SPIRE-ALC-DOC-001891

Tcl/TOPE Coding Constraints for MOIS Reverse Engineering

Version:	Issue 1.0
Date:	18-Nov-2003
Author:	Damien Callet
Reference:	ALC-MOIS-TN-RHEA-0002
Filename:	TCLTOPERulesv1.doc
Approved by:	Ivo Bettens MOIS Product Manager

Rhea System S.A.
New Tech Center,
Avenue Einstein 2a,
B-1348 Louvain-La-Neuve,
Belgium

TEL : + 32 10 48 72 50
FAX : + 32 10 45 25 07
www.rheagroup.com

DISTRIBUTION

Name	Number of Copies
F. Chatte ALCATEL SPACE	1
I. Bettens RHEA	1

DOCUMENT STATUS SHEET

Date	Version	Author	Reason for change

DOCUMENT CHANGE RECORD

Date	Version	Changed Pages/Paragraphs

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Purpose

This document defines the Tcl/TOPE coding rules that must be applied to manually written Tcl/TOPE scripts so that they can be usefully interpreted and converted to a MOIS procedure by the MOIS Tcl/TOPE Reverse Engineering Tool.

1.2 Overview

The document is divided into the following sections:

- Constraints on the overall syntax (section 2)
- Constraints on the TCL structures (section 3)
- Constraints on the TOPE and Tcl statements (section)

1.3 Definitions

L_i Represents a line number.

1.4 Acronyms and Abbreviations

OL	Operation language
FCT	Function Statement in a MOIS procedure
DIR	Directive Statement in a MOIS procedure

1.5 Applicable Documents

ID	Document	Reference

1.6 Reference Documents

ID	Document	Reference
RD1	Tcl/TOPE Reverse engineering to MOIS version draft 1	ALC-MOIS-RQ-RHEA-0001

2 SYNTAX RULES

2.1 File format

Unix format files are not directly supported and must first be converted to DOS format. The 2 files have different line terminators as shown below:

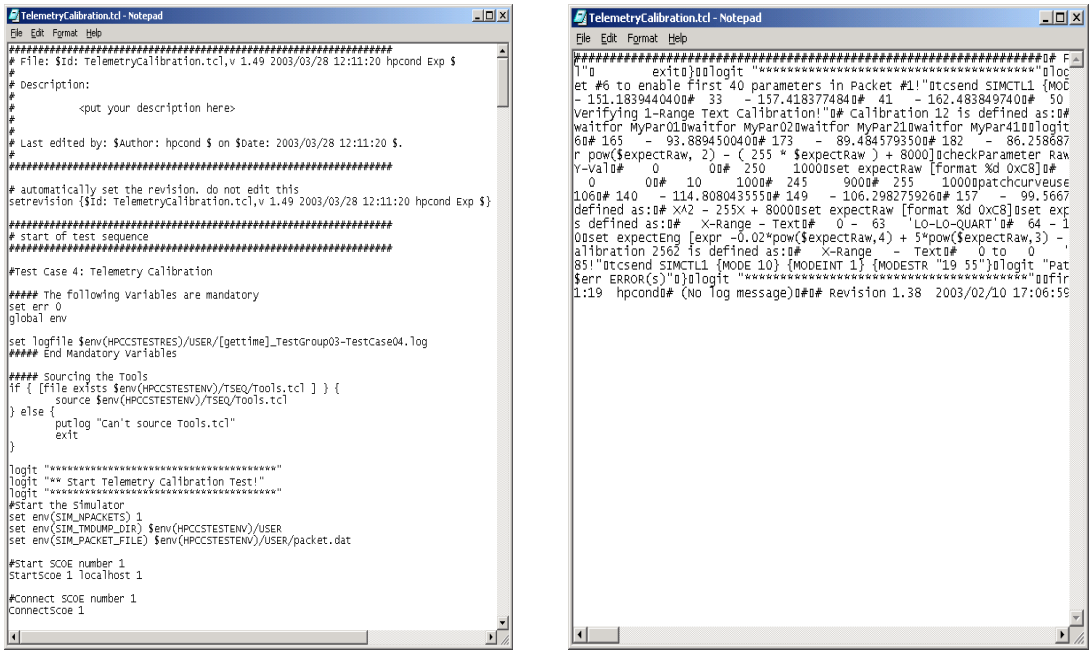

Figure 1

The content of these files are identical but the format is different. However, the conversion can be performed quite easily so this is not a real constraint.

