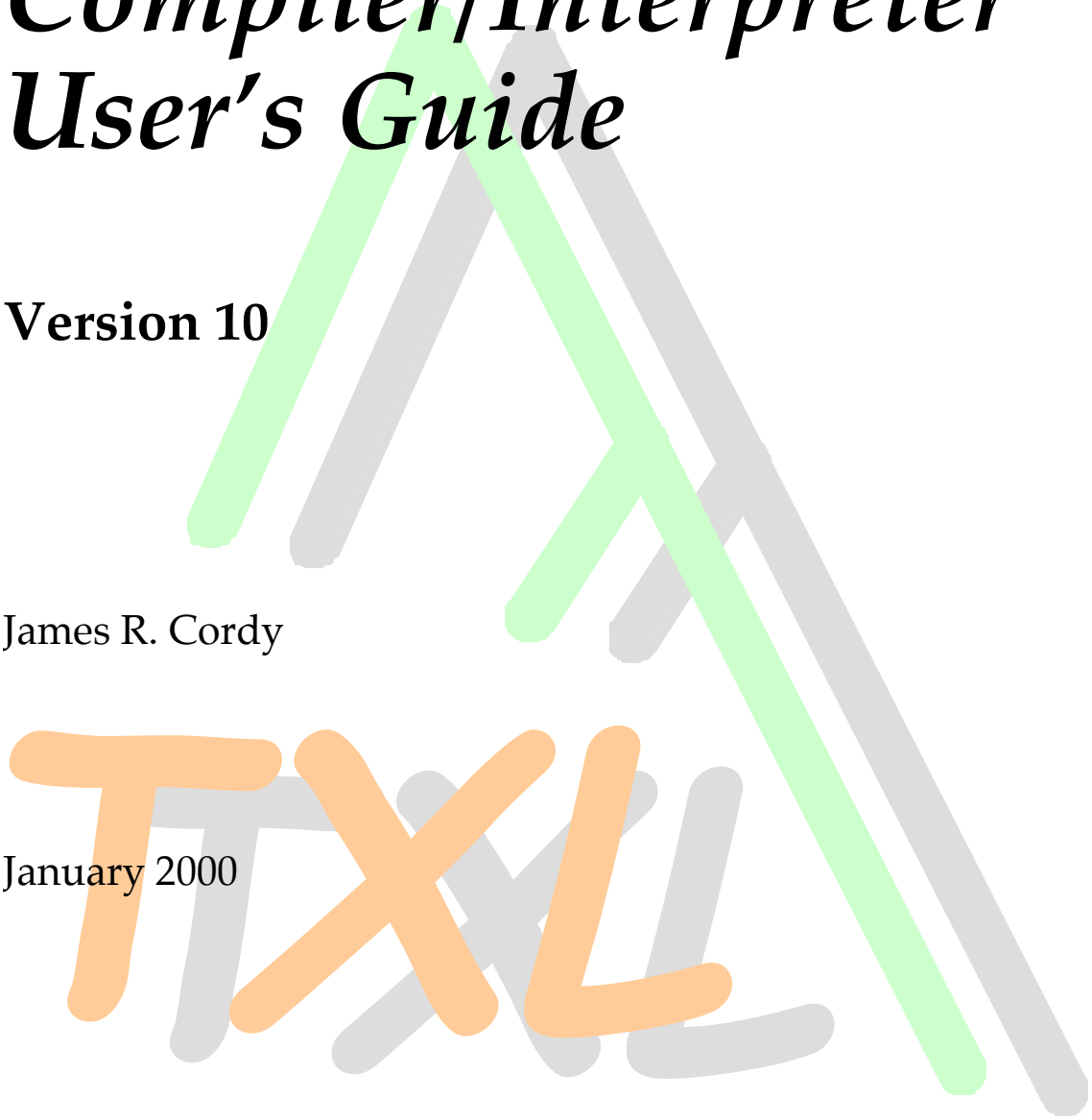
The TXL Compiler/Interpreter User's Guide

Version 10

James R. Cordy



January 2000

James R. Cordy

The TXL Compiler/Interpreter User's Guide
Version 10

Legasys Corporation
366 King Street East
Kingston, Ontario K7L 6Y3
Canada

+1 (613) 545 9454 • legasys@legasys.on.ca

Table of Contents

1. The TXL Compiler / Interpreter

The TXL compiler/interpreter is the language processor and executor for TXL language programs. TXL is a programming language specifically designed to support transformational programming. The basic paradigm of TXL involves transforming input to output using a set of structural transformation rules that describe by example how different parts of the input are to be changed into output. Each TXL program defines its own context free grammar according to which the input is to be parsed, and rules are constrained to preserve grammatical structure in order to guarantee a well-formed result.

1.1 TXL Processor Commands

The TXL transformation system provides three commands for compiling, executing and debugging TXL programs. The *txl* command invokes the TXL compiler/interpreter to compile, load and execute a TXL transformation on an input file. The *txldb* command additionally invokes the TXL rule debugger to allow stepwise execution of the TXL program in an interactive debugging environment. The *txlprof* command compiles and runs a TXL program with grammar and ruleset profiling, producing a detailed time and space profile of the program's input parse and ruleset execution for a given input. The command *txlapp* uses the TXL compiler to translate a TXL program directly to a standalone executable application.

1.2 File Arguments and Option Flags

The *txl* and *txldb* commands normally take only one file argument, the name of the input file to be transformed. By convention the input file name is expected to be of the form *inputfile.dialect* where *dialect* is the name of the TXL transformation to be run on the file. The source for the TXL program for the corresponding transformation is then assumed to be in the file *dialect.Txl* . For example, if the command

```
txl Expression.Calculator
```

were to be run, the TXL processor would assume that the TXL program to be run is *Calculator.Txl* .

The TXL program to be run need not be in the present working directory, but may instead be in either the user's TXL library directory, *./Txl* , or the system's TXL library directory, normally */usr/local/lib/Txl*. For example, if the command above were run, then the program *Calculator.Txl* could be either in the present working directory (i.e., file *./Calculator.Txl*), the *Txl* subdirectory of the present working directory (i.e., file *./Txl/Calculator.Txl*), or the system TXL library (i.e., file */usr/local/lib/Txl/Calculator.Txl*). In cases where an appropriate TXL source program is present in more than one of these, the user's library takes precedence over the system library, and the present working directory takes precedence over the user's library.

Files referred to by *include* statements in the TXL program are resolved in similar fashion. Included files are searched for first in the directory of the including source file, then in the *Txl* subdirectory of the present working directory, then in the system TXL library directory.

It is also possible to explicitly specify the TXL program to be run using two file arguments, for example:

```
txl count.c c2p.Txl
```

In this case the name of the input file need not be named using the suffix corresponding to the TXL program. In all cases, TXL main program files must be named using the *.Txl* suffix.

All TXL processor commands implement several standard options, including *-V* and *-help*. The *-V* option always prints the TXL processor version information to the standard error stream and halts. The *-help* option prints a short explanation of command usage and options to the standard error stream and halts.

2. The 'txl' Command

txl [*options*] *inputfile.dialect* [*dialect.Txl*] [- *useroptions*]

txl -compile [*options*] *dialect.Txl*

The *txl* command provides a convenient interface for compiling, executing and transforming input files using TXL programs. The first command argument, *inputfile.dialect*, specifies the input file to be transformed by the TXL program, and the second argument, *dialect.Txl*, is the TXL program itself. Under normal circumstances only the first file argument, the input file to be transformed, is given, and the name of the TXL program file is inferred from its suffix. For example, if the given input file to be transformed is *myinput.pas* then the the TXL program file is inferred to be *pas.Txl* . If the second argument is given then it is taken to be the name of the TXL program to be run and overrides any inference from the input file suffix. Whether inferred or explicitly given, the TXL program file must be named ending in the suffix *.Txl* and must be either in the current working directory, in the *Txl* subdirectory of the current directory, or in the system TXL library (e.g., */usr/local/lib/txl*).

Unless the *-compile* flag is given, in which case execution is suppressed, *txl* normally compiles and executes the TXL program to transform the given input file. Transformed output (only) is sent to the standard output stream and may be redirected to a file. TXL progress and error messages are sent to the terminal independently via the standard error stream.

2.1 'txl' Command Option Flags

The following options are recognized by the *txl* command.

`-q[uiet]` Quiet operation - turn off all progress messages.

The TXL processor normally prints a version identification and short progress messages indicating the stage of processing on the standard error stream as it runs. The *-quiet* option suppresses all such messages, and allows TXL to print error messages only.

`-v[erbose]` Verbose operation - give greater detail in progress messages.

Causes TXL to print more detailed progress messages giving information on space and files used by each stage of processing. Also enables a number of common warning messages that are suppressed by default.

`-c[ompile]` Compile TXL program to tree code only (do not execute).

The compiled tree code for the TXL program *dialect.Txl* is output to the file *dialect.CTxl* . The tree code file can be loaded and executed directly by subsequent txl commands using the *-load* option, thus avoiding the overhead of recompiling the program on every run. Tree code files can also be converted to C and compiled for use in standalone TXL applications by the *txlapp* command (see "The 'txlapp' Command" below).

`-l[oad]` Load and transform input using a previously compiled TXL program.

The compiled tree code for the TXL program *dialect.Txl* is loaded directly from the file *dialect.CTxl*, which must be a TXL tree code file created by a previous *txl* command that specified the *-compile* option. The name of the TXL tree code file can either be inferred from the suffix of the first argument file or be explicitly given as the second command argument. In either case the suffix of the TXL program file is ignored and *.CTxl* is used instead.

`-d[efine] SYMBOL` Define the TXL preprocessor symbol SYMBOL.

Sets the preprocessor symbol SYM to defined, so that *#ifdef SYMBOL* preprocessor directives will succeed in the compile of the TXL program.

`-comment` Treat comments in the input file as input items.

Normally TXL discards any comments in the input before parsing. When this option is used, comments are treated as input items to be parsed like any other. Care must be take to insure that the input language grammar specified in the TXL program explicitly allows comments in all the expected places, otherwise syntax errors will be flagged.

`-char[input]` Treat all input characters (including newlines and spaces) as significant.

Normally TXL treats newlines, spaces and tabs ("white space") in input as separators only. When this option is specified, all characters in the input are treated as significant and categorized as tokens to be parsed according to the grammar. This option enables the predefined token classes [space] (any sequence of spaces and tabs) and [newline] (a newline character). The grammar must be crafted to accept newlines and spaces wherever they may occur in the input. This option automatically suppresses all output spacing.

`-token[input]` Treat newlines and spaces as separators only.

Disables character level input, and treats all newlines, spaces and tabs as separators only. This is the default input mode.

`-txl` Treat the input being transformed as TXL source.

This option makes it possible to transform TXL programs themselves using TXL. Sets the lexical conventions of the input to TXL's own defaults. In particular, disables unquoted character literals and enables treatment of single quotes as separate input tokens.

`-card` Treat the input as IBM 80-column Cobol card images.

Trims columns 1 through 5 and 73 through 80 (or more) off of input lines before scanning.

`-cobol` Treat the input as Tandem/Microfocus format Cobol.

Implements Cobol continuation and comment conventions, using column 1 as the indicator column. If used with *-card*, input card images are trimmed first.

`-ibmcobol` Treat the input as IBM mainframe Cobol.

Same as *-card -cobol*.

`-tandemcobol` Treat the input as Tandem Cobol.

Same as *-cobol*.

`-cobolout` Format output as IBM mainframe Cobol card images.

Formats output as 80 column IBM Cobol card images, using IBM Cobol continuation and column conventions.

`-lscobol` Handle input and output in LS/Cobol format.

Both input and output are treated as LS/Cobol format, using column 1 as the Cobol indicator column (normally blank) and using no continuations (i.e., wide lines).

`-attr` Print attributes in the transformed output.

Normally attributes (items of type `[attr X]` for some X) are not printed in the output of a TXL program.

`-raw` Output transformed source in raw (unspaced) format.

Normally TXL uses a set of built-in spacing rules appropriate to most high-level programming languages for formatting output. This option turns off all spacing and line wrapping in output except where the output would otherwise be ambiguous (e.g. between two adjacent identifiers), and as explicitly specified in the grammar using [SP] and [NL]. When used in conjunction with [SP], [NL] and [IN]/[EX], this option gives the user complete control over output formatting.

`-id[chars] 'CCC'` Treat the characters 'CCC' as valid following characters in [id] tokens.

Adds the given characters as valid following (second and subsequent) characters in the [id] predefined token class. More general control over [id] can be achieved using the *tokens* statement in the TXL program.

`-sp[chars] 'CCC'` Treat the characters 'CCC' as white space.

Adds the given characters to the set of characters treated as equivalent to the space character and treated as separators. If *-charinput* is specified, these characters will be added to those matched in the [space] predefined token class.

`-esc[char] 'C'` Use 'C' as the escape character in string and character literals.

Sets the string and character literal escape character to 'C'.
By default the escape character is '\' (e.g., `"Here: \" is an embedded quote"`.
If either `"'"` or `'"'` is specified, then `'"'` is used for string literals
(e.g., `"Here: "" is an embedded quote"`) and `"'"` is used for character literals
(e.g., `'Here: '' is an embedded quote'`).

`-upper` Translate all unquoted input to upper case.

Translates all input tokens except [stringlit] and [charlit] to upper case on input.

`-lower` Translate all unquoted input to lower case.

Translates all input tokens except [stringlit] and [charlit] to lower case on input.

`-w[idth] NNN` Set the maximum output line width to NNN characters.

By default TXL formats output in at most 80 characters per line. NNN must be a positive integer between 20 and 32767. This option has no effect when the *-raw* option is used.

`-in[dent] NN` Set the output indentation increment to NN characters.

Sets the number of character positions indented by [IN] and exdented by [EX] directives to NN characters. The default is 4 characters.

`-tabnl` Output [TAB_NN] directives may force a newline.

Allows [TAB_NN] directives to force a new line in the output if necessary to align the next output token at output column NN. This is the default.

`-I[nclude] DIR` Add DIR to the TXL include file search path.

Adds DIR to the set of directories searched for TXL *include* files that are not present in the original compilation directory or its Txl subdirectory. The directories are searched in the order that their -I directives are given on the command line. If an *include* file is not found in any of these directories, the TXL system include directory (normally */usr/local/include/txl)* is searched last.

`-s[ize] MM` Set the TXL transform size to MM megabytes.

Sets the virtual memory allocated to TXL compiler and transformer data structures to the indicated size. MM must be a positive integer between 2 and 999. In order to maximize transform efficiency, TXL liberally exploits the operating system's native virtual memory by artificially pre-allocating a fixed amount of static storage rather than attempting to manage storage dynamically. This strategy places static limits on the size of the input that can be processed as well as on the complexity and depth of the transformation that can be performed. This option explicitly sets the amount of static storage available and thus can be used to increase these limits. This option is normally used only when the system default limits have been exceeded or when permanently trimming the virtual memory used by a compiled TXL application. See also the *-usage* option.

`-u[sage]` Report TXL resource usage statistics at the end of the run.

Prints a table of the static limits on the various TXL internal data structures and the amount of each actually used by the TXL run. This option can be used to choose an appropriate transform size for typical input data (see the *-size* option above).

`-o[utput] FILE` Write standard output to file FILE.

Normally TXL writes the output of the run to the standard output. This option redirects output to the specified file instead.

`-noOPT` Turn command line option OPT off.

Explicitly turns off any command line option (e.g., *-noraw*).

