causes Teleport Exec to wait between requests for all servers. If not checked, the delay time applies only to "slow" servers (see above. Default: Off.

`/pry[referrer]` *Set Referrer:* Forces the Referer [sic] HTTP header to the specified string. If a referrer string is not specified, Teleport uses the appropriate URL by default (normally, the linked-from page's URL). . If no string follows this switch, the existing setting is cleared. Default: Not set.

`/pri[a|t|m[version]|n[version]|c<identity>]` *Set Agent Identity:* Sets Teleport Exec's agent identity. These parameters correspond to those listed at the bottom of Teleport Pro's Project Properties, Netiquette page. Only one identity type may be specified. The identity options are—

- a   *Anonymous:* Sends no identity string.

- t   *Teleport Exec:* Identifies itself as the current version of Teleport Exec.

- m   *Microsoft Internet Explorer:* Identifies itself as Microsoft Internet Explorer, with an optional version string. This is the default identity setting.

- n   *Firefox:* Identifies itself as Mozilla Firefox, with an optional version string.

- c   *Custom Identity:* Identifies itself as the given identity string.

It is considered good netiquette for a client to identify itself, and to do so accurately. However, because many websites and webservers will modify their responses based on the client's identity (for example, some sites will not send frame files to clients they do not recognize as capable of handling frames), the user may find it useful to have Teleport Exec assume another identity.

`/prds<date>` *Set Start Date/Time Filter:* Sets Teleport Exec's start date filter to `<date>`, where `<date>` is given in the format `mm/dd/yy hh:mm:ss [am|pm]`. You may use 24-hour time, or specify am/pm. If the time (`hh:mm:ss`) field is omitted, the start time is assumed to begin at 12:00:00 am. (Any non-digit character may be used as delimiters in specifying the date, so you can substitute, for example, `mm/dd/yy-hh:mm:ss`, and omit to use quotes around the parameter.) You may also use 4-digit years; 2-digit years are interpreted as year 2000+ years if the value is less than 70. Default: No starting date filter.

When filtering by start date, Teleport Exec will **save** files that it retrieves only if (in addition to meeting all other filtering parameters) they are reported by the remote server to have a "Last Modified Date" **equal to or later than** `<date>`. When the remote server does not report the file's last modification date, Teleport Exec assumes that date/time to be the same as the date/time at which the file was retrieved (measured according to client machine's system clock).

Note that start date filtering is not performed on embedded files, when the parent file has already been loaded (i.e., the parent file has met all of the project specifications and has been saved to disk). This ensures that all of the components of a web page will be saved, so long as that page meets the project filtering requirements.

`/prde<date>` *Set End Date/Time Filter:* Sets Teleport Exec's end date filter to `<date>`, where `<date>` is given in the format `mm/dd/yy hh:mm:ss [am|pm]`. You may use 24-hour time, or specify am/pm. If the time (`hh:mm:ss`) field is omitted, the start time is field is omitted, the end time is assumed to end at 11:59:59 pm (midnight). (Any non-digit character may be used as delimiters in specifying the date, so you can substitute, for example, `mm/dd/yy-hh:mm:ss`, and omit to use quotes around the parameter.) You may also use 4-digit years; 2-digit years are interpreted as year 2000+ years if the value is less than 70. Default: No ending date filter.

When filtering by end date, Teleport Exec will **save** files that it retrieves only if (in addition to meeting all other filtering parameters) they are reported by the remote server to have a "Last Modified Date" **earlier than** `<date>`. When the remote server does not report the file's last modification date, Teleport Exec assumes that date/time to be the same as the date/time at which the file was retrieved (measured according to client machine's system clock).

Note that end date filtering is not performed on embedded files, when the parent file has already been loaded (i.e., the parent file has met all of the project specifications and has been saved to disk). This ensures that all of the components of a web page will be saved, so long as that page meets the project filtering requirements.

`/pro[p<name>]` *Set Other Parameters:* Sets or clears one or more of the project's "other" parameters. Unspecified options are given their default values. The "other parameters" are—

p *Product Name:* Causes Teleport Exec to use the name `<name>` when writing any user-readable files (such as message files created by the `/prl1` command). Default: "Teleport Exec."