2.2 Overall Syntax Rules

2.2.1 Basic syntax limitations

- Only one command per line is allowed.

  Example:

  ```
  Set var1 2
  Set var2 [expr $var1 + 1]
  ```

  ~~`Set var1 2; set var2 [expr $var1 + 1];`~~

- Space delimiters must be included between fields even where Tcl allows more flexibility:

  ```
  Set var1 [expr 1 + 2]
  ```

  ~~`Set var1 [expr 1+2]`~~

2.2.2 Comments

- All comment lines will be converted directly to MOIS Comment statements. Therefore comments included to make the script file more readable may unnecessarily clutter the converted MOIS procedure.
- Comments at the end of a line are not allowed.

```
~~Set var1 0 #Set variable 1~~
```

2.2.3 Variable handling

- Tcl lists and arrays must not be used as they are not supported by MOIS.
- Compound statements are not allowed. Instead intermediate variables must be used:

~~set seconds [expr [format %d [expr 0x[string trimleft [string range [getrawdata $myAcks] 20 27] 0]]] + [getepoch]]~~

```
set var0 [getrawdata $myAcks]
set var1 [string range $var1 20 27]
set var2 [string trimleft $var2 0]
set var3 [expr 0x$var2 ]
set var4 [format %d $var3]
set var5 [getepoch]
set seconds  [expr $var4 + $var5]
```

- Use of intermediate variables must not be used for all functions relative to telemetry (*function* [fetch <*telemetryname*>]). For example:

`Set var1 [isengvaluevalid [fetch TMA1001]].` In this case the `fetch TMA1001` will not have to be replaced by a `set variable`.

2.2.4 TOPE and Tcl Statements

- A list of supported TOPE and Tcl statements can be found in annex 1. If a function not defined in the annex 1 is used, it will be translated as a MOIS Free Text Directive (i.e. it will be stored in the MOIS Procedure as an uninterpreted string).
- The returned value of a TOPE/Tcl statement must be stored in a variable before it can be used:

```
“~~If { [expr 1 + 2] == 3 }{~~“
“Set var1 [expr 1 + 2]
If { $var1 == 3}{“
```

- Some Tcl/TOPE statements have an unlimited number of arguments as is the case for the `binary` Tcl statement. This not supported; the number of required arguments is given in the rules column of the table in annex 1.

3 TCL CONTROL STRUCTURES RULES

Certain Tcl control structures cannot be represented easily in a MOIS procedure and are therefore forbidden.

- `Foreach` has no exact equivalent in a MOIS procedure and is therefore not allowed.
- Imbricated control structures must not be used. For example:

```
If { $var1 == 0 } {
      If { $var2 == 1} {
            body
      }
}
```

must be replaced by

```
If { $var1 == 0 } && { $var2 == 1}{
      body
}
```

- Conditions in Tcl control structures must be simple checks against variables except for particular cases, which are listed later on section 4.3. For example:

```
if { [file exists [file join $env(HPCCSTESTENV) TSEQ Tools.tcl ]] } {
      source [file join $env(HPCCSTESTENV) TSEQ Tools.tcl]
} else {
      putlog "Can't source Tools.tcl"
      exit
}
```

must be replaced with the following code:

```
Set var0 [file join $env(HPCCSTESTENV) TSEQ Tools.tcl ]
Set var1 [file exists $var0]
if { $var1 == true } {
      source $var0
} else {
      putlog "Can't source Tools.tcl"
      exit
}
```

- Control structures must be constructed as shown in the following examples. In each case L_i represents the line number in the script:

  - `If` control structure.

```
L1: If { $count < 5 }{
L2:       Puts “count is less than five”
```

```
L3:} else {
L4:       Puts “count is not less than five”
L5:  }
```