`- USERARGS` Pass all remaining command arguments to the TXL program as user command line arguments.

Passes all following command line arguments to the TXL program in the predefined global variable *TXLargs*. For example, the command line :

```
txl eg.in in.Txl -s 100 - -myopt foo -otheropt
```

will initialize *TXLargs* to the [**repeat** stringlit] value `"-myopt" "foo" "-otheropt"`.

2.2 'txl' Command Debugging Options

The following additional debugging options are recognized by *txl* for use by experienced TXL users. Some or all of these options may be disabled in versions of the TXL processor that have been tuned for speed (but they are always available in *txldb*). All output produced by these options is sent to the terminal via the standard error stream.

`-V[ersion]` Print TXL processor version information on the standard error stream and halt.

`-L[ibrary] DIR` Use the version of the TXL library stored in directory DIR.

Runs the version of the TXL compiler/interpreter in directory DIR rather than the installed default (normally */usr/local/lib/txl*). This option allows the use of two or more different versions of TXL on the same machine.

`-analyze` Use extended grammar and rule set analysis.

Causes the TXL compiler to perform an additional set of checks on the grammar and rule set, including a check for ambiguities and potential efficiency issues in the grammar. This option may significantly slow down the compile.

`-Dgrammar` Print the input language grammar "tree" to the standard error stream.

TXL compiles input language grammars to a compact generic tree format used as both a pattern and a generator for parse trees of input. This option prints the compiled grammar tree for use in debugging subtle problems with a grammar.

`-Dparse` Print the input parse tree to the standard error stream.

Print the input parse tree on the terminal via the standard error stream. Useful when debugging grammars, or in understanding why a pattern has not matched the input.

`-Dfinal` Print the output parse tree to the standard error stream.

Print the output parse tree on the terminal via the standard error stream. Useful when debugging grammars, or in understanding why output is not formatted as expected.

`-Dpattern` Print all pattern and replacement parse trees to the standard error stream.

The parse tree of every pattern and replacement in the TXL program's rule set is printed on the terminal via the standard error stream. This option is useful in debugging patterns of rules that fail to match input as expected.

`-Dtrees` Print parser progress in building pattern and input parse trees as we parse.

An extremely verbose trace of the backtracking used in resolving parse trees for both rule patterns and input is printed on the terminal via the standard error stream. Not normally of interest to most TXL users.

`-Dapply` Print out the actual transformations made by rule applications on the terminal as they happen.

Useful for following the progress of a transformation. Transformations are output in the form A => B, where A and B are the text output form of the original and result scope of the transforming rule respectively.

`-Drules` Print out the names of rules to the standard error stream as they are applied.

A convenient trace of the order in which the rules and functions of the TXL program are actually invoked.

`-Dsharing` Print tree sharing optimization information on the terminal as rules are applied.

This option is ignored unless -Drules is specified as well. Used in debugging the TXL processor itself and is not normally of interest to most TXL users.

`-Dall` Turn on all of the options *-Dparse -Dapply -Dtrees -Drules -Dfinal* (deadly verbose!).

3. The 'txldb' Command

`txldb` [*`txloptions`*] *`inputfile.dialect`* [*`dialect.Txl`*] [*`- useroptions`*]

The *txldb* command provides a convenient interface for compiling, executing, debugging and transforming input files using TXL programs. The first command argument, *inputfile.dialect*, specifies the input file to be transformed by the TXL program, and the second argument, *dialect.Txl* , is the TXL program itself. Under normal circumstances only the first file argument, the input file to be transformed, is given, and the name of the TXL program file is inferred from its suffix. For example, if the given input file to be transformed is *myinput.pas* then the the TXL program file is inferred to be *pas.Txl* . If the second argument is given then it is taken to be the name of the TXL program to be run and overrides any inference from the input file suffix. Whether inferred or explicitly given, the TXL program file must be named ending in the suffix *.Txl* and must be either in the current working directory, in the *Txl* subdirectory of the current directory, or in the system TXL library (e.g., */usr/local/lib/txl*).

txldb compiles and executes the the TXL program under control of an interactive debugging interface to transform the given input file. Transformed output (only) is sent to the standard output stream and may be redirected to a file. Debugger prompts and error messages are sent to the terminal independently via the standard error stream, and debugger commands are accepted from the standard input.

3.1 'txldb' Command Option Flags

The following TXL options are recognized by *txldb*. For more detail on TXL options see the section "The 'txl' Command" above.

`-q[uiet]`	Quiet operation - turn off all progress messages.
`-v[erbose]`	Verbose operation - give greater detail in progress messages.
`-d[efine] SYMBOL`	Define the TXL preprocessor symbol SYMBOL.
`-comment`	Treat comments in the input file as input items.
`-char[input]`	Treat all input characters (including newlines and spaces) as significant.
`-token[input]`	Treat newlines and spaces as separators only.
`-txl`	Treat the input being transformed as TXL source.
`-card`	Treat the input as IBM 80-column Cobol card images.
`-cobol`	Treat the input as Tandem/Microfocus format Cobol.
`-ibmcobol`	Treat the input as IBM mainframe Cobol.
`-tandemcobol`	Treat the input as Tandem Cobol.
`-cobolout`	Format output as IBM mainframe Cobol card images.
`-lscobol`	Handle input and output in LS/Cobol format.

`-attr`	Print attributes in the transformed output.
`-raw`	Output transformed source in raw (unspaced) format.
`-id[chars] 'CCC'`	Treat the characters 'CCC' as valid following characters in [id] tokens.
`-sp[chars] 'CCC'`	Treat the characters 'CCC' as white space.
`-esc[char] 'C'`	Use 'C' as the escape character in string and character literals.
`-upper`	Translate all unquoted input to upper case.
`-lower`	Translate all unquoted input to lower case.
`-w[idth] NNN`	Set the maximum output line width to NNN characters.
`-in[dent] NN`	Set the output indentation increment to NN characters.
`-tabnl`	Output [TAB_NN] directives may force a newline.
`-I[nclude] DIR`	Add DIR to the TXL include file search path.
`-s[ize] MM`	Set the TXL transform size to MM megabytes.
`-u[sage]`	Report TXL resource usage statistics at the end of the run.
`-o[utput] FILE`	Write standard output to file FILE.
`-noOPT`	Turn command line option OPT off.
`- USERARGS`	Pass the rest of the command arguments to the TXL program as user command line arguments.

3.2 'txldb' Command Debugging Options

The following TXL debugging options are recognized by *txldb*. For more details see "'txl' Command Debugging Options" above. All output produced by these options is sent to the terminal via the standard error stream.

`-V[ersion]`	Print TXL processor version information on the standard error stream and halt.
`-L[ibrary] DIR`	Use the version of the TXL library stored in directory DIR.
`-Dgrammar`	Print the input language grammar "tree" to the standard error stream.
`-Dparse`	Print the input parse tree to the standard error stream.
`-Dfinal`	Print the output parse tree to the standard error stream.
`-Dpattern`	Print all pattern and replacement parse trees to the standard error stream.
`-Dtrees`	Print parser progress in building pattern and input parse trees as we parse.
`-Dapply`	Print out the actual transformations made by rule applications on the terminal as they happen.
`-Drules`	Print out the names of rules to the standard error stream as they are applied.
`-Dsharing`	Print tree sharing optimization information on the terminal as rules are applied.
`-Dall`	Turn on all of the options *-Dparse -Dapply -Dtrees -Drules -Dfinal* (deadly verbose!).

3.3 TXL Debugger Commands

txldb provides an interactive interface for stepping through a transformation on a rule-by-rule basis. Once the TXL program is compiled and execution begins, the TXL rule debugger is automatically entered and continued execution of the TXL program proceeds under control of a small set of interactive debugging commands. The TXL debugger provides the following commands:

`rules`
: List the names of all of the rules and functions in the TXL program on the terminal.

rule
: Print the name of currently executing rule on the terminal.

`set [RuleName]`
: Set a breakpoint at rule or function *RuleName* (default current rule). TXL will return control to the debugger whenever a breakpoint is encountered.

`clear [RuleName]`
: Clear the breakpoint at rule or function *RuleName* (default current).

`showbps`
: Print a list of all currently set breakpoints on the terminal.

`scope`
: Print the text of the scope of application of the current rule invocation on the terminal. Valid only on entry to a rule, before a pattern match has been found.

`match`
: Print the text of the current pattern match on the terminal. Valid only after pattern match and before replacement in a rule.

`matchcontext`
: Print the text of the current pattern match highlighted in the context of the scope on the terminal. Valid only after pattern match and before replacement in a rule.

`result`
: Print the text of the result of the current construct or rule replacement.

`vars`
: Print a list of the names and types of all currently visible TXL variables.

`VarName` *or* `'VarName`
: Print the text of the current binding of TXL variable *VarName* on the terminal.

`tree VarName`
: Print the tree form of the current binding of TXL variable *VarName* on the terminal.

`where`
: Print the current rule name and execution state on the terminal.

`show [RDname]`
: Print the source code of the rule, function or nonterminal type definition *RDname* (default current rule) on the terminal.

`run`
: Continue execution until the next breakpoint or end of transformation.

`go`
: Same as *run*.

`next`
: Continue execution until the next statement (*construct*, *deconstruct*, *where* or *by* clause) in the current rule or function.

`.`
: Same as *next*.

`/RuleName`
: Continue execution until the next main pattern match of *RuleName* (default current rule), or end of transformation.

`/`
: Continue execution until the next main pattern match of the current rule, or end of transformation.

`//`
: Continue execution until next pattern match (of any rule), or end of transformation.

`step N`
: Step trace execution for *N* (default 1) steps.

`step`
: Step trace execution for one step.

`RETURN`
: Same as *step*.

`help`
: Print a summary of TXL debugger commands on the terminal.

`quit`
: Abort the transformation and exit TXL.

For further details on the TXL debugger and its command set, see the document "Tutorial Introduction to the TXL Rule Debugger".

4. The 'txlprof' Command

```
txlprof 'txlcommand' [ profoptions ]
txlprof profoptions
```

The *txlprof* command provides a convenient interface for profiling the grammar and transformation rules of a TXL program for a given input. The TXL command, which must be quoted, is exactly the command you would normally use to compile and run the program with the desired input, for example:

```
txlprof 'txl -s 100 -d FLUBBER input.stuff program.Txl'
```

txlprof compiles and executes the the TXL program using a special profiling version of the TXL processor, and outputs the result profile to the terminal on the standard error stream. By default, a profile of the rule set is output, sorted by cumulative time spent executing each rule and its subrules. Using the *-parse* command line option, a profile of the grammar in parsing the input is output instead. Options allow sorting of the profile output by most time, space, number of search cycles, or parse efficiency.

When run with no quoted TXL command, *txlprof* re-interprets the results of the previous *txlprof* run as indicated by the command line options. This is done by re-reading the raw profile data stored in the files *txl.pprofout* and *txl.rprofout*, which are created by *txlprof* when it is first run.

4.1 'txlprof' Command Option Flags

The following profiling options are recognized by *txlprof*.

`-parse`	Output a grammar parsing profile rather than the default rule transformation profile.
`-time`	Sort the profile by most cumulative time per rule or nonterminal (the default).
`-space`	Sort the profile by most cumulative space used per rule or nonterminal.
`-calls`	Sort the profile by most calls per rule or nonterminal.
`-cycles`	Sort the profile by most search/match cycles per rule or most parse cycles per nonterminal.
`-eff`	Sort the profile by least parse efficiency per nonterminal.
`-percall`	Show average time, space and cycles per rule invocation or nonterminal call.

5. The 'txlapp' Command

txlapp [*txloptions*] *filename.Txl*

The *txlapp* command provides a convenient interface for compiling a TXL program to a standalone application. The single command argument *filename.Txl* is the TXL program to be compiled. The TXL program file must be named ending in the suffix *.Txl* and must be either in the current working directory, in the *Txl* subdirectory of the current directory, or in the system TXL library (e.g., */usr/local/lib/txl*).

txlapp uses the TXL compiler/interpreter *txl* (see "The 'txl' Command" above) to compile the program to a TXL tree code file, and then runs the TXL tree code converter to produce a C program that is compiled and linked with the TXL transformation engine to produce the output file *filename*.x, a standalone executable program.

All progress and error messages are sent to the terminal via the standard error stream.

5.1 'txlapp' Command Option Flags

The following TXL options are recognized by txlapp. See "The 'txl' Command" above for more detailed explanations. Options specified to *txlapp* become the defaults of the compiled result standalone application. Most can be overridden each time that the application is run by specifying new options on the application's own command line (see "Standalone Application Options" below).

`-q[uiet]` Quiet operation - turn off all progress messages when the application is run. This is the default for standalone applications.

`-v[erbose]` Verbose operation - give detailed TXL progress messages when the application is run.

`-d[efine] SYMBOL`
Define the TXL preprocessor symbol SYMBOL.

`-comment` The application is to treat comments as input items. By default standalone applications discard comments when parsing their input.

`-char[input]`
The application is to treat all input characters (including newlines and spaces) as significant. By default applications treat newlines and spaces as separators only.

`-token[input]`
The application is to treat newlines and spaces as separators only. This is the default for standalone applications.

`-txl` The application is to treat its input as TXL source. This option makes it possible to make applications that transform TXL programs.

`-card` The application is to treat its input as IBM 80-column Cobol card images.

`-cobol` The application is to treat its input as Tandem/Microfocus format Cobol.

`-ibmcobol` The application is to treat its input as IBM mainframe Cobol.

`-tandemcobol`
The application is to treat its input as Tandem Cobol.

`-cobolout` The application is to format its output as IBM mainframe Cobol card images.

`-lscobol` The application is to handle input and output in LS/Cobol format.

`-attr` The application is to include attributes in its transformed output.
By default, attributes (items of type `[attr X]` for some X) are not printed in the output of a standalone application.

`-raw` The application is to output transformed source in raw (unspaced) format.

`-id[chars] 'CCC'`
The application is to treat the characters 'CCC' as valid following characters in [id] tokens.

`-sp[chars] 'CCC'`
The application is to treat the characters 'CCC' as white space.

`-esc[char] 'C'`
The application should use 'C' as the escape character in string and character literals.

`-upper` The application should translate all unquoted input to upper case.

`-lower` The application should translate all unquoted input to lower case.

`-w[idth] NNN`
Set the default maximum output line width of the application to NNN characters.

`-in[dent] NN`
Set the output indentation increment of the application to NN characters (default 4).

`-tabnl` [TAB_NN] directives may force a newline in the application's output (default).

`-I[nclude] DIR`
Add DIR to the TXL include file search path.

`-L[ibrary] DIR`
Use the version of the TXL library stored in directory DIR to compile the application.

`-s[ize] MM` Set the application's default TXL transform size to *MM* megabytes.

`-u[sage]` Report TXL resource usage statistics at the end of every run of the application.

`-noOPT` Turn command line option OPT off.

6. Running TXL Standalone Applications

```
program.x [ txloptions ] inputfile [ - useroptions ]
```

Standalone applications compiled using *txlapp* are run as shown above. Transformed output is sent to the standard output and may be redirected to a file. By default all TXL progress messages are turned off and only error messages are sent to the terminal via the standard error stream. If the *-verbose* option is used, then progress messages are also sent to the terminal via the standard error stream.