`/prc` *Cautious Encoding:* Causes Teleport Exec to use "cautious" encoding when formulating URLs for retrieval. The cautious encoding scheme is designed to accommodate older servers that have trouble accepting unencoded "special" characters (such as ":").

`/prh<header>` *Set Additional Headers:* Sets optional additional headers that Teleport Exec will pass along with its requests to HTTP servers. Headers should be given in the form

```
field-name: field-value
```
To specify multiple headers, separate them with semicolons. To use a semicolon as a literal character within a field or value, prefix it with a backslash character.

While you may use this mechanism to pass any additional headers you wish, Teleport Exec may override some of them with its own headers if, for example, it is asked to identify itself as a browser.

You may use the following macros in any additional headers:

$URL   will be replaced by the requested URL

$FILE   will be replaced by the filename part of the requested URL (including query parameters)

$HOST   will be replaced by the domain name (host name) of the requested URL

/prg   *Synchronize local site:* When enabled, causes Teleport Exec to delete "dead" files from the project after exploration is complete. "Dead" files are those that Teleport retrieved in an earlier project session, but which are now not available on the server. (Remember that Teleport's default mode is to "accumulate" files, meaning that it retains files even after they're no longer on the server. Enabling synchronization changes this behavior.) The dead files will be deleted from the project folder, and links to them will be rewritten in the manner that Teleport normally rewrites dead links (controlled by the project's Browsing/Mirroring properties).

/prj   *Borrow cookies*: Instructs Teleport Exec to use Internet Explorer's cookies when it communicate with webservers. This feature is only compatible with Internet Explorer. To copy a site that requires form-based login or some other complex authentication procedure, enable this option, and then log in or otherwise access the website with Internet Explorer. Once you've completed the login using Internet Explorer, you can run the Exec project. (You can sometimes close Internet Explorer, but often it is best to leave it running, because some cookies will automatically expire when the browser is closed.) Teleport will pick up whatever cookies it needs from Internet Explorer, and use them when it crawls the site.

d. Switches for Running, Relinking, and Reporting on Projects

/a<connectionname> *Autoconnect:* Attempts to establish a connection to the Internet from the host computer, via Windows Dial-Up Networking, using the named RAS (Remote Access Service) entry. The parameter must be the complete name of the RAS entry; Teleport Exec will not search for a partial name match. If the connection is already open, Teleport Exec will not attempt to open it again. Teleport Exec will use the default parameters (account name, password, access number, and dialing attempts) stored in the RAS registry to establish the connection.

Special feature: A ? (question mark) symbol can be substituted for the connectionname parameter, to specify that Teleport Exec should connect using the user's default connection entry.

/ca<account> *Connection Account:* Specifies the user account for autoconnecting with dial-up networking. If this parameter is omitted, Teleport Exec will attempt to connect using the account stored in the Dial-Up Networking phonebook.

/cp<password> *Connection Password:* Specifies the user's password for autoconnecting with dial-up networking. If this parameter is omitted, Teleport Exec will attempt to connect using the password stored in the Dial-Up Networking phonebook.

/cn<number> *Connection Number:* Specifies the telephone number to dial for autoconnecting through dial-up networking. If this parameter is omitted, Teleport Exec will attempt to dial the number stored in the Dial-Up Networking phonebook.

/d *Disconnect:* Automatically disconnects the host computer from the Internet, by hanging up the RAS entry used to autoconnect, when the project is finished running, or when Teleport Exec exits, whichever occurs first.

/e *Explore in depth-first order:* Puts Teleport Exec in "depth-first" mode. In this mode, Teleport queues "deeper" files – files that are linked further from the starting address – for retrieval before shallower files, which results in a depth-first (rather than the default "breadth-first") exploration

order. Exploring in depth-first order has some side effects: (1) While Teleport will still attempt to get embedded images and other files required to make up a page at the same time as the page, most other links on a page will remain unresolved for a considerably longer time, on average, than when doing breadth-first exploration, so as a result it is less likely that shallow pages, especially pages near the starting address, will not be completely browsable until very near the end of the project session. (2) The "level complete" progress message is sent whenever Teleport reaches a new level for the first time, and does *not* indicate that the previous level is complete. (3) The retrieval queue is on average much shorter for depth-first exploration, which results in slightly better performance; but because link resolution is delayed, more file rewriting is required, and therefore overall performance is somewhat slower. However, if link localization is disabled, performance can be expected to increase very slightly.