- `For` control structure:

```
Set var1 [getrawdata $echo]
Set var2 [string length $var1]
L1: for { set i 0} { $i < $var2 } { set i [expr $i + 4] } {
L2:       body
L3:}
```

- `While` control structure:

```
L1: while { [getrawvalue [fetch TMA1001]] != 0 } {
L2:           body
L3: }
```

- `Switch` control structure:

```
L1: switch -- $count
L2:    1{
L3:    body
L4:} 2 {
L5: body2
L6:}
```

- The `“Proc”` and `“return”` keywords are not allowed.

4 TCL/TOPE STATEMENT RESTRICTIONS

4.1 "Tcsend" command

4.1.1 Supported options

Some tcsend options will not be taken into account in the reverse processing to a MOIS procedure. Only the **releasetime**, **executiontime, checks, ack and NOCRC** options will be taken into account; **patch** will be ignored.

4.1.2 Defined structure using a 'waitfor'

The supported structure for a tcsend command followed by a `waitfor` to synchronise its execution is:

```
L1: tcsend CMDA1001 referby rCMDA1001
L2: waitfor sCMDA1001 –until [getcompleted $rCMDA1001]
```

This will be converted to a MOIS CMD statement followed by a MOIS CTL SEV statement.

4.2 Wait for packet

The supported structure is:

```
L1: subscribepacket <pktid> referby <varname>
L2: waitfor –timeout <time> <varname>
L3: unsubscribepacket <pktid>
```

This will be converted to a MOIS PKT statement.

4.3 Verify Telemetry

The supported structure is:

```
L1: if { [getrawvalue [fetch <param>]] != <valuetocheck>}{
Li:       …
Ln-1      exit
Ln: }
```

This structure will be translated to a MOIS verify TLM statement within a PERFORM step. Both `getrawvalue` function and `getengvalue are supported.` If `getrawvalue` is used then the verify TLM statement will use the raw value. Otherwise if `getengvalue` is used the verify TLM statement will use the alias or calibrated value (depending on the telemetry type).

ANNEX 1

Tcl/TOPE statements	MOIS implementation	Mechanism	Arguments	Rules	Recognized formulation
append <varname> ?<value><value> …?	**DIR statement**		<varname> STRING ALS <value> STIRNG STR	1- The number of values is limited to 10	append <varname> ?<value> … <value>?
array	**n/a**				
asdtomsec <delta-time-string>	**FCT statement**		<delta-time-string> STRING STR	1- <delta-time-string> must be double quoted	set <varname> [asdtomsec <delta-time-string>]
asdtosec <time-string> ?<secs-var> <usecs_var>?	**FCT statement**		<time-string> STRING STR <secs-var> STRING ALS <usecs_var> STRING ALS	1- <time-string> must be double quoted, other arguments not.	set <varname> [asdtosec <time-string> ?<secs-var> <usecs_var>?]
attach <name>	**DIR statement**		<name> STRING ALS	1- <name> must not be double quoted	attach <name>
authorise ?-revoke? <name>	**DIR statement**		revoke **Optional** switch <name> STRING ALS	1- <name> must not be double quoted	authorise ?-revoke? <name>
bintohex <string>	**FCT statement**		<string> STRING STR	1- <string> must be double quoted	set <varname> [bintohex <string>]
bgerror	**n/a**				
binary format <formatString> ?<arg> <arg> …? binary scan <string> <formatString> ?<varname> <varname>…?	**FCT statement**		<formatString> STRING ALS <arg> STRING ALS <string> STRING ALS <varname> STIRNG ALS	1- The number of arguments will be limited to 10	1- set <varname> [binary format ?<arg> … <arg>?] 2- set <varname> [binary scan <string> <formatString> ?<varname> … <varname>?]
call <name> ?arg …?	**FOP Statement (synchronous)**		n/a	1- Name of FOP must not be delimited by double quotes. The number of arguments is limited to10.	call <name> ?arg …?
callasync ?-referby var? name ?arg…?	**FOP Statement (Asynchronous)**		n/a	1- Name of FOP must not be delimited by double quotes. The number of arguments is limited to10. 2- referby arguments are not taken into account	callasync name ?arg…?
cd ?<dirname>?	**DIR statement**		<dirname> STRING STR		1- cd ?<dirname>?
clock clicks ?-milliseconds? clock format <clockvalue> ?-format <string>? ?-gmt <boolean>? clock scan <dateString> ?-base <clockVal>? ?-gmt <boolean>? clock <seconds>	**FCT statement**		<milliseconds> STRING ALS <clockvalue> STRING ALS <string> STRING ALS <boolean> BOOLEAN DEC <dateString> STRING ALS <clockVal> STRING ALS <seconds> STRING ALS		1- set <varname> [clock seconds] 2- set <varname> [clock scan <dateString> ?-base <clockVal>? ?-gmt <boolean>?] 3- set <varname> [clock format <clockvalue> ?-format <string>? ?-gmt <boolean>?] 4- set <varname> [clock clicks ?- milliseconds?]