6.1 Standalone Application Option Flags

The following TXL options are recognized by applications created using *txlapp*. In all cases, options specified on the application command line override any defaults set when the application was compiled. See "The 'txl' Command" above for detailed explanations of the options.

```
-h[elp]
-v[erbose]
-comment
-attr
-raw
-w[idth] NNN
-s[ize] MM
-u[sage]
- USERARGS
```

Appendix A. TXL Error Messages

The following error messages can be generated by TXL and standalone applications created using it. The list is numerically sorted by error code.

```
* TXL0101E - (TXL implementation limit) Input too large
      (total text of unique tokens > NNN chars)
```

The total length of the input text to the TXL program, counting the text of each different input item only once even if it appears many times, is greater than the TXL implementation can handle at the present size.

Recommended action: Increase TXL implementation limits using the *-size* option.
For technical reasons, if this particular problem is encountered in a compiled standalone TXL application, then the application must be completely recompiled using the larger size.

```
* TXL0102E - (TXL implementation limit) Input too large
      (total number of unique tokens > NNN)
```

The total length of the input to the TXL program, in terms of input items and counting each different input item only once even if it appears many times, is greater than the TXL implementation can handle at the present size.

Recommended action: Increase TXL implementation limits using the *-size* option.
For technical reasons, if this particular problem is encountered in a compiled standalone TXL application, then the application must be completely recompiled using the larger size.

```
* TXL0111E - (TXL implementation limit) Too many defined types (> NNN)
```

The total number of defined nonterminals, including both *define* statements and built-in nonterminals, is larger than the TXL implementation can handle.

Recommended action: This limit is not extensible. Please report this problem as a TXL bug.

```
* TXL0112E - [TYPENAME] has not been defined
```

The indicated nonterminal is used but not defined in the TXL program.

Recommended action: Check the spelling of the name and correct any uses if necessary.
If the nonterminal name is spelled correctly, add an appropriate *define* statement for it.
If the nonterminal name that appears in this message is a TXL internal name (one that begins with 'TXL_'), file a TXL bug report.

```
* TXL0121E - (TXL internal error) Fatal TXL error in backup_tree
* TXL0122E - (TXL internal error) Fatal TXL error in is_empty
```

These errors indicate a catastrophic failure in the TXL compiler/interpreter.

Recommended action: Save a copy of the entire TXL program, its include files, and the input (if any) used for the failing run. Report the problem as a TXL bug, including the copy with the bug report.

```
* TXL0123W define 'TYPENAME' - (Warning) Empty recursion could not be
      resolved with lookahead 'TOKEN' after NNN recursions
      (using pruning heuristic to recover)
```

The nonterminal definition 'TYPENAME' in the TXL program's grammar contains a [repeat X] where [X] derives the empty string, and TXL was unable to resolve an end to the parse while parsing at or near the indicated symbol.

Recommended action: Modify the grammar to avoid repeating nonterminals that derive the empty string, either by changing [repeat X] to [repeat X+] or by changing the definition of [X] such that it does not derive the empty string. Analyze the grammar using the *-analyze* command line option to identify other potential problems.

```
* TXL0126E - Maximum parse depth exceeded
* TXL0127E - Parse time limit (NNN cycles) exceeded
```

These messages indicate that the TXL parser ran out of time or stack space when trying to parse the indicated context.

Recommended action: Check for the possibility of an infinite parse by re-running using the verbose (-v) option. Analyze the grammar for other potential problems using the *-analyze* option. If necessary, modify the grammar to avoid any potentially infinite parses.
If the grammar is sound, then increase TXL limits using the *-size* option.

```
* TXL0128E - (TXL internal error) Fatal TXL error NNN in parse
```

This error indicates a catastrophic failure in the TXL compiler/interpreter.

Recommended action: Save a copy of the entire TXL program, its include files, and the input (if any) used for the failing run. Report the problem as a TXL bug, including the copy with the bug report.

```
* TXL0129E - Unable to create TXL profile file 'txl.pprofout'
```

This error indicates that the TXL profiler was unable to create one of its output files, probably because the user does not have permission to create a file in the current working directory.

Recommended action: Change directory to a directory where you have permission to create files, and rerun the TXL profiler.

```
* TXL0130E - Fatal TXL error in parse (signal)
```

The TXL parser caught an unexpected interrupt while parsing.

Recommended action: Possibly caused by a full virtual memory swap space. Reboot the computer and try the run again. If the problem persists, save a copy of the entire TXL program, its include files, and the input (if any) used for the failing run. Report the problem as a TXL bug, including the copy with the bug report.

```
* TXL0141E - (TXL implementation limit) Input too large
      (total length > NNN tokens)
```

The total length of the input to the TXL program, in terms of individual input items, is longer than the TXL implementation can handle at the present size.

Recommended action: Increase TXL implementation limits using the *-size* option.

```
* TXL0144E - (TXL implementation limit) Input line too long
      (> 32767 characters)
```

An line in the input is longer than the maximum that the TXL implementation can handle.

Recommended action: Check that the input file is a valid text file and reduce line size to a maximum of 32767 characters if so.

```
* TXL0145E - (TXL implementation limit) TXL program line too long
      (> 254 characters)
```

A line in the TXL program source is longer than the maximum that the TXL implementation can handle (254 characters).

Recommended action: Reduce the TXL program source to a maximum line length of 254 characters.

```
* TXL0146E - (TXL implementation limit) Input line too long
      (> 32767 characters)
```

An line in the input is longer than the maximum that the TXL implementation can handle.

Recommended action: Check that the input file is a valid text file and reduce line size to a maximum of 32767 characters if so.

```
* TXL0147E - (TXL implementation limit) TXL program line too long
      (> 254 characters)
```

A line in the TXL program source is longer than the maximum that the TXL implementation can handle (254 characters).

Recommended action: Reduce the TXL program source to a maximum line length of 254 characters.

```
* TXL0148W - (Warning) Obsolete 'TxlExternals' include file ignored
```

The TXL program contains an include statement for the obsolete 'TxlExternals' include file.

Recommended action: 'TxlExternals' is no longer required. Remove the include statement.

```
* TXL0149E - (TXL implementation limit) Too many source include files (> NNN)
```

The total number of source files in the input is larger than the TXL implementation can handle.

Recommended action: Increase TXL implementation limits using the *-size* option.

`* TXL0150E - Unable to find include file 'INCLUDENAME'`

The indicated include file could not be found by TXL in the present working directory, the *Txl* subdirectory of the present working directory, any of the include directories specified using *-I* options, or the system TXL library directory.

Recommended action: Most often this is due to a misspelling of the include file name. Check the spelling of the include file name. If correct, then check to see that the file exists in one of the specified directories. Remember that TXL include file names are case sensitive.

`* TXL0151E - (TXL implementation limit) Include file nesting too deep (> NNN)`

The depth of include files that themselves contain include statements is more than the TXL implementation can handle. A possible cause of this error is an include file with an include statement that includes the file itself or one of the files that included it, forming an infinite chain of included files.

Recommended action: Fix any infinite include chains. If there are none, then work around the problem by hand-editing some included files directly into their including file. Please report this problem as a TXL bug.

`* TXL0152E - Unable to open source file 'FILENAME'`

The TXL program source file could not be found by TXL in the present working directory, the *Txl* subdirectory of the present working directory, any of the include directories specified using *-I* options, or the system TXL library directory.

Recommended action: Most often this is due to a misspelling of the TXL source file name or input file extension. Check the spelling of the source file name and that the file actually exists in one of the specified directories. If the file exists and is spelled correctly, then see the section "The 'txl' Command" of this manual to check that you are using the command correctly. Remember that TXL source file names are case sensitive.

`* TXL0153E - Preprocessor syntax error: missing #endif directive`

No matching *#end if* or *#endif* preprocessor directive was found for a *#if* or *#ifdef* directive.

Recommended action: Check that every *#if* and *#ifdef* directive in the TXL source program is balanced with a matching *#end if* or *#endif*.

`* TXL0154E - (TXL implementation limit) Too many preprocessor symbols (> NNN)`

The total number of defined preprocessor symbols used in the TXL program is larger than the maximum that the TXL implementation can handle.

Recommended action: This limit is not extensible. Reduce the number of defined preprocessor symbols used in the program. Report this problem as a bug.

```
* TXL0155E - (TXL implementation limit) #ifdef nesting too deep
      (> NNN levels deep)
```

The depth of preprocessor *#if* or *#ifdef* nesting used in the TXL program is larger than the maximum that the TXL implementation can handle.

Recommended action: This limit is not extensible. Reduce the number of defined preprocessor symbols used in the program. Report this problem as a bug.

```
* TXL0156E - Preprocessor syntax error: too many #endif directives
      (no matching #if)
```

An extra *#end if* or *#endif* preprocessor directive was found with no matching *#if* or *#ifdef*.

Recommended action: Check that every *#if* and *#ifdef* directive in the TXL source program is balanced with a matching *#end if* or *#endif* and vice-versa.

```
* TXL0157E - Preprocessor syntax error: missing #endif directive
```

No matching *#end if* or *#endif* preprocessor directive was found for a *#if* or *#ifdef* directive.

Recommended action: Check that every *#if* and *#ifdef* directive in the TXL source program is balanced with a matching *#end if* or *#endif*.

```
* TXL0158E - Preprocessor syntax error: missing symbol in #define or
      #undefine directive
```

A *#define*, *#def*, *#undefine* or *#undef* preprocessor directive in the TXL program is malformed.

Recommended action: Check each *#define*, *#def*, *#undefine* and *#undef* in the program and correct any errors.

```
* TXL0159E - Preprocessor syntax error: missing symbol in #if or #elsif
      directive
* TXL0160E - Preprocessor syntax error: missing symbol in #if or #elsif
      directive
```

A *#if, #elsif* or *#elif* preprocessor directive in the TXL program is malformed.

Recommended action: Check each *#if, #elsif* and *#elif* in the program and correct any errors.

```
* TXL0161E - Preprocessor syntax error: #elsif not nested inside #if
* TXL0162E - Preprocessor syntax error: #else not nested inside #if
```

A *#elsif, #elif* or *#else* preprocessor directive in the TXL program was found outside the range of a *#if* or *#ifdef*.

Recommended action: Check each *#elsif, #elif* and *#else* in the program and correct any errors.

* TXL0163E - Preprocessor directive syntax error at or near:

A syntax error was found in the indicated preprocessor directive.

Recommended action: Correct any syntax errors in the preprocessor directive.

* TXL0164E - Syntax error - comment ends at end of file

The input file contains a bracketed comment with no ending bracket.

Recommended action: Check that all bracketed comments are properly ended.

* TXL0165W - (Warning) Token pattern for 'TYPENAME' accepts the null string

The pattern specified for TYPENAME in a *tokens* statement accepts empty as a token. While the program will probably run correctly, this can severely slow down parsing and should be corrected.

Recommended action: Change the token pattern to require at least one character in the token.

* TXL0166E - (TXL implementation limit) Too many token patterns (links) (> NNN)

The total number of token patterns defined in *tokens* statements is larger than the TXL implementation can handle.

Recommended action: This limit is not extensible. Work around the problem by merging multiple patterns for one token kind into a single token pattern. Please report this problem as a TXL bug.

* TXL0167E - Syntax error in token pattern for 'TYPENAME' (\ or # at end of pattern)

In a *tokens* statement, the pattern string for a token definition is ended with one of the metacharacter escape characters '\' or '#'.

Recommended action: If the metacharacter was intended to escape the closing string quote of the pattern, add another quote to end the pattern string. If a literal '\' or '#' was intended, escape the metacharacter itself by preceding it with another '\'.

* TXL0168W - (Warning) Obsolete character pattern '\t' found in token pattern for 'id' : '\u' assumed instead

In a *tokens* statement, a pattern string for [id] contains the character pattern '\t', which has been replaced by '\u'. '\t' now refers to the tab character.

Recommended action: Change the '\t' in the pattern to '\u', unless the tab character was intended.

```
* TXL0169W - (Warning) Escaped character \C in token pattern for 'TYPENAME'
      is not a valid token pattern meta-character
```

The indicated escaped (i.e., proceeded by backslash) character appears in a token pattern but is not a token pattern metacharacter. The backslash is ignored and \C is taken to mean C in the token pattern.

Recommended action: If a literal character C was intended, then remove the backslash. Otherwise replace C with the intended metacharacter.

```
* TXL0170E - Syntax error in token pattern for 'TYPENAME'
      (unbalanced () or [])
* TXL0171E - Syntax error in token pattern for 'TYPENAME'
      (unbalanced () or [])
```

In a *tokens* statement, the pattern string for a token definition contains one or more unbalanced ( ) or [ ] metacharacters. Possibly a literal parenthesis or bracket was intended, in which case the metacharacters should be escaped using '\'.

Recommended action: Check that all ( ) and [ ] metacharacters are properly nested and balanced in the pattern string. If a literal was intended, escape it by preceding it with '\' in the pattern.

```
* TXL0172E - Syntax error - expected 'end WORD1', got 'end WORD2'
```

In a *comments, compounds, keys* or *tokens* statement, the *end* statement does not match the statement kind. Usually this indicates a missing *end* in one or more statements.

Recommended action: Check that each of the statements has a matching *end*.

```
* TXL0173E - (TXL implementation limit) Too many compound literals (> NNN)
```

The total number of compound tokens specified in the *compounds* statements of the TXL program is larger than the TXL implementation can handle.

Recommended action: This limit is not extensible. Reduce the number of compound tokens specified if possible. Work around the problem by removing some of the least used compounds and surrounding them by [SPOFF] and [SPON] in the grammar instead. Please report this problem as a TXL bug.

```
* TXL0174E - (TXL implementation limit) Too many keywords (> NNN)
```

The total number of keywords specified in the *keys* statements of the TXL program is larger than the TXL implementation can handle.

Recommended action: This limit is not extensible. Work around the problem by removing "lesser" keywords from the list. Under normal circumstances this will not affect parsing and transformation except in cases of erroneous input. Please report this problem as a TXL bug.

```
* TXL0175E - (TXL implementation limit) Too many comment conventions (> NNN)
```

The total number of comment brackets specified in the *comments* statements of the TXL program is larger than the TXL implementation can handle.

Recommended action: This limit is not extensible. Reduce the number of comment brackets specified if possible. If your grammar truly needs all the comment brackets you have specified, please report this problem as a TXL bug.

```
* TXL0176E - Syntax error in token pattern definition
      (expected token name, got 'TEXT')
```

In a *tokens* statement, unrecognized TEXT was found in place of a token name. The *tokens* statement must consist of a sequence of token name, token pattern pairs where each token name is a TXL identifier and each token pattern is a string literal.

Recommended action: Check the format of the *tokens* statement.

```
* TXL0177E - Syntax error in token pattern definition
      (expected pattern string, got 'TEXT')
```

In a *tokens* statement, unrecognized TEXT was found in place of a pattern string. Each defined token name must be followed by a string literal giving the pattern for the token.

Recommended action: Check that each defined token name has a string literal pattern.

```
* TXL0178E - (TXL implementation limit) Too many user-defined token patterns
      (> NNN)
```

The total number of token patterns defined in *tokens* statements is larger than the TXL implementation can handle.

Recommended action: This limit is not extensible. Work around the problem by merging multiple patterns for one token kind into a single token pattern. Please report this problem as a TXL bug.

```
* TXL0179E - (TXL implementation limit) Too many user-defined token kinds
      (> NNN)
```

The total number of different user-defined token names defined in *tokens* statements is larger than the TXL implementation can handle.

Recommended action: This limit is not extensible. Work around the problem by merging one or more different token classes into one. Please report this problem as a TXL bug.

```
* TXL0180E - Token pattern for [id] does not allow TXL keywords
```

In a *tokens* statement, the token pattern string give for [id] does not allow recognition of standard TXL identifiers and keywords. In order to allow processing of the TXL program itself, token definitions are constrained to allow recognition of all TXL keywords and special symbols as themselves.