/pxu[proxyaddress] *Proxy URL Address:* Sets the proxy server address used to connect to the Internet. The address should be given as a domain name, and does not require a complete URL specification. If no address is specified, any existing default proxy is cleared. When a proxy is specified, it becomes the program default, stored in the system registry for future use.

/pxo<proxyport> *Proxy Port:* Sets the port on which the proxy server is located. When a proxy port is specified, it becomes the program default, stored in the system registry for future use.

/pxa[proxyaccount] *Proxy Account:* Sets the user account for the proxy server. When a proxy account is specified, it becomes the program default, stored in the system registry for future use. If no account is given, this parameter is cleared.

/pxp[proxypassword] *Proxy Password:* Sets the password for the proxy server. When a proxy password is specified, it becomes the program default, stored in the system registry for future use. If no password is given, this parameter is cleared.

/r[r|u|c] *Run Project:* Begins running this project.

> /rr *Run Regular:* Runs the project in the normal fashion; all new (unretrieved) files in the project database are queued for retrieval, and all old (already retrieved) files are queued behind them, for updating. This is the default mode, if no run parameter is given.

> /ru *Run Update:* Runs the project in Update Mode; Teleport Exec will only update files already retrieved in this project, and will not retrieve new ones, unless they are discovered while updating an HTML file that contains new links.

> /rc *Run Complete:* Runs the project in Complete Retrieval mode; all eligible files in the database are queued for retrieval, whether previously retrieved or not.

/t<timelimit> *Time Limit:* Specifies a time limit for running the project, in minutes. The project will stop automatically if this limit is reached.

/f<spacelimit> *Free Space Limit:* Specifies a minimum disk free space limit, in megabytes. The project will stop automatically if this limit is reached. The free space calculation includes all temporary and operating system cache files open on the volume at run-time.

/z<sizelimit>[k] *Retrieved Size Limit:* Specifies the maximum total bytes that the project can retrieve and store into the project folder, in megabytes, in a single session. If the sizelimit parameter is followed by a "k" then the limit will be interpreted as kilobytes, not megabytes. The project will abort automatically if the size limit is reached. Note that this limits the number of bytes *added* to the project, not the total size of all files in the project folder.

/ts<foldersizelimit>[k] *Folder Size Limit:* Specifies the maximum total bytes that the project folder can store, in megabytes, in a single session. If the sizelimit parameter is followed by a "k" then the limit will be interpreted as kilobytes, not megabytes. The project will abort automatically if the size limit is reached. If the size limit is reached, the Teleport will iteratively relink, delete files, and rewrite the end-of-session log until the total size of the project folder is below the limit.

/l *Relink Project:* Relinks all HTML files in the project folder, using the project's current linking specifications. When this option is given, Teleport Exec will simply relink the files and exit, and will perform no other tasks.

/o *Clear Project Database:* Clears the project database, which removes all non-starting URLs, deletes all files in the project folder, and resets all status counters. When this option is given, Teleport Exec will clear the project database and exit, and will perform no other tasks.

/g<filename> *Generate end-of-session report:* At the end of a project session, writes a tab-delimited text report at the given filename, listing the status of all files in the project address database. If /g is specified but <filename> is omitted or <filename> contains only a path (*i.e.*, it ends in a backslash) the report filename defaults to the path given (or the current directory, if no path is given), plus the name of the project, plus the extension .rep.

/h<filename> *Generate end-of-session log:* At the end of a project session, writes a tab-delimited log entry at the end of the file with the given name, listing the time the project complete, the project name, and the completion status of the project. A typical entry might read, for example, "Project Completed. 57 files were read, of which 34 were saved to disk." If /h is specified but <filename> is omitted or <filename> contains only a path (*i.e.*, it ends in a backslash) the log filename defaults to the path given (or the current directory, if no path is given), plus the name of the project, plus the extension .log.