Tcl/TOPE statements	MOIS implementation	Mechanism	Arguments	Rules	Recognized formulation
close <channelId>	**DIR statement**		<channelId> INTEGER DEC		close <channelId>
concat ?<arg> <arg> … ?	**FCT statement**		<arg> STRING ALS		set <varname> [concat ?<arg> … <arg>?]
connect <name>	**DIR statement**		<name> STRING ALS	1- <name> must not be double quoted	connect <name>
detach <name>	**DIR statement**		<name> STRING ALS	1- <name> must not be double quoted	detach <name>
disconnect <name>	**DIR statement**		<name> STRING ALS	1- <name> must not be double quoted	disconnect <name>
displaystatus <message>	**DIR statement**		<message> STRING STR	1- <message> must be double quoted.	displaystatus <message>
enableparam <param-list>	**DIR statement**		<param-list> STIRNG ALS	1- Each parameter of the list must be separated by a space character. 2- The list must have braces as delimiters.	enableparam {<param-list>}
enablepacket <spid>	**DIR statement**		<spid> STRING ALS		enablepacket <spid>
enablegroup <grpid>	**DIR statement**		<grpid> STRING ALS		enablegroup <grpid>
eof <channelId>	**FCT statement**		<channelId> INTEGER DEC		set <varname> [eof <channelId>]
exec	**n/a**				
exit	**CTL/XIT statement**			1- Error code will not be supported 2- When exit is in an 'If' statement then it will be interpreted as a Verify TLM statement in a perform step.	exit
expr <arg> ?<arg> <arg> …?	**FCT statement**		<arg> STRING ALS	Number of arguments is limited to 10	set <varname> [expr <arg> ?<arg> … <arg>?]
fconfigure <channelId>	**FCT statement**		<channelId> INTEGER DEC		set <varname> [fconfigure <channelId>]
fcopy <inchan> <outchan> ?-size <size>? ?-command <callback>?	**DIR statement**		<inchan> STRING ALS <outchan> STRING ALS <size> INTEGER DEC <callback> STRING ALS		
Fetch	n/a	n/a	n/a	This TOPE statement cannot be called on its own in a Tcl/TOPE script. It will only be used as a switch in getParameterdata functions.	set <varname> [getname [fetch <par-name>]]
fileevent <channelId> <readable> ?<script>? fileevent <channelId> <writable> ?<script>?	**n/a**				
file	**n/a**				
format <formatString> ?<arg> <arg> …?	**FCT statement**		<formatString> STRING ALS <arg> STRING STR	Number of arguments is limited to 10.	set <varname> [format <formatString> ?<arg> … <arg>?]
fpblocked <channelId>	**FCT statement**		<channelId> INTEGER DEC		set <varname> [fpblocked <channelId>]

Tcl/TOPE statements	MOIS implementation	Mechanism	Arguments	Rules	Recognized formulation
getname [fetch <par-name>]	**SET statement**	This formulation will be directly replaced by the param-name.	n/a	1- A set varname must be placed before the getname function 2- No double quotes for the parameter name.	set <varname> [getname [fetch <par-name>]]