Recommended action: Extend the token pattern for [id] to allow TXL keywords.

```
* TXL0181W - (Warning) Obsolete 'System' include file ignored
```

The TXL program contains an include statement for the obsolete 'System' include file.

Recommended action: 'System' is no longer required. Remove the include statement.

```
* TXL0191E at end of FILENAME - Syntax error near: TEXT
```

TXL was unable to complete a successful parse of the indicated file according to the grammar in the TXL program. The parse succeeded up until the end of the file and could not be resolved at that point.

Recommended action: Check that the indicated file is legal input to the TXL program according to the program's grammar. Most likely any errors of syntax are near the indicated position.

```
* TXL0192E line NNN of FILENAME - Syntax error at or near: TEXT
```

TXL was unable to complete a successful parse of the input according to the grammar of the TXL program. The parse succeeded up until the point indicated by >>> <<< in TEXT and could not be resolved beyond that point.

Recommended action: Check that the indicated file is legal input to the TXL program according to the program's grammar. Most likely any errors of syntax are near the indicated position.

```
* TXL0193E predefined function [FUNCTIONNAME], called from [RULENAME] -
     Syntax error parsing FILENAME as a [TYPENAME], at or near: TEXT
```

The indicated predefined function was unable to complete a successful parse of FILENAME according to the nonterminal definition for [TYPENAME] in the grammar of the TXL program. The parse succeeded up until the point indicated by >>> <<< in TEXT and could not be resolved beyond that point.

Recommended action: Check that the indicated file is legal input to the TXL program according to the program's grammar for [TYPENAME]. Most likely any errors of syntax are near the indicated position.

```
* TXL0194E - (TXL internal error) Fatal TXL error in printPatternToken
```

This error indicates a catastrophic failure in the TXL compiler/interpreter.

Recommended action: Save a copy of the entire TXL program, its include files, and the input (if any) used for the failing run. Report the problem as a TXL bug, including the copy with the bug report.

```
* TXL0195E PARTNAME of RULENAME with goal production [TYPENAME] -
     Syntax error at or near: TEXT
```

TXL was unable to complete a successful parse of the indicated pattern or replacement according to the nonterminal definition for [TYPENAME] in the grammar of the TXL program. The parse succeeded up until the point indicated by >>> <<< in TEXT and could not be resolved beyond that point.

Recommended action: Check that the pattern or replacement legally matches the nonterminal definition in the TXL program's grammar. Most likely any errors of syntax are near the indicated position.

```
* TXL0196E PARTNAME of RULENAME - Parse interrupted at or near: TEXT
```

Compilation of the TXL program was halted by a termination signal (normally ^C typed on the terminal) while parsing the indicated context. If the compile was taking much longer than expected, it is possible that the grammar is circular or infinitely recursive. Use the *-v* option to get more grammar analysis messages, and the *-Dstack* option to get more information about the parse stack.

Recommended action: Check for infinitely recursive nonterminal definitions using the *-v* option. Check the grammar for other potential problems using the *-analyze* option.

```
* TXL0197E line NNN of FILENAME - Parse interrupted at or near: TEXT
```

Parsing of the input to a TXL program was halted by a termination signal (normally ^C typed on the terminal) while parsing the indicated context. If the parse was taking much longer than expected, it is possible that the grammar is circular or infinitely recursive. Use the *-v* option to get more grammar analysis messages, and the *-Dstack* option to get more information about the parse stack.

Recommended action: Check for infinitely recursive nonterminal definitions using the *-v* option. Check the grammar for other potential problems using the *-analyze* option.

```
* TXL0198E PARTNAME of RULENAME - Stack use limit (NNNk) reached,
      at or near: TEXT
```

The TXL parser ran out of stack space while trying to parse the indicated context.

Recommended action: Check for the possibility of an infinite parse by re-running using the verbose (-v) option. Check the grammar for other potential problems using the *-analyze* option. If necessary, modify the grammar to avoid any potentially infinite parses.
If the grammar is sound, then increase TXL stack limit using the *-size* option.

```
* TXL0199E line N of FILENAME - Stack use limit (NNNk) reached,
      at or near: TEXT
```

The TXL parser ran out of stack space while trying to parse the indicated context.

Recommended action: Check for the possibility of an infinite parse by re-running using the verbose (-v) option. Check the grammar for other potential problems using the *-analyze* option. If necessary, modify the grammar to avoid any potentially infinite parses.
If the grammar is sound, then increase TXL stack limit using the *-size* option.

```
* TXL0201W - (Warning) Type name 'TYPENAME' used as a literal identifier
      (use [TYPENAME] or 'TYPENAME instead)
```

A nonterminal type name was used as an unquoted literal identifier in a pattern or replacement in a rule or function.

Recommended action: If the identifier was really intended to be a literal identifier, quote it using a single leading quote '.

```
* TXL0202E - (TXL internal error) Fatal TXL error in processOneKid
```

This error indicates a catastrophic failure in the TXL compiler/interpreter.

Recommended action: Save a copy of the entire TXL program, its include files, and the input (if any) used for the failing run. Report the problem as a TXL bug, including the copy with the bug report.

```
* TXL0203E define 'TYPENAME' - (TXL implementation limit) Too many elements
      or alternatives in one define (maximum NNN)
```

The indicated nonterminal definition has more alternatives or items in one alternative than the TXL implementation can handle in one piece.

Recommended action: Rewrite the indicated nonterminal definition. In the case where there are too many alternatives, break the nonterminal into two alternatives each of which has half of the original alternatives. In the case of too many items in one alternative, factor the alternative using sub-defines to reduce the number of items. For technical reasons critical to TXL efficiency, this limit is not generally extensible in TXL implementations.

```
* TXL0204E define 'TYPENAME' - 'list_', 'repeat_', 'opt_', 'attr_', 'lit_'
      and 'TXL_' name prefixes are reserved for TXL internal use
```

Nonterminal definition TYPENAME is named using one of the indicated prefixes. Names beginning with these prefixes are reserved by TXL for its own internal use.

Recommended action: Change the name of the nonterminal to something that does not begin with any of the reserved prefixes.

```
* TXL0205E define 'TYPENAME' - Define overrides token definition for
      [TYPENAME]
```

Nonterminal definition TYPENAME has the same name as a token definition.

Recommended action: Change the name of the nonterminal to something that does not clash with the token name.

```
* TXL0206W define 'TYPENAME' - (Warning) Define overrides previous declaration
      ('redefine' should be used if override intended)
```

A second *define* statement for the nonterminal TYPENAME was encountered while compiling the TXL program. TXL has assumed that the new definition is intended to replace the old in the grammar.

Recommended action: If the new definition for TYPENAME was intended to override the old, use *redefine* instead. Otherwise, choose a different name for the new nonterminal.

```
* TXL0207E define/redefine 'TYPENAME' - Extended define 'TYPENAME' has not
      been previously defined
```

A *redefine* or *define* statement for the nonterminal TYPENAME uses the '...' extension notation, but no previous definition for the nonterminal exists.

Recommended action: If the define/redefine was intended to extend a later definition, reorder the grammar such that the extended definition appears first. If a literal '...' was intended in the definition, quote it using a leading single quote.

```
* TXL0208E define 'TYPENAME' - Empty defines not allowed - use [empty] instead
```

The nonterminal definition for TYPENAME is implicitly empty (i.e, has no items in it). Nonterminals intended to derive nothing must be explicitly specified using [empty].

Recommended action: Use [empty].

```
* TXL0209E define 'TYPENAME' - Empty alternatives not allowed -
      use [empty] instead
```

The nonterminal definition for TYPENAME has an alternative which is implicitly empty (i.e, has no items in it). Empty alternatives must be explicitly specified using [empty]. This error can be due to an attempt to specify an alternative beginning with '|' as a literal terminal symbol. If '|' is intended to be a terminal symbol in the grammar, then it must be quoted using a leading single quote to distinguish it from the TXL alternative symbol.

Recommended action: If an empty alternative was intended, use [empty] to specify it explicitly. If a literal '|' was intended, then quote it using a leading single quote. Otherwise remove the extra '|' from the definition.

```
* TXL0210E define 'TYPENAME' - (TXL implementation limit) Too many elements
      or alternatives in one define (maximum NNN)
```

The indicated nonterminal definition has more alternatives or items in one alternative than the TXL implementation can handle in one piece.

Recommended action: Rewrite the indicated nonterminal definition. In the case where there are too many alternatives, break the nonterminal into two alternatives each of which has half of the original alternatives. In the case of too many items in one alternative, factor the alternative using sub-defines to reduce the number of items. For technical reasons critical to TXL efficiency, this limit is not generally extensible in TXL implementations.

```
* TXL0211E define 'TYPENAME' - Empty alternatives not allowed -
      use [empty] instead
```

The nonterminal definition for TYPENAME has an alternative which is implicitly empty (i.e, has no items in it). Empty alternatives must be explicitly specified using [empty]. This error can be due to an attempt to specify an alternative beginning with '|' as a literal terminal symbol. If '|' is intended to be a terminal symbol in the grammar, then it must be quoted using a leading single quote to distinguish it from the TXL alternative symbol.

Recommended action: If an empty alternative was intended, use [empty] to specify it explicitly. If a literal '|' was intended, then quote it using a leading single quote. Otherwise remove the extra '|' from the definition.

```
* TXL0212E - Defines cannot be both pre- and post-extended in the
      same definition (split into two redefines)
```

A *redefine* or *define* statement for the nonterminal TYPENAME uses the '...' extension notation as both the first and the last alternative. Nonterminal extensions can be made either to the beginning or the end of the alternatives in the original definition, but not to both in the same extension.

Recommended action: Split the extended definition into two cascaded definitions.

```
* TXL0213I define 'TYPENAME' - (Information) Optimized left recursive define
```

The TXL compiler has found an opportunity to use left-factoring to increase parse efficiency.

Recommended action: None.

```
* TXL0214E define 'TYPENAME' - Definition is circular
```

The definition of TYPENAME is directly circular (i.e., consists entirely of or has an alternative consisting entirely of exactly itself).

Recommended action: Remove the direct circular reference.

```
* TXL0215W - (Warning) [TYPENAME1] is combinatorially ambiguous when parsing
      sequences of [TYPENAME2]
* TXL0216W - (Warning) [TYPENAME1] is combinatorially ambiguous when parsing
      sequences of [TYPENAME2]
```

Nonterminal TYPENAME1 derives sequences of sequences of [TYPENAME2]. This allows for a combinatorial number of different parses, and can lead to slow parsing and syntax error detection.

Recommended action: Modify the grammar to make it clear at which level such sequences are to be handled.

```
* TXL0217W - (Warning) [TYPENAME1] is locally ambiguous when parsing
      sequences of [TYPENAME2]
```

Nonterminal TYPENAME1 derives leading sequences of leading sequences of [TYPENAME2]. This allows for a combinatorial number of different parses, and can lead to slow parsing and syntax error detection.

Recommended action: Modify the grammar to make it clear at which level such sequences are to be handled.

```
* TXL0218W - [TYPENAME] has not been defined
```

The indicated nonterminal is used but not defined in the TXL program.

Recommended action: Check the spelling of the name and correct any uses if necessary. If the nonterminal name is spelled correctly, add an appropriate *define* statement for it. If the nonterminal name that appears in this message is a TXL internal name (one that begins with 'TXL_'), file a TXL bug report.

```
* TXL0219E - [program] has not been defined
```

The required goal nonterminal [program] has not been defined in the TXL program. The goal symbol is the nonterminal as which the entire input to the TXL program is to be parsed.

Recommended action: Add a *define* statement for the nonterminal [program].

```
* TXL0220W - (Warning) Obsolete define [root] used in place of [program]
```

The obsolete goal nonterminal [root] has been defined instead of [program] in the TXL program.

Recommended action: Change the name of the nonterminal definition for 'root' to 'program'.

```
* TXL0221W - (Warning) Obsolete define [root] overrides definition for
      [program]
```

The obsolete goal nonterminal [root] has been defined as well as [program] in the TXL program. TXL has assumed that the definition for [root] is the intended goal nonterminal.

Recommended action: Change the name of the nonterminal definition for 'root' to 'program', and change the name of the nonterminal [program] to some other name if it is used, otherwise remove the definition for it.

```
* TXL0222I - (Information) Analyzing the object language grammar
      (this may take a while)
```

The TXL compiler has begun analyzing the TXL program's grammar definition in detail as requested by the *-analyze* option. This can take a significant amount of time, up to several minutes depending on the grammar.

Recommended action: None.

```
* TXL0301W PARTNAME of 'RULENAME' - (Warning) Scope of function 'FUNCTIONNAME'
      is not of target type
```

In the indicated context, the non-searching function FUNCTIONNAME is applied to a TXL variable of a type different from the pattern type of the function. Because they match their pattern only to the entire scope, it does not make sense for non-searching TXL functions to be applied to scopes of types other than the function's pattern type.

Recommended action: If the function was intended to search, use 'replace *' to make it a searching function, or make it a rule. If the function was not intended to search, check that the type of the scope of the function application is exactly the same as the function's pattern type.

```
* TXL0302E PARTNAME of 'RULENAME1' - 'replace' rule 'RULENAME2' used as
      'where' condition
```

The rule/function RULENAME2 is declared as a transformation rule (one that uses the keyword *replace*) but is used as a condition rule in the indicated context. Condition rules (those used to test conditions in a *where* clause, and declared using the *match* keyword) are constrained to do pattern matching only - they must not construct a replacement.

Recommended action: If the rule is intended to be a condition rule only, change the *replace* keyword to *match* in the rule declaration and remove any *by* clause. If it is intended to be used as both a transformation rule and a condition, then annotate uses of it as a condition in *where* clauses using '?'. If it is intended to be a transformation rule only, then replace any uses of it in *where* clauses with an appropriate condition rule.

```
* TXL0303E PARTNAME of 'RULENAME1' - Number of parameters to rule/function
      'RULENAME2' differs from definition or previous call
```

Rule/function RULENAME2 is declared or previously used with a different number of parameters than it is used with in the indicated context.

Recommended action: Check that the number of formal parameters in the rule declaration agrees with the number of actual parameters specified in each call to the rule.

```
* TXL0304E PARTNAME of 'RULENAME1' - Rule/function 'RULENAME2' used with empty
      or double 'each'
```

The indicated rule/function application uses the keyword *each* incorrectly in its parameter list. *Each* can only be used when followed by at least one actual parameter that is of type [list X] or [repeat X] for some [X].

Recommended action: Check that at least one parameter of the appropriate kind follows *each*.

```
* TXL0305E PARTNAME of 'RULENAME1' - 'each' argument of rule/function
      'RULENAME2' is not a list or repeat
```

A parameter following *each* in a call to the rule/function RULENAME2 is not of type [list X] or [repeat X} for some [X]. *Each* is defined only for actual parameters that are sequences or lists, and applies to all parameters following *each* in the parameter list.

Recommended action: Check that the type of all parameters following *each* are sequences or lists.

```
* TXL0306E PARTNAME of 'RULENAME1' - Type of actual parameter 'VARNAME' of
      rule/function 'RULENAME2' does not agree with definition or previous
      call
* TXL0307W PARTNAME of 'RULENAME1' - (Warning) Literal actual parameter
      'VARNAME' of rule/function 'RULENAME2' is not quoted
```

The function or rule RULENAME2 is called in the indicated context with an identifier actual parameter that is not a TXL variable, and is therefore assumed to be intended to be a literal identifier. Often this indicates that a TXL variable name has been misspelled.