/i *Write Update Status Codes:* Alters the end-of-session report so that it prints the "update" status code for each address, rather than the "best" code (which is normally what is reported; see below). For updated files, the end-of-session report will list the status code received from the server *in response to the update request*. For new files, the status code will be the code returned by the ordinary GET request. For files that were not retrieved or updated, the status code will be zero.

/j<filename> *Generate Saved File Log:* Generates a saved file log for the session. The saved file log receives a new entry for each file that is written into the project folder, at the moment the file is written. Entries are written when files are saved after first being retrieved, and upon any subsequent updates; but not when a file is relinked. The log has the following format—

ID *tab* LastModifiedDate *tab* URL *tab* OriginalURL *tab* LocalPath *tab* Depth *CRLF*

The report's elements are—

ID   Each address in the Teleport Exec database is assigned a unique 32-bit identifier that remains constant for the life of the project. Calling applications can use this identifier to track changes to a file, regardless of where the file is later stored. The address identifier is also broadcast in the File Saved and File Relinked progress messages (see above).

LastModifiedDate   The file's modification date as reported by the remote server, in the form mm/dd/yyyy hh:mm:ss. If the server did not report a date, or the reported date was invalid, this time is the same as the date on which the file was saved, according to the local system clock. The date is reported in GMT.

URL   The URL to this file.

OriginalURL   If this URL was the result of a redirection from another URL, this field contains the originating URL. Otherwise, this field is blank.

LocalPath   The full path to this file.

Depth   The address's depth, or distance from its starting point, measured as the number of hyperlinks that must be followed to reach this address from the starting point. For more detail on the meaning of this field, please read its full description in the section describing the End of Session report format, below.

Each time an entry is made in saved file log, the **"Teleport Exec File Saved"** progress message is broadcast. The parameters of that message identify the file ID and the location within the logfile that the entry was made, enabling a listening application to determine the details of the newly saved file. For more information, see the section on Progress Messages, below.

`/debug` *Debug Interface:* Displays the program's debug interface, for development purposes. This interface shows the status line and thread states, and allows the developer to save, pause, stop, or abort the project.

`/m<filename>` *Generate Response File:* Generates a response file containing all the switches required to reproduce the current project. You can use this command to convert an existing Teleport Pro project into a response file that Teleport Exec can use to re-create the project. The response file will contain all of the commands necessary to create the project properties and the starting address(es) for the project. It will *not*, however, contain commands for non-project properties such as those for dialing a modem or using a proxy server.

`/v<string>` *Set Instance Identifier Value:* Assigns a string value to this instance of Teleport Exec. The instance value can be any string; it will be appended to the project name in the (usually hidden) main window for the program. The instance identifier allows you to identify a particular instance of Teleport Exec, which can be useful if running multiple projects with the same project name. Without an instance identifier, Teleport Exec's main window title will read "*project* – Teleport Exec"; with an instance identifier, the title will read "*project<instance>* – Teleport Exec" (where *project* is the name of the project, and *instance* is the assigned instance value).

`/s<IP>` *Bind to IP:* Binds Teleport to a specific IP local address. This option is useful only for multi-homed host machines. The IP must be given in standard dotted number format, *e.g.*, 192.168.0.1. Note that you cannot bind Teleport to a specific outbound port (*e.g.*, you cannot bind to 192.168.0.1:1390), since Teleport uses multiple connections and must therefore have access to the full range of outbound ports.

Progress Messages

During a project session, Teleport Exec broadcasts system-wide progress messages, via Windows™ messaging, to all running applications. Listening processes can handle these messages to learn of the project's progress on a file-by-file basis.

Teleport Exec broadcasts progress messages using the Win32 BroadcastSystemMessage() function. Messages are posted, not sent, so a listening process need not respond to the message. Because posted messages are placed immediately in the application message queue and not processed until the application is ready, these progress messages may lag slightly behind the events they report.

All progress messages have IDs in the range 0xC000–0xFFFF, the range newly reserved for private application messages. (This range was formerly reserved for Windows™.) The listening process can find the message ID for each message by calling the Win32 RegisterWindowMessage() function with the message identifier string.