Getrawvalue [fetch <par-name>]	**TLM statement/FCT statement**	1- In case of check against a value, it will implemented as a TLM statement.
		2- In case of a set it will be implemented as a FCT statement
			<par-name> STRING ALS	1- getrawvalue must have a fetch argument.
				2- to intepret it as an FCT statement a set <varname> must be placed before the getrawvalue function.
				3- to interpret it as a TLM statement the getrawvalue must be inside a Tcl control structure (while or If)
				4- No double quotes for the parameter name.
					set <varname> [getrawvalue [fetch <par-name>]]
					while(getrawvalue[fetch <par-name>]) …
					if{getrawvalue[fetch <par-name>]}{

getrawvalidity [fetch <par-name>]	**FCT statement**		<par-name> STRING ALS	1- A set varname must be placed before the getrawvalidity function. 2- No double quotes for the parameter name.	set <varname> [getrawvalidity [fetch <par-name>]]
getengvalue [fetch <par-name>]	**TLM statement/FCT statement**	1- In case of check against a value, it will implemented as a TLM statement. 2- In case of a set it will be implemented as a FCT statement	<par-name> STRING ALS	1- getengvalue must have a fetch argument. 2- to intepret it as an FCT statement a set <varname> must be placed before the getengvalue function. 3- to interpret it as a TLM statement the getengvalue must be inside a Tcl control structure (while or If) 4- No double quotes for the parameter name.	1- set <varname> [getengvalue [fetch <par-name>]] 2- while(getengvalue[fetch <par-name>]) … if{getengvalue[fetch <par-name>]}{
getengvalidity [fetch <par-name>]	**FCT statement**		<par-name> STRING ALS	1- A set varname must be placed before the getengvalidity function. 2- No double quotes for the parameter name.	set <varname> [getengvalidity [fetch <par-name>]]
getdefaultvalue [fetch <par-name>]	**FCT statement**		<par-name> STRING ALS	1- A set varname must be placed before the function. 2- No double quotes for the parameter name.	set <varname> [getdefaultvalue [fetch <par-name>]]
getdefaultvalidity [fetch <par-name>]	**FCT statement**		<par-name> STRING ALS	1- A set varname must be placed before the function 2- No double quotes for the parameter name.	set <varname> [getdefaultvalidity [fetch <par-name>]]
getextractedvalue [fetch <par-name>]	**FCT statement**		<par-name> STRING ALS	1- A set varname must be placed before the function 2- No double quotes for the parameter name.	set <varname> [getextractedvalue [fetch <par-name>]]
getsccstate [fetch <par-name>]	**FCT statement**		<par-name> STRING ALS	1- A set varname must be placed before the function 2- No double quotes for the parameter name.	set <varname> [getsccstate [fetch <par-name>]]