Recommended action: If the actual parameter was intended to be a literal identifier then quote it using a preceding single quote. Otherwise correct the spelling of the intended TXL variable name.

```
* TXL0308E PARTNAME of 'RULENAME1' - Type of actual parameter 'VARNAME' of
      rule/function 'RULENAME2' does not agree with definition or previous
      call
```

The type of the indicated actual parameter of rule/function RULENAME2 is not the same as the type of the corresponding parameter in the rule declaration or a previous call.

Recommended action: Check that the type of the formal parameter in the rule declaration agrees with the type of the corresponding actual parameter specified in each call to the rule.

```
* TXL0309E PARTNAME of 'RULENAME1' - Number of parameters passed to
      rule/function 'RULENAME2' differs from definition or previous call
* TXL0310E PARTNAME of 'RULENAME1' - Number of parameters passed to
      rule/function 'RULENAME2' differs from definition or previous call
```

Rule/function RULENAME2 is declared or previously used with a different number of parameters than it is used with in the indicated context.

Recommended action: Check that the number of formal parameters in the rule declaration agrees with the number of actual parameters specified in each call to the rule.

```
* TXL0311E PARTNAME of 'RULENAME1'- (TXL implementation limit) Too many
      parameters passed to rule/function 'RULENAME2' (> NNN)
```

The indicated rule call implies that RULENAME2 has more parameters than the TXL implementation can handle.

Recommended action: Reprogram RULENAME2 to reduce the number of parameters by passing higher-level parameters and using deconstructs to access the original lower-level parts. This limit is not generally extensible in TXL implementations.

```
* TXL0312E PARTNAME of 'RULENAME1' - Rule/function 'RULENAME2' used with empty
      or double 'each'
```

The indicated rule/function application uses the keyword *each* incorrectly in its parameter list. *Each* can only be used when followed by at least one actual parameter that is of type [list X] or [repeat X] for some [X].

Recommended action: Check that at least one parameter of the appropriate kind follows *each*.

```
* TXL0313E PARTNAME of 'RULENAME1' - 'each' argument of rule/function
      'RULENAME2' is not a list or repeat
```

A parameter following *each* in a call to the rule/function RULENAME2 is not of type [list X] or [repeat X} for some [X]. *Each* is defined only for actual parameters that are sequences or lists, and applies to all parameters following *each* in the parameter list.

Recommended action: Check that the type of all parameters following *each* are sequences or lists.

```
* TXL0314W PARTNAME of 'RULENAME1' - (Warning) Literal actual parameter
      'VARNAME' of rule/function 'RULENAME2' is not quoted
```

The function or rule RULENAME2 is called in the indicated context with an identifier actual parameter that is not a TXL variable, and is therefore assumed to be intended to be a literal identifier. Often this indicates that a TXL variable name has been misspelled.

Recommended action: If the actual parameter was intended to be a literal identifier then quote it using a preceding single quote. Otherwise correct the spelling of the intended TXL variable name.

```
* TXL0315W - (Warning) Type name 'TYPENAME' used as a literal identifier
      (use [TYPENAME] or 'TYPENAME instead)
```

A nonterminal type name was used as an unquoted literal identifier in a pattern or replacement in a rule or function.

Recommended action: If the identifier was really intended to be a literal identifier, quote it using a single leading quote '.

```
* TXL0316W - (Warning) Variable name 'VARNAME' is quoted as a literal
      identifier (possibly by mistake?)
```

The TXL variable VARNAME was used as a quoted literal identifier in a pattern or replacement in a rule or function.

Recommended action: If the identifier was intended to refer to the variable, remove the leading single quote. If the identifier was really intended to be a literal identifier, then it is recommended that the name of the variable be changed to avoid any possible confusion on the part of the maintainer.

```
* TXL0317W - (Warning) Type name 'TYPENAME' used as a literal identifier
      (use [TYPENAME] or 'TYPENAME instead)
```

A nonterminal type name was used as an unquoted literal identifier in a pattern or replacement in a rule or function.

Recommended action: If the identifier was really intended to be a literal identifier, quote it using a single leading quote '.

```
* TXL0318E - (TXL internal error) Fatal TXL error in checkVarOrExp
```

This error indicates a catastrophic failure in the TXL compiler/interpreter.

Recommended action: Save a copy of the entire TXL program, its include files, and the input (if any) used for the failing run. Report the problem as a TXL bug, including the copy with the bug report.

```
* TXL0319E PARTNAME of 'RULENAME' - (TXL implementation limit) Pattern or
      replacement too large (> NNN tokens)
```

The indicated pattern or replacement contains more items than the TXL implementation can handle in one piece.

Recommended action: This limit is not extensible. Factor the pattern or replacement into parts using TXL variables, constructs and deconstructs. If this is not practical or the problem persists, report the problem as a TXL bug.

```
* TXL0320E PARTNAME of 'RULENAME' - Constructed variable has already been
      defined
```

The indicated *construct* statement binds a variable that has already been bound in the rule. TXL does not permit reassignment of local variables.

Recommended action: Change the name of the constructed variable to a new name.

```
* TXL0321E PARTNAME of 'RULENAME' - Type required for exported variable
```

The indicated *export* statement exports a variable that has not yet been bound in the rule. A nonterminal type must be given for the variable.

Recommended action: Add a nonterminal type for the variable to the *export* statement.

```
* TXL0322E PARTNAME of 'RULENAME' - Anonymous construct cannot be of type
      [TYPENAME]
```

The indicated anonymous *construct* statement (i.e., construct using '_' as the replacement) is not of an appropriate type. Anonymous constructs are defined only for the types [id], [stringlit], [charlit], [number], [opt X], [repeat X] and [list X] for some type [X].

Recommended action: Check that the type of the anonymous construct is one of the allowed set. If not, insert a new *construct* statement for a new variable with an explicit instance of the type preceding the anonymous construct, and replace the '_' in the anonymous construct with the new variable name.

```
* TXL0323I PARTNAME of 'RULENAME' - (Information) Optimized skipping
      deconstruct of repeat
```

The TXL compiler has found an opportunity to use iterative searching to increase pattern search efficiency.

Recommended action: None.

```
* TXL0324E PARTNAME of 'RULENAME' - 'where' condition requires a rule call
```

The indicated *where* statement does not invoke a condition rule or function to implement the test.

Recommended action: Add a condition rule or function call following the variable name in the *where*.

```
* TXL0325E PARTNAME of 'RULENAME' - Imported variable has already been defined
```

The indicated *import* statement imports a variable that has already been bound in the rule. TXL does not permit reassignment of local variables.

Recommended action: Change the name of the previously bound variable to a new name.

```
* TXL0326E PARTNAME of 'RULENAME' - Type required for imported variable
```

The indicated *import* statement imports a variable that has not been previously imported in the rule. A nonterminal type must be given for the variable.

Recommended action: Add a nonterminal type for the variable to the *import* statement.

```
* TXL0327W PARTNAME of 'RULENAME' - (Warning) Imported variable already has a
      type (import type ignored)
```

The indicated *import* statement imports a variable that has already been previously imported in the rule, and hence already has a nonterminal type.

Recommended action: Remove the nonterminal type from the *import* statement.

```
* TXL0328E PARTNAME of 'RULENAME' - Type required for exported variable
```

The indicated *export* statement exports a variable that has not yet been bound in the rule. A nonterminal type must be given for the variable.

Recommended action: Add a nonterminal type for the variable to the *export* statement.

```
* TXL0329W PARTNAME of 'RULENAME' - (Warning) Exported variable already has a
      type (export type ignored)
```

The indicated *export* statement exports a variable that has already been bound in the rule, and hence already has a nonterminal type.

Recommended action: Remove the nonterminal type from the *export* statement.

```
* TXL0330E PARTNAME of 'RULENAME' - Exported variable has not been bound
      (export value required)
```

The indicated *export* statement binds a new variable but does not give a replacement for the variable's value.

Recommended action: Add a replacement for the variable to the *export* statement. Empty replacements are not permitted for *export* statements. If the intended value is empty, use a *construct* statement to bind the variable to an empty value, then export the constructed variable.

```
* TXL0331E rule/function 'RULENAME' - Number of formal parameters does not
      agree with previous call
```

The indicated rule/function is declared with a different number of parameters than it has been used with.

Recommended action: Check that the number of formal parameters in the rule declaration agrees with the number of actual parameters specified in each call to the rule.

* `TXL0332E rule/function 'RULENAME' - Type of formal parameter 'VARNAME' does not agree with previous call`

The indicated formal parameter is of a type different from the actual parameters passed to it in rule calls.

Recommended action: Check that the formal parameter and the actual parameters passed to it are all of the same nonterminal type.

* `TXL0333E rule/function 'RULENAME' - Number of formal parameters does not agree with previous call`

The indicated rule/function is declared with a different number of parameters than it has been used with.

Recommended action: Check that the number of formal parameters in the rule declaration agrees with the number of actual parameters specified in each call to the rule.

* `TXL0334E rule/function 'RULENAME' - (TXL implementation limit) Too many formal parameters (> NNN)`

The indicated rule has more formal parameters than the TXL implementation can handle.

Recommended action: Reprogram the indicated rule to reduce the number of formal parameters by passing higher-level parameters and using deconstructs to access the original lower-level parts. This limit is not generally extensible in TXL implementations.

* `TXL0335E rule/function 'RULENAME' - (TXL implementation limit) Rule/function has too many constructs, deconstructs and conditions (> NNN)`

The indicated rule has more *construct*, *deconstruct*, *import*, *export* and *where* statements than the TXL implementation can handle.

Recommended action: Reprogram the indicated rule to reduce the number of constructs, deconstructs and conditions either by amalgamating or by using additional subrules. For technical reasons critical to TXL efficiency, this limit is not generally extensible in TXL implementations.

* `TXL0336I PARTNAME of 'RULENAME' - (Information) Potential optimized skipping match/replace`

The TXL compiler has found an opportunity to use iterative searching to increase pattern search efficiency.

Recommended action: None.

```
* TXL0337E rule/function 'RULENAME' - (TXL implementation limit) Rule/function
      has too many constructs, deconstructs and conditions (> NNN)
```

The indicated rule has more *construct*, *deconstruct*, *import*, *export* and *where* statements than the TXL implementation can handle.

Recommended action: Reprogram the indicated rule to reduce the number of constructs, deconstructs and conditions either by amalgamating or by using additional subrules. For technical reasons critical to TXL efficiency, this limit is not generally extensible in TXL implementations.

```
* TXL0338E rule/function 'RULENAME' - 'replace' rule/function must have a
      replacement
```

The indicated rule/function is declared as a transformation rule (one that uses the keyword *replace*) but has no *by* clause. Transformation rules must construct a replacement.

Recommended action: If the rule is intended to be a transformation rule, add a *by* clause for the result. If the rule is intended to be a condition rule, change the *replace* keyword to *match* in the rule declaration.

```
* TXL0339E rule/function 'RULENAME' - 'match' rule/function cannot have a
      replacement
```

The indicated condition rule/function contains a *by* clause. Condition rules, indicated by the *match* keyword, are constrained to do pattern matching only - they must not construct a replacement.

Recommended action: If the rule is intended to be a transformation rule, change the *match* keyword to *replace* in the rule declaration. If it is also intended to be used as a condition, annotate uses of it as a condition in *where* clauses using '?'. If it is intended to be a condition rule only, then remove the *by* clause.

```
* TXL0340E rule/function 'RULENAME' - 'list_', 'repeat_', 'opt_', 'attr_',
      'lit_' and 'TXL_' name prefixes are reserved for TXL internal use
```

Rule/function RULENAME is named using one of the indicated prefixes. Names beginning with these prefixes are reserved by TXL for its internal use.

Recommended action: Change the name of the rule/function to something that does not begin with any of the reserved prefixes.

```
* TXL0341W rule/function 'RULENAME' - (Warning) Rule/function declaration
      overrides predefined function
```

Rule/function RULENAME is named using one of the TXL predefined function names, so the predefined function will not be available in this program.

Recommended action: Change the name of the rule/function to another name, unless the intention is to replace the predefined function for this program.

```
* TXL0342E rule/function 'RULENAME' - Rule/function has been previously
      defined
```

A rule/function with the indicated name has been previously declared.

Recommended action: Change the name of one or the other of the rules involved and check all calls to see that the intended one of the two is called in each case.

```
* TXL0343E rule/function 'RULENAME' - 'replace' rule/function has been
      previously used as a 'where' condition
```

The indicated rule/function is declared as a transformation rule (one that uses the keyword *replace*) but is used as a condition rule. Condition rules (those used to test conditions in a *where* clause, and declared using the *match* keyword) are constrained to do pattern matching only - they must not construct a replacement.

Recommended action: If the rule is intended to be a condition rule only, change the *replace* keyword to *match* in the rule declaration and remove any *by* clause. If it is intended to be used as both a transformation rule and a condition, then annotate uses of it as a condition in *where* clauses using '?'. If it is intended to be a transformation rule only, then replace any uses of it in *where* clauses with an appropriate condition rule.

```
* TXL0344W rule/function 'RULENAME' - (Warning) 'replace' rule has target type
      [any] (results may not be well-formed)
```

The indicated rule/function is a *replace* rule with target type [any]. Such rules are potentially dangerous since they may invalidate the grammatical structure of the scope.

Recommended action: None. This warning is a reminder of the inherently dangerous nature of this feature.

```
* TXL0345W rule/function 'RULENAME' - (Warning) Rule/function declaration
      overrides predefined function
```

Rule/function RULENAME is named using one of the TXL predefined function names, so the predefined function will not be available in this program.

Recommended action: Change the name of the rule/function to another name, unless the intention is to replace the predefined function for this program.

```
* TXL0346E rule/function 'RULENAME' - Rule/function has been previously
      defined
```

A rule/function with the indicated name has been previously declared.

Recommended action: Change the name of one or the other of the rules involved and check all calls to see that the intended one of the two is called in each case.

```
* TXL0347E rule/function 'RULENAME' - 'replace' rule/function has been
      previously used as a 'where' condition
```

The indicated rule/function is declared as a transformation rule (one that uses the keyword *replace*) but is used as a condition rule. Condition rules (those used to test conditions in a *where* clause, and declared using the *match* keyword) are constrained to do pattern matching only - they must not construct a replacement.

Recommended action: If the rule is intended to be a condition rule only, change the *replace* keyword to *match* in the rule declaration and remove any *by* clause. If it is intended to be used as both a transformation rule and a condition, then annotate uses of it as a condition in *where* clauses using '?'. If it is intended to be a transformation rule only, then replace any uses of it in *where* clauses with an appropriate condition rule.

```
* TXL0348W rule/function 'RULENAME' - (Warning) Target type of function does
      not match scope of previous calls
```

The non-searching function RULENAME is declared with a pattern type different from one or more variables it is applied to in the program. Because they match their pattern only to the entire scope, it does not make sense for non-searching TXL functions to be applied to scopes of types other than the function's pattern type.

Recommended action: If the function was intended to search, use 'replace *' to make it a searching function, or make it a rule. If the function was not intended to search, check each application of the function in the program to see that the variable it is applied to is of the same type as the function's pattern type.

```
* TXL0349W rule/function 'RULENAME' - (Warning) 'replace' function has target
      type [any] (results may not be well-formed)
```

The indicated rule/function is a *replace* rule with target type [any]. Such rules are potentially dangerous since they may invalidate the grammatical structure of the scope.