Teleport Exec posts the following progress messages—

> **Message ID="Teleport Exec Starting":** Teleport Exec broadcasts this message just before it begins running a project.
>
> WPARAM = 0.
> LPARAM = the window handle of the Teleport Exec instance window.

> **Message ID="Teleport Exec Thread Completed":** Teleport Exec broadcasts this message whenever a retrieval thread terminates, whether successfully (file retrieved or updated) or unsuccessfully (file unavailable). Since threads terminate with some frequency, this message serves as a constant status update on the project's overall progress.
>
> WPARAM = length of the remaining retrieval queue.
> LPARAM = the window handle of the Teleport Exec instance window.

> **Message ID="Teleport Exec File Saved":** Teleport Exec broadcasts this message whenever it saves a file in the project folder. Note that a file save may occur after a file is retrieved, updated, or relinked.
>
> WPARAM = the file's 32-bit identifier (see End-of-Session reports, below).
> LPARAM = the window handle of the Teleport Exec instance window.

> **Message ID="Teleport Exec File Logged":** Teleport Exec broadcasts this message whenever it makes an entry in the Saved File log. (This occurs when the project is run with the "\j" option; see the section on switches for running and relinking projects, above.) This message is always followed by a Teleport Exec File Saved message. Unlike the File Saved message, however, this message is only sent after an entry is put in the Saved File log, which occurs only when a file is retrieved or updated (never when relinked).
>
> WPARAM = the offset in the logfile of the beginning of the entry for this item.
> LPARAM = the window handle of the Teleport Exec instance window.

> **Message ID="Teleport Exec Level Complete":** Teleport Exec broadcasts this message whenever it *probably* has finished downloading all files at a given depth. The message is broadcast in ascending depth order. Teleport normally explores in a breadth-first manner, exploring all files at depth zero before moving on to files at depth 1, depth 2, and so on. This message is sent just after Teleport receives what it believes is the first file at a new depth level. It is only an *approximate* notification of this event, because many factors affect the way in which the retrieval queue is prioritized: domain spreading, re-queuing items that were incompletely loaded or which failed to load, and the need to gather embedded files before non-embedded files all can change the breadth-first ordering slightly, but sometimes enough that this message will be sent before a level is actually complete, or significantly after the level is complete.

If depth-first exploration is requested (using the /e command), then this message is still sent, but it instead indicates only that Teleport has reached a new level for the first time — and *not* that the previous level has been completed.

Note that a level completion notice is *not* sent when the project is finished. You may assume upon completion that all levels are complete.

Note also that level completion notifications will also be sent when a project is *updated*. In this case, they indicate that (again, approximately) all files on a given level were updated.

WPARAM = the depth level just completed.
LPARAM = the window handle of the Teleport Exec instance window.

Message ID="Teleport Exec File Relinked": Teleport Exec broadcasts this message whenever it dynamically relinks an HTML file, during a project session. A **File Saved** message will always immediately precede this message. End-of-session relinking does not generate this message.

WPARAM = the file's 32-bit identifier (see End-of-Session reports, below).
LPARAM = the window handle of the Teleport Exec instance window.

Message ID="Teleport Exec Dialing": Teleport Exec broadcasts this message whenever it begins dialing an ISP through Dial-Up Networking.

WPARAM = 0.
LPARAM = the window handle of the Teleport Exec instance window.

Message ID="Teleport Exec Dialer Successful": Teleport Exec broadcasts this message whenever it successfully dials an ISP through Dial-Up Networking.

WPARAM = 0.
LPARAM = the window handle of the Teleport Exec instance window.

Message ID="Teleport Exec Dialer Failed": Teleport Exec broadcasts this message whenever it cannot connect to an ISP through Dial-Up Networking for any reason (*e.g.*, no dial tone, wrong number, bad password, service unavailable).

WPARAM = 0.
LPARAM = the window handle of the Teleport Exec instance window.

Message ID="Teleport Exec Connect Successful": Teleport Exec broadcasts this message the first time it successfully connects to *any* Internet server, which usually occurs at the start of a project session. Note that Teleport Exec will attempt to determine if it is connected by querying a number of "known good" servers, including those for Microsoft, Google, and IBM, if initial queries to the project's starting addresses fail.

WPARAM = 0.
LPARAM = the window handle of the Teleport Exec instance window.