Recommended action: None. This warning is a reminder of the inherently dangerous nature of this feature.

```
* TXL0350W rule/function 'RULENAME' - (Warning) Obsolete external declaration
      of predefined function ignored
```

The TXL program contains an external declaration for the predefined function RULENAME. External declarations of predefined functions are no longer required.

Recommended action: Remove the external declaration.

```
* TXL0351E rule/function 'RULENAME' - Rule/function has been previously
      defined
```

A rule/function with the indicated name has been previously declared.

Recommended action: Change the name of one or the other of the rules involved and check all calls to see that the intended one of the two is called in each case.

```
* TXL0352E - [?] is not defined on predefined function [RULENAME]
```

The *replace* predefined function RULENAME is used as a condition in a *where* clause using [?RULENAME]. *Replace* predefined functions can not be used as condition functions.

Recommended action: Write an appropriate condition function to test the desired condition.

```
* TXL0353E - Rule/function 'RULENAME' has not been defined
```

The indicated rule/function is used in the TXL program but never declared.

Recommended action: Check calls to the rule for correct spelling of the rule name. If the rule name is spelled correctly, add a declaration for the missing rule.

```
* TXL0354E PARTNAME of 'RULENAME' - Type of imported/exported variable
      'VARNAME' does not match global variable
* TXL0355E PARTNAME of 'RULENAME' - Type of imported/exported variable
      'VARNAME' does not match global variable
```

The nonterminal type given for a global variable in the indicated *import* or *export* statement is not the same as the type it is given in other rules/functions. Global variables must have the same nonterminal type throughout the program.

Recommended action: Check that all *import*s and *export*s of the global variable in the program have the same nonterminal type, and correct any that differ.

```
* TXL0356W PARTNAME of 'RULENAME1' - (Warning) Scope 'VARNAME [TYPENAME]' of
      call to rule/function 'RULENAME2' can never contain a match
```

The indicated variable has a nonterminal type that can never contain the target type of the main pattern of the rule/function RULENAME2. This may not be an error if the scope has been constructed using [any] rules/functions, or if the RULENAME2 is being called only for global side effects of its pre-pattern statements.

Recommended action: Check that the type of the variable and the rule/function RULENAME2 are as intended, and correct any errors.

```
* TXL0357W PARTNAME of 'RULENAME1' - (Warning) Call to rule/function
      'RULENAME2' with scope 'VARNAME [repeat TYPENAME1+]' may yield an empty
      result for embedded [repeat TYPENAME2]
```

Although the variable VARNAME is of nonterminal type [repeat TYPENAME1+], indicating that it should not be empty, it can still yield an empty result in the given context when acted on by rule/function RULENAME2, which can replace it with an empty [repeat TYPENAME2].

Recommended action: Check whether this behaviour was intended, and reprogram the grammar to make the empty choice explicit if so.

```
* TXL0358I rule/function 'RULENAME' - (Information) Rejected optimized
      skipping match/replace
```

The TXL compiler has abandoned an opportunity to use iterative searching to increase pattern search efficiency due to unresolvable rule complexity.

Recommended action: None.

```
* TXL0359E - (TXL internal error) Fatal TXL error in checkRuleCallScopes
* TXL0360E - (TXL internal error) Fatal TXL error in callsRule
```

These errors indicate a catastrophic failure in the TXL compiler/interpreter.

Recommended action: Save a copy of the entire TXL program, its include files, and the input (if any) used for the failing run. Report the problem as a TXL bug, including the copy with the bug report.

```
* TXL0361E - Rule/function 'main' has not been defined
```

The TXL program does not declare a rule/function named *main*. Execution of a TXL program consists of applying the main rule to the entire input parse tree. Thus a main rule is required to execute the program.

Recommended action: Write a main rule or rename an existing rule to be the main rule.

```
* TXL0362W - (Warning) Obsolete rule/function name 'mainRule' used in place of
      'main'
```

The obsolete main rule *mainRule* has been declared instead of *main* in the TXL program.

Recommended action: Change the name of the rule/function *mainRule* to *main*.

```
* TXL0401E - Not enough memory to compile TXL program
      (a larger size is required for this program)
```

When compiling using *txlapp* or *txl -compile*, a garbage collection was required to compile the TXL program due to a shortage of tree memory. For technical reasons, compiled TXL programs cannot be stored if a garbage collection has occurred.

Recommended action: Recompile with increased tree memory using the *-size* command line option.

```
* TXL0402E - Load file is not a compiled TXL object file
```

This error indicates that, when using the *-load* option, the compiled.*CTxl* object file being loaded was either not in *.CTxl* format, or was in a format used by a different version of TXL.

Recommended action: Recompile the *.CTxl* file using the *-compile* option.
If the problem persists, report the problem as a TXL bug.

* `TXL0403E - (TXL internal error) TXL size does not match compiled size`

    This error indicates that, when using the *-load* option, the compiled.*CTxl* object file being loaded was found to be corrupted, or to be in a format used by another version of TXL.

    Recommended action: Recompile the *.CTxl* file using the *-compile* option.
    If the problem persists, report the problem as a TXL bug.

* `TXL0404E - TXL object file was compiled by a different version of TXL`

    This error indicates that, when using the *-load* option, the compiled.*CTxl* object file being loaded was compiled by a different version of TXL.

    Recommended action: Recompile the *.CTxl* file using the *-compile* option.
    If the problem persists, report the problem as a TXL bug.

* `TXL0405E - (TXL internal error) Synchronization error NNN on compiled file`

    This error indicates that, when using the *-load* option, either the compiled.*CTxl* object file being loaded was found to be corrupted, or the TXL loader has encountered a catastrophic failure.

    Recommended action: Recompile the *.CTxl* file using the *-compile* option.
    If the problem persists, report the problem as a TXL bug.

* `TXL0406E - Load file is not a compiled TXL object file`

    This error indicates that, when using the *-load* option, the compiled.*CTxl* object file being loaded was either not in *.CTxl* format, or was in a format used by a different version of TXL.

    Recommended action: Recompile the *.CTxl* file using the *-compile* option.
    If the problem persists, report the problem as a TXL bug.

* `TXL0407E - (TXL internal error) TXL size does not match compiled size`

    This error indicates that, when using the *-load* option, the compiled.*CTxl* object file being loaded was found to be corrupted, or to be in a format used by another version of TXL.

    Recommended action: Recompile the *.CTxl* file using the *-compile* option.
    If the problem persists, report the problem as a TXL bug.

* `TXL0408E - TXL object file was compiled by a different version of TXL`

    This error indicates that, when using the *-load* option, the compiled.*CTxl* object file being loaded was compiled by a different version of TXL.

    Recommended action: Recompile the *.CTxl* file using the *-compile* option.
    If the problem persists, report the problem as a TXL bug.

```
* TXL0501E rule/function 'RULENAME' - Maximum search depth (NNN) exceeded when
      searching for pattern match (a larger size is required for this
      transform)
```

The depth of the tree being searched for the indicated pattern is more than the TXL implementation can handle. A larger transform size is required to complete this transformation.

Recommended action: Increase TXL limits using the *-size* option.

```
* TXL0502E rule/function 'RULENAME1' - (TXL implementation limit) Maximum call
      depth (NNN) exceeded when calling rule/function 'RULENAME2'
      (Probable cause: infinite recursion)
```

The depth of rule calls is more than the TXL implementation can handle. Either the transform size is too small for the transformation, or the indicated rules are involved in a nonterminating transform.

Recommended action: Increase TXL limits using the *-size* option. If a nonterminating transform is suspected, check the indicated rules and the rules they call for termination. If necessary, add appropriate termination conditions.

```
* TXL0503E rule/function 'RULENAME' - Imported global variable 'VARNAME' has
      not been bound
```

The indicated *import* statement imports a global variable that has not yet be exported from a rule/function.

Recommended action: Check that the order of rule/function invocation is such that the global variable is exported before it is imported. If the import really was intended to be first, add an initial *export* for the variable to the main rule.

```
* TXL0504E rule/function 'RULENAME' - Assertion on 'VARNAME' failed
```

The indicated *assert* condition failed. Assertions are required to succeed in all circumstances.

Recommended action: If the *assert* was intended to be a guard rather than an assertion, then change it to be a *where*. Otherwise, use the TXL debugger or add debugging output to investigate why the assertion has failed.

```
* TXL0505E - (TXL implementation limit) 'each' limited to at most two
      parameters
```

A rule call using *each* has more than two actual parameters following the *each*. In this implementation of TXL, the number of parallel each'ed parameters is limited to two.

Recommended action: Use a recursive function to parallel deconstruct each of the three parameters into first element, rest of elements on each recursion instead.

```
* TXL0506E - (TXL internal error) Fatal TXL error in
      resolveReplacementExpressions
```

This error indicates a catastrophic failure in the TXL compiler/interpreter.

Recommended action: Save a copy of the entire TXL program, its include files, and the input (if any) used for the failing run. Report the problem as a TXL bug, including the copy with the bug report.

```
* TXL0507E - Repeated space failure in same replacement
      (a larger size is required for this transform)
```

The TXL garbage collector was unable to recover enough tree space to successfully continue the transformation.

Recommended action: Increase TXL limits using the *-size* option.

```
* TXL0508E - (TXL internal error) Fatal TXL error in processParts
```

This error indicates a catastrophic failure in the TXL compiler/interpreter.

Recommended action: Save a copy of the entire TXL program, its include files, and the input (if any) used for the failing run. Report the problem as a TXL bug, including the copy with the bug report.

```
* TXL0509E rule/function 'RULENAME' - One pass rule failed to terminate
      (probable cause: part of replacement matches pattern)
```

The available stack space was exhausted while running the transformation. In a one-pass rule, this can only mean that the rule will never terminate since the replacement is directly matching the rule's pattern.

Recommended action: Modify the pattern or add appropriate *where* conditions to the rule to avoid directly matching previous replacements.

```
* TXL0510E rule/function 'RULENAME' - (TXL implementation limit) Transform
      recursion limit exceeded
```

The available stack space was exhausted while running the transformation. Normally this means that the allocated stack space was too small to run the transform, or possibly the indicated rule or the rules it calls are involved in a nonterminating transform.

Recommended action: Increase TXL limits using the *-size* option. If a nonterminating transform is suspected, check the indicated rule and the rules it calls for termination. If necessary, add appropriate termination conditions.

```
* TXL0511E rule/function 'RULENAME' - Transform interrupted by user
```

Execution of a TXL program was halted by a termination signal (normally ^C typed on the terminal) while executing the indicated rule. If the run was taking much longer than expected, it is possible that the indicated rule or the rules it calls are involved in a nonterminating transform.

Recommended action: Make sure that the rules involved will terminate. If necessary, add a termination condition.

```
* TXL0512E rule/function 'RULENAME' - Fatal TXL error (signal)
```

The TXL transformer caught an unexpected interrupt while applying the indicated rule/function.

Recommended action: Possibly caused by a full virtual memory swap space. Reboot the computer and try the run again. If the problem persists, save a copy of the entire TXL program, its include files, and the input (if any) used for the failing run. Report the problem as a TXL bug, including the copy with the bug report.

```
* TXL0513W - (Warning) NNN garbage recoveries were required
     (larger size recommended for improved performance)
```

A larger than usual number of garbage recoveries was required to complete the transformation. Garbage recoveries significantly slow down TXL execution.

Recommended action: Increase TXL limits using the *-size* option. If the size required seems inordinately large, use the TXL profiler *txlprof* to analyze rule space usage and tune rules appropriately.

```
* TXL0514E - Unable to create TXL profile file 'txl.rprofout'
```

This error indicates that the TXL profiler was unable to create one of its output files, probably because the user does not have permission to create a file in the current working directory.

Recommended action: Change directory to a directory where you have permission to create files, and rerun the TXL profiler.

```
* TXL0521E - Out of tree space - NNN trees have been allocated
```

The total number of trees needed to implement the transformation is larger than the TXL implementation can handle at the present size. Normally this indicates that the TXL size is too small for the transformation, but it is also a possible indication of an infinite transform (i.e., a program that never terminates).

Recommended action: Increase TXL implementation limits using the *-size* option. If an infinite transform is suspected, double check that the rules of the program will terminate.

```
* TXL0522E - Out of kid space - NNN kids have been allocated
```

The total number of kids (subtree cells) needed to implement the transformation is larger than the TXL implementation can handle at the present size. Normally this indicates that the TXL size is too small for the transformation, but it is also a possible indication of an infinite transform (i.e., a program that never terminates).

Recommended action: Increase TXL implementation limits using the *-size* option. If an infinite transform is suspected, double check that the rules of the program will terminate.

```
* TXL0531E - (TXL implementation limit) Too many rule/function definitions
     (> NNN)
```

The total number of rules and functions, including both built-in and programmer-defined rules and functions, is larger than the TXL implementation can handle.

Recommended action: Work around the problem by merging rules and functions where possible, or by splitting the transformation into a sequence of two separate programs. For technical reasons critical to TXL efficiency, this limit is not generally extensible in TXL implementations. If the problem persists, report this limit as a TXL bug.

```
* TXL0532E PARTNAME of 'RULENAME' - Variable 'VARNAME' has not been defined
```

In the indicated rule, TXL variable 'VARNAME' is used to but never bound, or used before it is bound. In TXL all variables must be bound before they are used.

Recommended action: Check that the indicated variable is bound before it is used.

```
* TXL0533E PARTNAME of 'RULENAME' - Variable 'VARNAME' has already been
      defined
```

The indicated pattern binds a variable that has already been bound in the rule. TXL does not permit reassignment of local variables.

Recommended action: Change the name of the bound variable to a new name.

```
* TXL0534E PARTNAME of 'RULENAME' - (TXL implementation limit) Rule/function
      has too many local variables (> NNN)
```

The indicated rule has more local variables than the TXL implementation can handle.

Recommended action: Reprogram the rule to reduce the number of constructs, deconstructs, imports, exports and pattern variables using subrules. For technical reasons critical to TXL efficiency, this limit is not generally extensible in TXL implementations.

```
* TXL0535E PARTNAME of 'RULENAME' - (TXL implementation limit) Rule/function
      has too many rule calls (> NNN)
```

The indicated rule has more subrule calls than the TXL implementation can handle.

Recommended action: Reprogram the rule to reduce the number of subrule calls in the rule by introducing intermediate functions to apply sets of subrules. For technical reasons critical to TXL efficiency, this limit is not generally extensible in TXL implementations.

```
* TXL0536E PARTNAME of 'RULENAME' - Scope of [.] predefined function is not a
      repeat
```

In the indicated context, the [.] built-in function is applied to a TXL variable that is not a [repeat]. [.] is defined only when the scope is of type [repeat X] or [repeat X+] for some [X], and the parameter is either of type [repeat X], [repeat X+] or [X].

Recommended action: Check that the scope of the function application is of type [repeat X] or [repeat X+] for some type [X].

* `TXL0537E PARTNAME of 'RULENAME' - Parameter of [.] predefined function does not match scope type`

  The [.] built-in function is used in the indicated context with an actual parameter that does not match the type of the TXL variable the function is applied to. [.] is defined only when the scope is of type [repeat X] or [repeat X+] for some [X], and the parameter is either of type [repeat X], [repeat X+] or [X].

  Recommended action: Check that the scope and parameter of the function application are of appropriate matching types.

* `TXL0538E PARTNAME of 'RULENAME' - Scope of [,] predefined function is not a list`

  In the indicated context, the [,] built-in function is applied to a TXL variable that is not a list. [,] is defined only when the scope is of type [list X] or [list X+] for some [X], and the parameter is either of type [list X], [list X+] or [X].

  Recommended action: Check that the scope of the function application is of type [list X] or [list X+] for some type [X].

* `TXL0539E PARTNAME of 'RULENAME' - Parameter of [,] predefined function does not match scope type`

  The [,] built-in function is used in the indicated context with an actual parameter that does not match the type of the TXL variable the function is applied to. [,] is defined only when the scope is of type [list X] or [list X+] for some [X], and the parameter is either of type [list X], [list X+] or [X].

  Recommended action: Check that the scope and parameter of the function application are of appropriate matching types.

* `TXL0540E PARTNAME of 'RULENAME' - Scope of [+], [-], [*], [/], [div] or [rem] is not a number, string or token`

  In the indicated context, one of the [+], [-], [*], [/], [div] or [rem] built-in functions is applied to a TXL variable that is not of an appropriate type. [+] is defined only when both the scope and the parameter are of type [number] (for addition) or [id], [stringlit], [charlit], or other token type (for text concatenation). [-], [*], [/], [div] and [rem] are defined only when both the scope and the parameter are of type [number].

  Recommended action: Check that the scope and parameter of the function application are of appropriate matching types.

* `TXL0541E PARTNAME of 'RULENAME' - Parameter of [+], [-], [*], [/], [div] or [rem] predefined function does not match scope type`

  One of the [+], [-], [*], [/], [div] or [rem] built-in functions is used the indicated context with an actual parameter of a type different from the type of the TXL variable the function is applied to. [+] is defined only when both the scope and the parameter are of type [number] (for addition) or [id], [stringlit], [charlit], or other token type (for text concatenation). [-], [*], [/], [div] and [rem] are defined only when both the scope and the parameter are of type [number].

Recommended action: Check that the parameter is of a type compatible with the TXL variable the function is applied to.

* `TXL0542E PARTNAME of 'RULENAME' - Scope of [:] predefined function is not a string, identifier or token`

In the indicated context, the [:] (textual substring) built-in function is applied to a TXL variable of a type other than [id], [stringlit], [charlit] or other token type. [:] is defined only when the scope is an [id], [stringlit], [charlit] or other token type and both parameters are of type [number].

Recommended action: Check that the scope of the function application is of type [id], [stringlit, [charlit] or other token type.

* `TXL0543E PARTNAME of 'RULENAME' - Parameters of [:] predefined function are not numbers`

The [:] (textual substring) built-in function is used the indicated context with one or both parameters of a type other than [number]. [:] is defined only when the scope is an [id], [stringlit], [charlit] or other token type and both parameters are of type [number].

Recommended action: Check that both parameters are of type [number].

* `TXL0544E PARTNAME of 'RULENAME' - Scope of [#] predefined function is not a number`

In the indicated context, the [#] (textual length) built-in function is applied to a TXL variable that is not of type [number]. [#] is defined only when the scope is a [number] and the parameter is an [id], [stringlit], [charlit] or other token type.

Recommended action: Check that the scope of the function application is of type [number].

* `TXL0545E PARTNAME of 'RULENAME' - Parameter of [#] predefined function is not a string, identifier or token`

The [#] (textual length) built-in function is used the indicated context with an actual parameter that is not of type [id], [stringlit], [charlit] or other token type. [#] is defined only for parameters of those types.

Recommended action: Check that the parameter is of one of the appropriate types.
If the length of a [repeat] or [list] was intended, use the [length] predefined function instead.

* `TXL0546E PARTNAME of 'RULENAME' - Parameter of [=] or [~=] predefined function does not match scope type`

The [=] or [~=] built-in function is used the indicated context with an actual parameter of a type different from the type of the TXL variable the function is applied to. [=] and [~=] are defined only when both the scope and the parameter are of the same nonterminal type.

Recommended action: Check that the parameter is of the same type as the TXL variable the function is applied to.

* `TXL0547E PARTNAME of 'RULENAME' - Scope of [>], [<], [>=] or [<=] predefined function is not a number, string or identifier`

  In the indicated context, one of the [>], [<], [>=] or [<=] built-in functions is applied to a TXL variable that is not of an appropriate type. [>], [<], [>=] or [<=] are defined only when both the scope and the parameter are of type [id], [stringlit], [charlit] or [number].

  Recommended action: Check that the scope of the function application is of type [id], [stringlit], [charlit] or [number].

* `TXL0548E PARTNAME of 'RULENAME' - Parameter of [>], [<], [>=] or [<=] predefined function does not match scope type`

  One of the [>], [<], [>=] or [<=] built-in functions is used the indicated context with an actual parameter of a type different from the type of the TXL variable the function is applied to. [>], [<], [>=] or [<=] are defined only when both the scope and the parameter are of type [number] (for numerical ordering), or [id], [stringlit], [charlit] or other token type (for textual ordering).

  Recommended action: Check that the parameter is of a type compatible with the TXL variable the function is applied to.

* `TXL0549E PARTNAME of 'RULENAME' - Scope of [^] predefined function is not a repeat`

  In the indicated context, the [^] built-in function is applied to a TXL variable that is not a [repeat]. [^] is defined only when the scope is of type [repeat X] for some type [X].

  Recommended action: Check that the scope of the function application is of type [repeat X] for some type [X].

* `TXL0550E PARTNAME of 'RULENAME' - Parameters of [$] predefined function are of different types`

  The [$] built-in function is used the indicated context with a second actual parameter of a type different from the type of the first parameter. [$] is defined only when both parameters are of the exact same nonterminal type.

  Recommended action: Check that the parameters are of the same type, or write an appropriate one-pass rule to do the replacement.

* `TXL0551E PARTNAME of 'RULENAME' - Scope of [!] predefined function is not an identifier`

  In the indicated context, the [!] (uniqueify identifier) built-in function is applied to a TXL variable that is not of type [id] or other identifier type. [!] is defined only when the scope is an [id] or other identifier type.

  Recommended action: Check that the scope of [!] is of type [id] or other identifier type.

* `TXL0552E PARTNAME of 'RULENAME' - Scope of [_] predefined function is not an identifier`

  In the indicated context, the [_] (concatenate using underscore) built-in function is applied to a TXL variable that is not of type [id] or other identifier type. [_] is defined only when both the scope and the parameter are of type [id] or other identifier type.

  Recommended action: Check that the scope of [_] is of type [id] or other identifier type.

* `TXL0553E predefined function [_], called from 'RULENAME' - Parameter of [_] predefined function is not an identifier`

  The [_] (concatenate using underscore) built-in function is used the indicated context with an actual parameter that is not of type [id] or other identifier type. [_] is defined only when both the scope and the parameter are of type [id] or other identifier type.

  Recommended action: Check that the parameter of [_] is of type [id] or other identifier type.

* `TXL0554E predefined function [quote/unparse], called from 'RULENAME' - Scope of [quote] or [unparse] predefined function is not a string, identifier or token`

  In the indicated context, the [quote] or [unparse] built-in function is applied to a TXL variable that is not of type [id], [stringlit], [charlit] or other token type. [quote] and [unparse] are defined only when both the is of type [id], [stringlit], [charlit] or other token type.

  Recommended action: Check that the scope of [quote] or [unparse] is of type [id], [stringlit], [charlit] or other token type.

* `TXL0555E PARTNAME of 'RULENAME' - Scope of [unquote] predefined function is not an identifier or token`

  The [unquote] built-in function is applied to a TXL variable that is not of type [id] or other unquoted token type. [unquote] is defined only when the scope is an identifier or other token type other than [charlit] and [stringlit].

  Recommended action: Check that the scope of the function application is of type [id] or other unquoted token type. If conversion to another type is required, use [parse] instead.

* `TXL0556E PARTNAME of 'RULENAME' - Parameter of [unquote] predefined function is not a string`

  The [unquote] built-in function is used with a parameter that is not of type [stringlit] or [charlit]. [unquote] is defined only when the parameter is of type [stringlit] or [charlit].

  Recommended action: Check that the parameter of the function application is of type [stringlit] or [charlit]. If conversion from another type is required, use [quote] to convert it to a [stringlit] or [charlit] first.

```
* TXL0557E PARTNAME of 'RULENAME' - Parameter of [parse] predefined function
      is not a string, identifier or token
```

The [parse] (scan and parse text) built-in function is used the indicated context with an actual parameter that is not of type [id], [stringlit], [charlit] or other token type. [parse] is defined only for parameters of those types.

Recommended action: Check that the parameter is of one of the appropriate types. If not, and the text to be reparsed is limited to at most 32767 characters, use [quote] to create a parseable string. If a more general large-scale reparse is required, use [reparse] or [write] followed by [read].

```
* TXL0558E PARTNAME of 'RULENAME' - Parameter of [read] or [write] predefined
      function is not a string or identifier
```

The [read] (scan and parse file) or [write] (unparse and write text to file) built-in function is used in the indicated context with an actual parameter that is not of type [stringlit], [charlit], [id] or other identifier type. The file name must be given as a string or identifier.

Recommended action: Check that the file name parameter is given as a string or identifier.

```
* TXL0559E PARTNAME of 'RULENAME' - Parameter of [getp] predefined function is
      not a string
```

The [getp] (interactive input with prompt) built-in function is used in the indicated context with an actual parameter that is not of type [stringlit] or [charlit]. The input prompt must be given as a string.

Recommended action: Check that the input prompt parameter is given as a [stringlit] or [charlit].

```
* TXL0560E PARTNAME of 'RULENAME' - Parameter of [putp] predefined function is
      not a string
```

The [putp] (interactive output with format) built-in function is used in the indicated context with an actual parameter that is not of type [stringlit] or [charlit]. The output format must be given as a string.

Recommended action: Check that the output format parameter is given as a [stringlit] or [charlit].

```
* TXL0561E PARTNAME of 'RULENAME' - Scope of [index] predefined function is
      not a number
```

In the indicated context, the [index] (textual substring search) built-in function is applied to a TXL variable that is not of type [number]. [index] is defined only when the scope is a [number] and the parameters are both of type [id], [stringlit], [charlit] or other token type.

Recommended action: Check that the scope of the function application is of type [number].

* `TXL0562E PARTNAME of 'RULENAME' - Parameters of [index] predefined function are not strings, identifiers or tokens`

    In the indicated context, the [index] (textual substring search) built-in function is used with parameters that are not of type [id], [stringlit], [charlit] or other token type. [index] is defined only when the scope is a [number] and the parameters are both of type [id], [stringlit], [charlit] or other token type.

    Recommended action: Check that the parameters of the function are of appropriate types.

* `TXL0563E PARTNAME of 'RULENAME' - Scope of [grep] predefined function is not a string, identifier or token`

    In the indicated context, the [grep] (textual substring containment test) built-in function is used with a scope that is not of type [id], [stringlit], [charlit] or other token type. [grep] is defined only when the scope and parameter are both of type [id], [stringlit], [charlit] or other token type.

    Recommended action: Check that the scope of the function is of an appropriate type.

* `TXL0564E PARTNAME of 'RULENAME' - Parameter of [grep] predefined function is not a string, identifier or token`

    In the indicated context, the [grep] (textual substring containment test) built-in function is used with a parameter that is not of type [id], [stringlit], [charlit] or other token type. [grep] is defined only when the scope and parameter are both of type [id], [stringlit], [charlit] or other token type.

    Recommended action: Check that the parameter of the function is of an appropriate type.

* `TXL0565E PARTNAME of 'RULENAME' - Scope of [length] predefined function is not a number`

    The [length] (length of a sequence) built-in function is applied to a TXL variable that is not of type [number]. [length] is defined only when its scope is of type [number] and its parameter is a [repeat] or [list].

    Recommended action: Check that the scope of the function application is of type [number].

* `TXL0566E PARTNAME of 'RULENAME' - Parameter of [length] predefined function is not a [repeat] or [list]`

    The [length] (length of a sequence) built-in function is called using an actual parameter that is not a [repeat] or [list]. [length] is defined only when its scope is of type [number] and its parameter is of type [repeat X] or [list X] for some [X].

    Recommended action: Check that the actual parameter of the function application is a [repeat] or [list].

* `TXL0567E PARTNAME of 'RULENAME' - Scope of [select] predefined function is not a [repeat] or [list]`

    The [select] (subsequence) built-in function is applied to a TXL variable that is not a [repeat] or [list]. [select] is defined only when the scope is of type [repeat X] or [list X] for some [X].

Recommended action: Check that the scope of the function application is a [repeat] or [list].

```
* TXL0568E PARTNAME of 'RULENAME' - Parameters of [select] predefined function
      are not numbers
```

The [select] (subsequence) built-in function is used with one or more actual parameters that are not of type [number]. [select] is defined only when the scope is of type [repeat X] or [list X] for some [X] and the parameters are both of type [number].

Recommended action: Check that the parameters of the function application are both of type [number].

```
* TXL0569E PARTNAME of 'RULENAME' - Scope of [head] or [tail] predefined
      function is not a [repeat] or [list]
```

The [head] (leading subsequence) or [tail] (trailing subsequence) built-in function is applied to a TXL variable that is not a [repeat] or [list]. [head] and [tail] are defined only when the scope is of type [repeat X] or [list X] for some [X].

Recommended action: Check that the scope of the function application is a [repeat] or [list].

```
* TXL0570E PARTNAME of 'RULENAME' - Parameter of [head] or [tail] predefined
      function is not a number
```

The [head] (leading subsequence) or [tail] (trailing subsequence) built-in function is used with an actual parameter that is not of type [number]. [head] and [tail] are defined only when the scope is of type [repeat X] or [list X] for some [X] and the parameter is of type [number].

Recommended action: Check that the parameter of the function application is of type [number].

```
* TXL0571E PARTNAME of 'RULENAME' - Parameter of [quit] predefined function is
      not a number
```

The [quit] (immediate exit) built-in function is used with an actual parameter that is not a number. [quit] is defined only when the exit code parameter is of type [number].

Recommended action: Check that the exit code parameter is of type [number].

```
* TXL0572E PARTNAME of 'RULENAME' - Parameter of [fget] predefined function is
      not a string filename
```

The [fget] (interactive scan and parse line from file) built-in function is used in the indicated context with a parameter that is not of type [stringlit] or [charlit]. The file name parameter must be given as a string.

Recommended action: Check that the file name parameter is given as a string.

```
* TXL0573E PARTNAME of 'RULENAME' - Parameter of [fput] predefined function is
      not a string filename
```

The [fput] (interactive unparse and write line to file) built-in function is used in the indicated context with a parameter that is not of type [stringlit] or [charlit]. The file name parameter must be given as a string.

Recommended action: Check that the file name parameter is given as a string.

```
* TXL0574E PARTNAME of 'RULENAME' - First parameter of [fputp] predefined
      function is not a string filename
```

The [fputp] (interactive unparse and format line to file) built-in function is used in the indicated context with a first (file name) parameter that is not of type [stringlit] or [charlit]. The file name parameter must be given as a string.

Recommended action: Check that the file name parameter is given as a string.

```
* TXL0575E PARTNAME of 'RULENAME' - Second parameter of [fputp] predefined
      function is not a string
```

The [fputp] (interactive unparse and format line to file) built-in function is used in the indicated context with a second (format) parameter that is not of type [stringlit] or [charlit]. The format parameter must be given as a string.

Recommended action: Check that the format parameter is given as a string.

```
* TXL0576E PARTNAME of 'RULENAME' - Parameter of [fclose] predefined function
      is not a string filename
```

The [fclose] (close interactive line file) built-in function is used in the indicated context with a parameter that is not of type [stringlit] or [charlit]. The file name parameter must be given as a string.