Message ID="Teleport Exec Connect Failed": Teleport Exec broadcasts this message whenever it determines that it cannot connect to *any* remote server on the Internet. Before sending this message, Teleport Exec (1) attempts to connect at least once to every server that hosts an item in its current retrieval queue; and (2) attempts to connect to any of a set of "known good" servers, including those for Microsoft, Google, and IBM. Only after Teleport Exec fails to connect to any of these servers does it send this message; thus, this message is a good indication that the client's local network is down.

WPARAM = 0.
LPARAM = the window handle of the Teleport Exec instance window.

Message ID="Teleport Exec Finished": Teleport Exec broadcasts this message just before it terminates.

WPARAM = 0
LPARAM = the window handle of the Teleport Exec instance window

End-of-Session Report and Log

Teleport Exec can generate two files at the end of a project session, a report and a log file. The **report** lists the current status of all of the files in the project database. The **log file** lists the time the project was run and its completion status.

a.   The End-of-Session Report

At the end of a project session, if the `/g` switch was specified, Teleport Exec generates a tab-delimited text report on the status of each file in the project's address database. The report has the following format—

```
ID tab Status tab LinkType tab Date tab LastModifiedDate tab URL tab
OwnerURL tab LocalPath tab ModifyCode tab ContentType tab Depth tab
RedirectURL CRLF
```

The report's elements are—

ID   Each address in the Teleport Exec database is assigned a unique 32-bit identifier that remains constant for the life of the project. Calling applications can use this identifier to track changes to a file, regardless of where the file is later stored. The address identifier is also broadcast in the File Saved and File Relinked progress messages (see above).

Status   The retrieval status code for the address. For files that were queued for retrieval, this code is the HTTP response code sent by the webserver, usually 200 for retrieved files, 302 for redirected files, and 400–500 for not found or unavailable files.

Note that unless the *Report Update Status Codes* (`/i`) option is enabled, the status code reported will be the "best" status code received so far in the project. That is, if previous attempts to get a file were successful, the status code will be 200, even if later attempts fail. The "best" code is intended to represent the state of the mirrored site, not the current state of the remote server. If you wish to display the most recently obtained status code (*i.e.*, the code received on update attempts), use the `/i` option to display those status codes instead. (See above for more detail.)

Status codes above 600 are assigned by Teleport Exec, and indicate that a file was not queued for retrieval, or was retrieved but discarded, for failure to meet project specifications. The following is the list of currently defined status codes—

UNSUPPORTEDPROTOCOL	700
UNWANTEDTYPE	701
UNWANTEDJAVA	702
UNWANTEDFRAME	703
UNWANTEDFORM	704
UNWANTEDEMBEDDED	705
UNWANTEDBACKGROUND	706
EXCEEDSINNERDEPTH	707
EXCEEDSOUTERDEPTH	708
EXCEEDSINNERBOUNDS	709
EXCEEDSOUTERBOUNDS	710
EXCLUDEDTYPE	711
EXCLUDEDPATH	712
UNWANTEDSERVERMAP	713
EXTERNALEXTERNAL	714
ROBOTSEXCLUDED	715
OVERSIZEDFILE	716
UNDERSIZEDFILE	717
NOKEYWORDS	718
OWNERNOKEYWORDS	719
EXPLOREONLY	721

MATCHEDCRC	722
FAILEDCONNECT	723
HTMLEXPLORED	724
TOOOLDFILE	725
TOONEWFILE	726
<RESERVED>	727
INVALIDREDIRECT	728
CIRCULARREDIRECT	729
UNNEEDED	730
DELETED	731
FAILEDSSLHANDSHAKE	732
INVALIDADDRESS	799

The EXPLOREONLY and HTMLEXPLORED status codes indicate files which are marked for exploration but not retrieval; HTMLEXLORED files were successfully read, but not saved to disk.

The DELETED status code indicates that a file was deleted in order to prevent the project from exceeding a limit condition—for example, the project's total folder size limit. The UNNEEDED status code corresponds to URLs such as java applet codebases, which are not URLs to retrievable files.

The MATCHEDCRC status code is defined for development use only and is unused in this version of Teleport Exec.