Recommended action: Check that the file name parameter is given as a string.

```
* TXL0577E PARTNAME of 'RULENAME' - Parameter of [pragma] predefined function
      is not an options string
```

The [pragma] (interactively change options) built-in function is used in the indicated context with a parameter that is not of type [stringlit] or [charlit]. The options string parameter must be given as a string.

Recommended action: Check that the options string parameter is given as a string.

```
* TXL0578E PARTNAME of 'RULENAME' - Scope of [LS_explode] predefined function
      is not of type [repeat id]
```

In the indicated context, the [LS_explode] (Legasys private explode character string) built-in function is applied to a TXL variable that is not of type [repeat id]. [LS_explode] is defined only when the scope is of type [repeat id] and the parameter is of type [charlit].

Recommended action: Check that the scope is of type [repeat id].

```
* TXL0579E PARTNAME of 'RULENAME' - Parameter of [LS_explode] predefined
      function is not of type [charlit]
```

The [LS_explode] (Legasys private explode character string) built-in function is used in the \ indicated context with a parameter that is not of type [charlit]. [LS_explode] is defined only when the scope is of type [repeat id] and the parameter is of type [charlit].

Recommended action: Check that the parameter is of type [charlit].

```
* TXL0580E PARTNAME of 'RULENAME' - Scope of [LS_quote] predefined function is
      not of type [charlit]
```

The [LS_quote] (Legasys private unexplode character string) built-in function is used in the indicated context with a scope that is not of type [charlit]. [LS_quote] is defined only when the scope is of type [charlit].

Recommended action: Check that the scope of the function application is of type [charlit].

```
* TXL0581E PARTNAME of 'RULENAME' - Parameter of [system] predefined function
      is not a command string
```

The [system] (run system shell command) built-in function is used in the indicated context with a parameter that is not of type [stringlit] or [charlit]. The command string parameter must be given as a string.

Recommended action: Check that the command string parameter is given as a string.

```
* TXL0582E PARTNAME of 'RULENAME' - Scope of [pipe] predefined function is not
      a string
```

In the indicated context, the [pipe] (run piped system shell command) built-in function is applied to a TXL variable that is not of type [stringlit] or [charlit]. The scope of the [pipe] function must be a string.

Recommended action: Check that the scope of the function application is of type [stringlit] or [charlit].

```
* TXL0583E PARTNAME of 'RULENAME' - Parameter of [pipe] predefined function is
      not a command string
```

The [pipe] (run piped system shell command) built-in function is used in the indicated context with a parameter that is not of type [stringlit] or [charlit]. The command string parameter must be given as a string.

Recommended action: Check that the command string parameter is given as a string.

```
* TXL0591E external function [FUNCTIONNAME] called from [RULENAME] - Function
      is not implemented
```

The indicated external function is not implemented in the *user.c* file of the local version of TXL, or has not been linked into the TXL library.

Recommended action: Check that the function name is spelled correctly. If the function name is spelled correctly and should be implemented, report this problem to your local TXL administrator.

```
* TXL0592E predefined function [FUNCTIONNAME] called from [RULENAME] -
      Division by zero
```

In the indicated context, the [/], [div] or [rem] built-in function was called with a parameter that has value zero. Division is undefined for a divisor of zero.

Recommended action: Modify the logic of the program to avoid division by zero.

```
* TXL0592E predefined function [FUNCTIONNAME] called from [RULENAME] -
      Unable to open input file 'FILENAME' (too many open files)
```

In the indicated context, the indicated built-in function was unable to open its input file because too many files are open in the program.

Recommended action: Use [fclose] to close files as they are no longer needed. This limit is not extensible - if no files can be closed, modify program logic to reduce the need for so many simultaneously open files.

```
* TXL0592E predefined function [FUNCTIONNAME] called from [RULENAME] -
      Unable to open input file 'FILENAME'
```

In the indicated context, the indicated built-in function was unable to open its input file.

Recommended action: Check that the file name is spelled correctly. Remember that TXL file names are case sensitive. Check that the file exists and is readable. If the problem persists, report this as a TXL bug.

```
* TXL0592E predefined function [FUNCTIONNAME] called from [RULENAME] -
      Unable to open output file 'FILENAME' (too many open files)
```

In the indicated context, the indicated built-in function was unable to open its output file because too many files are open in the program.

Recommended action: Use [fclose] to close files as they are no longer needed. This limit is not extensible - if no files can be closed, modify program logic to reduce the need for so many simultaneously open files.

```
* TXL0592E predefined function [FUNCTIONNAME] called from [RULENAME] -
      Unable to open output file 'FILENAME'
```

In the indicated context, the indicated built-in function was unable to create its output file. Most often this is because you do not have write permission in the present working directory or the directory where the file is to be created.

Recommended action: Check that the program is being run in the intended directory and that the permissions on the directory where the file is to be created allow writing.

* `TXL0592E predefined function [LS_quote] called from [RULENAME] - [LS_quote] is obsolete, use [quote] instead`

  [LS_quote] has been superceded by a newer version of the [quote] built-in function.

  Recommended action: Replace all uses of [LS_quote] with [quote].

* `TXL0592E predefined function [pipe] called from [RULENAME] - [pipe] is not implemented under MPW`

  The MPW environment does not permit shell subprocesses, so [pipe] cannot be implemented in that environment.

  Recommended action: Run the program in a Unix or Windows environment.

* `TXL0592E predefined function [system] called from [RULENAME] - [system] is not implemented under MPW`

  The MPW environment does not permit shell subprocesses, so [system] cannot be implemented in that environment.

  Recommended action: Run the program in a Unix or Windows environment.

* `TXL0911W - (Warning) Available stack space too small for TXL stack at this size (TXL stack reduced from NNN to NNN to fit)`

  The available stack segment memory is smaller than the preferred stack size for the current TXL transform size. On Macintosh systems running MPW, this normally means that HEXA resource #128 of the MPW Shell is smaller than NNN + 128k. On UNIX systems, this normally means that the default per-process stack limit on the system is too small to run TXL at the current transform size.

  Recommended action: On UNIX, use the command 'unlimit stacksize' to remove the limit. On MPW, use the resource editor to edit the HEXA resource of the MPW Shell to be at least 0020 0000 hex.

* `TXL0912E - Available stack space too small for minimum TXL stack`

  The available stack segment memory is smaller than the minimum required to run TXL (approx. 500 kbytes). On Macintosh systems running MPW, this normally means that HEXA resource #128 of the MPW Shell is smaller than 600k (2000k is recommended for running TXL). On UNIX systems, this normally means that the default per-process stack limit on the system is too small to run TXL.

  Recommended action: On UNIX, use the command 'unlimit stacksize' to remove the limit. On MPW, use the resource editor to edit the HEXA resource of the MPW Shell to be at least 0020 0000 hex.

```
* TXL0921E - (TXL internal error) Garbage collection failure, variable
      'VARNAME'
```

This error indicates a catastrophic failure in the TXL compiler/interpreter.

Recommended action: Save a copy of the entire TXL program, its include files, and the input (if any) used for the failing run. Report the problem as a TXL bug, including the copy with the bug report.

```
* TXL0922E PARTNAME of 'RULENAME' - Garbage recovery unable to recover enough
      space to continue (a larger size is required for this transform)
```

The TXL garbage collector was unable to recover enough tree space to successfully continue the transformation from the indicated point.

Recommended action: Increase TXL limits using the *-size* option.

```
* TXL0931E TXL bootstrap - (TXL internal error) Too many symbols in TXL
      bootstrap
* TXL0932E TXL bootstrap - (TXL internal error) Syntax error in TXL bootstrap
      - expected 'TOKEN' + got 'TOKEN'
* TXL0933E TXL bootstrap - (TXL internal error) Syntax error in TXL bootstrap
      - expected '|', got 'TOKEN'
* TXL0934E TXL bootstrap - (TXL internal error) Syntax error in TXL bootstrap
      - multiple tokens in choice alternative
* TXL0935E TXL bootstrap - (TXL internal error) [TYPENAME] has not been
      defined
```

These errors indicate a catastrophic failure in the TXL compiler/interpreter.

Recommended action: Save a copy of the entire TXL program, its include files, and the input (if any) used for the failing run. Report the problem as a TXL bug, including the copy with the bug report.

```
* TXL0941E - (TXL implementation limit) Too many -I include directories
      (> NNN)
```

The total number of include libraries specified using *-I* command line options is larger than the TXL implementation can handle.

Recommended action: This limit is not extensible. Merge library directories or use symbolic links to reduce the number of include libraries. If the problem persists, report this problem as a bug.

```
* TXL0942E - (TXL implementation limit) Too many preprocessor symbols (> NNN)
```

The total number of different preprocessor symbols used in the program is larger than the TXL implementation can handle.

Recommended action: This limit is not extensible. Reduce the dependency on preprocessor symbols by physically splitting source versions. If the problem persists, report this problem as a bug.

```
* TXL0943E - Unrecognized debugging option '-DOPTION'
```

The indicated debugging switch is not implemented in this version of TXL.

Recommended action: Check the spelling of the option. TXL debugging options are always lower case identifiers following -D, for example '-Dparse'. If the desired option is not implemented, use other debugging options or *txdb* to investigate the problem.

```
* TXL0944E - Can't find TXL program file 'PROGNAME.Txl'
```

The indicated TXL program file could not be found by TXL in the present working directory, the *Txl* subdirectory of the present working directory, any of the directories specified using *-I* command line options, or the system TXL library directory. Most often this is due to a misspelling of the TXL program or input file name.

Recommended action: Check the spelling of the given file names. If correct, then see the section "The 'txl' Command" of this manual to check that you are using the command correctly.

```
* TXL0945E - Can't open TXL load file 'PROGNAME.CTxl'
```

When using the *-load* option, the indicated TXL compiled tree code file (i.e., *.CTxl* file) could not be found by TXL in the present working directory, the *Txl* subdirectory of the present working directory, any of the directories specified using *-I* command line options, or the system TXL library directory. Most often this is due to a misspelling of the TXL program or input file name, or incorrect use of the *-load* option.

Recommended action: Check the spelling of the given file names. If correct, then see the section "The 'txl' Command" of this manual to check that you are using the command correctly.

```
* TXL0951E - [debug] predefined function not implemented in standalone
      applications
```

A standalone TXL application compiled using *txlapp* uses the [debug] built-in function. For efficiency reasons, this function is not implemented in standalone applications.

Recommended action: Use [print], [message], [put] or [putp] to output the required information.

```
* TXL0952W - (Warning) Forced to split [SPOFF] output at line boundary
```

The range of an [SPOFF] formatting directive yielded an output sequence longer than the specified maximum output line length. The output was split as necessary to stay within the maximum output width.

Recommended action: Check that every [SPOFF] is matched by an [SPON]. If possible, split [SPOFF] ranges into a finer grain. Specify a longer output line length using *-w*.

```
* TXL0953W - (Warning) Output token too long for output width
* TXL0954W - (Warning) Output token too long for output width
```

A single output token was longer than the specified maximum output line length. The token was output on a line by itself.

Recommended action: Specify a longer output line length using *-w*.

```
* TXL0955W - (Warning) Forced to split [SPOFF] output at line boundary
```

The range of an [SPOFF] formatting directive yielded an output sequence longer than the specified maximum output line length. The output was split as necessary to stay within the maximum output width.

Recommended action: Check that every [SPOFF] is matched by an [SPON]. If possible, split [SPOFF] ranges into a finer grain. Specify a longer output line length using *-w*.

```
* TXL0956W - (Warning) Output token too long for output width
* TXL0957W - (Warning) Output token too long for output width
```

A single output token was longer than the specified maximum output line length. The token was output on a line by itself.

Recommended action: Specify a longer output line length using *-w*.

```
* TXL0958W - (Warning) Forced to use continuation card for non-stringlit
      (check output!)
```

Using card format output (*-cobolout*), a single output token was longer than 65 characters, forcing a continuation card. This most likely indicates an error in the program, since Cobol tokens other than character literals cannot be longer than 31 characters.

Recommended action: Check the program for errors in Cobol token handling.

```
* TXL0959E - (TXL implementation limit) Output recursion limit exceeded
      (increase TXL stack size)
```

The available stack space was exhausted while unparsing the result of the transformation. Normally this means that the allocated stack space was too small.

Recommended action: Increase TXL limits using the *-size* option. If the problem persists, report this problem as a TXL bug.

```
* TXL0960E - (TXL implementation limit) Result of [quote] predefined function
      exceeds maximum line length (32767 characters)
```

The output text of a TXL variable quoted by the [quote] built-in function was longer than the TXL implementation limit of 32,767 characters.

Recommended action: Write the text to a file using [write], and read it back using [read] as required.

```
* TXL0961E - (TXL implementation limit) Result of [LS_quote] predefined
      function exceeds maximum line length (32767 characters)
* TXL0962E - (TXL implementation limit) Result of [LS_quote] predefined
      function exceeds maximum line length (32767 characters)
```

The output text of a TXL variable quoted by the [LS_quote] built-in function was longer than the TXL implementation limit of 32,767 characters.

Recommended action: Write the text to a file using [write], and read it back using [read] as required.

```
* TXL0971E - (TXL internal error) Fatal TXL error in copyTree
```

This error indicates a catastrophic failure in the TXL compiler/interpreter.

Recommended action: Save a copy of the entire TXL program, its include files, and the input (if any) used for the failing run. Report the problem as a TXL bug, including the copy with the bug report.

```
* TXL0981E - Out of tree space - NNN trees have been allocated
* TXL0982E - Out of tree space - NNN trees have been allocated
* TXL0983E - Out of tree space - NNN trees have been allocated
```

The total number of trees needed to implement the transformation is larger than the TXL implementation can handle at the present size. Normally this indicates that the TXL size is too small for the transformation, but it is also a possible indication of an infinite transform (i.e., a program that never terminates).

Recommended action: Increase TXL implementation limits using the *-size* option. If an infinite transform is suspected, double check that the rules of the program will terminate.

```
* TXL0984E - Out of kid space - NNN kids have been allocated
* TXL0985E - Out of kid space - NNN kids have been allocated
* TXL0986E - Out of kid space - NNN kids have been allocated
* TXL0987E - Out of kid space - NNN kids have been allocated
* TXL0988E - Out of kid space - NNN kids have been allocated
* TXL0989E - Out of kid space - NNN kids have been allocated
```

The total number of kids (subtree cells) needed to implement the transformation is larger than the TXL implementation can handle at the present size. Normally this indicates that the TXL size is too small for the transformation, but it is also a possible indication of an infinite transform (i.e., a program that never terminates).

Recommended action: Increase TXL implementation limits using the *-size* option. If an infinite transform is suspected, double check that the rules of the program will terminate.

```
* TXL0991E - Unable to open output file 'FILENAME'
```

The output file specified in the *-o* command line option could not be created by TXL. Most often this is because you do not have write permission in the present working directory or the directory where the file is to be created.

Recommended action: Check that the program is being run in the intended directory and that the permissions on the directory where the file is to be created allow writing.

```
* TXL9999E - (Fatal) Unable to obtain TXL license
* TXL9999W - (Warning) TXL License has expired
* TXL9999E - (Fatal) TXL License has expired
* TXL9999I - (Information) Contact Legasys, license code NNN:NNN
```

These messages indicate problems or impending problems with your TXL license.

Recommended action: Contact Legasys quoting the license code and other information shown in the messages.