LinkType   The type of the first link that pointed to this address. The following is the list of currently defined link types—

INVALID	0
NORMAL LINK	1
IMAGE SOURCE	2
BACKGROUND IMAGE	3
FRAME SOURCE	4
NORMAL FORM ACTION	5
JAVA APPLET	6
REDIRECTED	7
META REFRESH	8
BASE HREF	9
JAVA CODEBASE	10
SERVER MAP	11
OLD JAVA APPLET	12
OLD JAVA CODEBASE	13
LOW IMAGE SOURCE	14
DYNAMIC IMAGE SOURCE	15
OPTION FORM ACTION	16
DATA FORM ACTION	17
OBJECT (ACTIVEX)	18
OBJECT ARCHIVE	19
OBJECT RESOURCE	20
OBJECT CODESOURCE	21
STYLESHEET RESOURCE	24
STYLESHEET	26
SCRIPT SOURCE	27

Date   The date and time on which the file was last retrieved or updated, in the form mm/dd/yyyy hh:mm:ss. This date is written in local time (not GMT).

`LastModifiedDate`   The file's modification date as reported by the remote server, in the form `mm/dd/yyyy hh:mm:ss`. If the server did not report a date, or the reported date was invalid, this time is the same as the `Date` field, above. This date is, however, written in GMT.

`URL`   The URL to this file.

`OwnerURL`   The URL to the "owning" web page for this file. The owning web page is the first web page Teleport encountered during its exploration, that linked to this file.

`LocalPath`   The full path to this file, if successfully retrieved and saved in the project folder.

`ModifyCode`   A code that indicates whether the file appeared to be new or modified *in this project session*. If the webserver reported that the file was new or modified (indicating either that this is the first time that Teleport Exec has retrieved the file, or that the server believes the file has changed since the last time that Teleport Exec retrieved the file), then this field will contain the letter "m." Otherwise, this field is blank.

Note that file modification is tracked on a session-by-session basis; at the start of a session, all files are considered to be unmodified. Note also that some webservers, especially those serving database- or script-generated files, may report that files are modified, even though their actual content has remained unchanged.

`ContentType`   The MIME content type for this file, as reported by the server. This is the same information listed in Teleport Pro under the "Type" column of the file list. Common MIME types are "text/html," "image/gif," and "image/jpeg." Most servers give files of unknown type the designation "application/octet-stream." If Teleport Exec has not retrieved this file, this field is left blank.

`Depth`   The address's depth, or distance from its starting point, measured as the number of hyperlinks that must be followed to reach this address from the starting point. For addresses on the same domain as a starting address, the depth is the distance from the starting address. For "external" addresses, or those that are on a domain not the same as any starting address, the depth is the distance from the earliest "gateway" address on the starting domain. (The "gateway" address is the first address that leads from the domain of a starting address to the domain of the destination address.) A starting address, therefore, always has a depth of zero. The depth for external addresses is given as a negative number; gateway addresses, therefore, have a depth of –1. All other addresses have positive depth.

Generally, embedded files—those that are required to make up a web page—are considered to have the same depth as the page they make up. For instance, a graphic file will have the same depth as the page that it is embedded on; and a frame source file will have the same depth as the frame master file. Where a file can have more than one depth—for instance, a graphic that is embedded on more than one web page— the lowest possible depth is listed.

For more information on how the depth field is calculated and used, see the section above on setting starting address parameters.

`RedirectURL`   When the file for this address is reported to be at a different location (*i.e.*, the client has been "redirected" by the server to a new address), this field will contain the redirection address. Where there is no redirect, or where the redirect is invalid (some servers send circular or invalid redirections), this field is left blank.

b.   The End-of-Session Log

At the end of a project session, if the `/h` switch was specified, Teleport Exec ***appends*** to the given log file a tab-delimited entry listing the date the project was run, its path, and its completion status. Because this command appends the log entry instead of creating a new file, you can use it to log the completion status of several projects to the same file. Conversely, if you want the log file to list only the last completion date and status, then you should delete the log file before running the project.

The end-of-session log has the following format—

    Date *tab* Path *tab* Status *CRLF*

The report's elements are—

Date   The time and date the project was last run, given in local time in the format `mm/dd/yyyy hh:mm:ss`.

Path   The complete path to the project.

Status   The completion status of the project. This is the same text you might see in the "Last Result" field of Teleport Pro Scheduler's schedule list. A typical completion status line reads, "Project Completed: 152 files were read, of which 137 were saved to disk."

A completion status line will always begin with the words "`Project Completed`" if the project completed normally (*i.e.*, the project was not aborted or stopped for any reason during the session). If the project was aborted, either by the user (by pressing the Abort button on the debug panel), or for cause (a space limit condition was reached, or an initial connection error prevented the project from running at all), the status line will begin with the words, "`Project Aborted`". If the project was stopped by the user (by pressing the Stop button on the debug panel), or stopped for cause (a project time limit was reached), the status line will begin with the words, "`Project Stopped`".

Runtime Control

Although it can be run from the command-line, Teleport Exec is not a "console" application and, once started, it cannot be terminated by issuing a Ctrl-C or closing the command window. The process can be properly shut down by pressing the "Abort" or "Stop" buttons on the debug interface, or by closing the interface window (which causes an immediate, graceful abort)—but these techniques are available only if the debug interface has been requested (by specifying the `/debug` parameter).

However, any running instance of Teleport Exec can also be controlled by executing `exec.exe` (or `execvlx.exe`, for Teleport Exec/VLX) with a "runtime" command-line. Runtime commands are available even if the debug window is not invoked. The two runtime commands are—

`/a[project|id]` *Abort:* Causes any running instance containing *project* or *id* (such as an id specified by the the /v command) immediately to abort, shutting down all running threads. Any partially retrieved files will be lost. This is the same as pressing the "Abort" button on the debug window's toolbar.

`/s[project|id]` *Stop:* Causes any running instance containing *project* or *id* (such as an id specified by the the /v command) to stop further retrieval, but any running threads will continue retrieving until they are finished. When all of the threads finish, the program will shut down. This is the same as pressing the "Stop" button on the debug window's toolbar.

Program Specifications

Executable Type:	All code Win32 rev 4.0, subsystem rev 4.0, for I386 CPUs.
Operating Systems:	Any Win32 operating system.
Imports:	USER32, KERNEL32, GDI32, COMDLG32, ADVAPI32, SHELL32, COMCTL32.
Runtime bindings:	WSOCK32 (required), RASAPI32 (when available).
Threading structure:	Multithreaded. Synchronization is achieved primarily through Windows™ messaging, although in limited circumstances process-specific mutexes are used.
	The retrieval engine and parsing engine are both multithreaded and run asynchronously, with the retrieval engine getting execution priority. Primary parsing is performed on an on-demand basis; relinking is performed on an as-available basis throughout the project session. Relinks not processed during the normal course of the project are handled in a single thread at end-of-session.
	The retrieval engine is limited to no more than 10 threads. The parsing engine is limited to no more than 8 threads.
Retrieval Engine:	Robust file retrieval from HTTP/0.9–, HTTP/1.0–, and HTTP/1.1–compliant webservers. Hand-tuned header parser accommodates non-compliant webservers (*e.g.*, some CERN and Apache implementations). Robust file retrieval from FTP servers compliant with RFCs 959 and 1123 (anonymous access only).
Parsing Engine:	Performs robust bottom-up single-pass HTML parsing and relinking. HTML 5 compliant. The parser is hand-tuned for efficiency and to accommodate common HTML authoring errors (missing delimiters, illegal characters). Handles all accepted methods of URL notation, including hexadecimal escape codes, nonstandard ports, SGML entities, ISO-8859-1 entities, bookmarking, and query specifiers. The parser is also capable of properly handling Internet addresses given as DBCS or Unicode strings.
Database Engine:	Proprietary. Optimized for lookup performance to handle the querying load imposed by simultaneous file parsing. Maximum size: 64k addresses. (Exec/VLX database size is 40M addresses.)

For Further Information

If you have any questions about this document, please contact us at:

Tennyson Maxwell Information Systems, Inc.
PO Box 34
Garrison, NY 10524

+1 845-214-0633 voice
+1 815-461-9518 fax
email: support@tenmax.com