Tcl/TOPE statements	MOIS implementation	Mechanism	Arguments	Rules	Recognized formulation
getoolstate [fetch <par-name>]	**FCT statement**		<par-name> STRING ALS	1- A set varname must be placed before the function 2- No double quotes for the parameter name.	set <varname> [gettoolstate [fetch <par-name>]]
gettimestamp [fetch <par-name>]	**FCT statement**		<par-name> STRING ALS	1- A set varname must be placed before the function 2- No double quotes for the parameter name.	set <varname> [gettimestamp [fetch <par-name>]]
getrequestid <vval>	**FCT statement**	<vval> must be only the name of a variable which is used in the referby of TOPE tcsend statement.	<vval> STRING ALS	1- vval must not be double quoted. 2- A set varname must be placed before the function	set <varname> [getrequestid <vval>]
getstage <vval>	**FCT statement**	<vval> must be only the name of a variable which is used in the referby of TOPE tcsend statement.	<vval> STRING ALS	1- vval must not be double quoted. 2- A set varname must be placed before the function	set <varname> [getstage <vval>]
getstatus <vval>	**FCT statement**	<vval> must be only the name of a variable which is used in the referby of TOPE tcsend statement.	<vval> STRING ALS	1- vval must not be double quoted. 2- A set var must be placed before the function	set <varname> [getstatus <vval>]
getstagehistory <vval>	**FCT statement**	<vval> must be only the name of a variable which is used in the referby of TOPE tcsend statement.	<vval>STRING ALS	1- vval must not be double quoted 2- A set var must be placed before the function.	set <varname> [getstagehistory <vval>]
getcompleted <vval>	**FCT statement/CTL statement**	This command is different from other because it can be implemented either as a CTL statement or FCT statement.	<vval> STRING ALS	1- vval must not be double quoted. 2- A set var must be placed before the 3- To be implemented as a CTL statement the getcompleted command must be included in a waitfor command as defined in recognized formulation (section 4.1.2).	1- set <varname> [getcompleted <vval>] 2- tcsend <command-name> referby <varname> waitfor <varname> -until {getcompleted $<varname>}
getshared <name>	**FCT statement**		<name> STRING ALS	1- <name> must not be double quoted 2- A set varname must be placed before the function	set <varname> [getshared <name>]
getupdatetime <vval>	**FCT statement**	<vval> must be only the name of a variable which is used in the referby of TOPE tcsend statement.	<vval> STRING ALS	1- vval must not be double quoted. 2- A set var must be placed before the function.	set <varname> [getupdatetime <vval>]
gets <channelId> ?<varname>?	**DIR statement**		<channelId> INTEGER DEC <varname> STRING ALS		
global <varname> ?<varname> …?	**n/a**				
glob <swicthes> ?<varname> …?	**FCT statement**		<switches> STRING ALS <varname> STRING ALS	Number of varname arguments is limited to 10	set <varname> [glob <switches> ?<varname> … <varname>?]
hextobin <hex-string>	**FCT statement**		<hex-string>STRING STR	<hex-string> must be double quoted	set <varname> [hextobin <hex-string>]
incr <varname> ?<increment>?	**DIR statement**		<varname> STRING ALS <increment> INTEGER DEC		incr <varname> ?<increment>?

Tcl/TOPE statements	MOIS implementation	Mechanism	Arguments	Rules	Recognized formulation
info	**n/a**				
inhibitgroup <grpid>	**DIR statement**		<grpid> STRING ALS		inhibitgroup <grpid>
inhibitparam <param-list>	**DIR statement**		<param-list> STRIING ALS	1- Each parameter of the list must be separated by a space character. 2- The list must have braces as delimiters.	inhibitparam {<param-list>}
inhibitpacket <spid>	**DIR statement**		<spid> STRING ALS		inhibitpacket <spid>
isdefaultvaluevalid [fetch <par-name>]	**FCT statement**		<par-name> STRING ALS	1- A set varname must be placed before the function 2- No double quotes for the parameter name.	set <varname> [isdefaultvaluevalid [fetch <par-name>]]
isengvaluevalid [fetch <par-name>]	**FCT statement**		<par-name> STRING ALS	1- A set varname must be placed before the function t 2- No double quotes for the parameter name.	set <varname> [isengvaluevalid [fetch <par-name>]]
israwvaluevalid [fetch <par-name>]	**FCT statement**		<par-name> STRING ALS	1- A set varname must be placed before the function 2- No double quotes for the parameter name.	set <varname> [israwvaluevalid [fetch <par-name>]]
join <list> ?<joinString>?	**FCT statement**		<list> STRING ALS <joinString> STRING ALS	List must be delimited by braces.	set <varname> [join {<list>} joinString]
lappend <varName> ?<value> <value> …?	**DIR statement**		<varName> STRING ALS <value> STRING ALS	Number of values is limited to10	lappend <varName> ?<value> … <value>?
lindex <list> <index>	**FCT statement**		<list> STRING ALS <index> STRING ALS		set <varname>[lindex <list> <index>]
linsert <list> <index> <element> ?<element> …?	**DIR statement**		<list> STRING ALS <index> STRING ALS <element> STRING ALS	1- Number of elements is limited to10.	linsert <list> <index> <element> ?<element> … <element>?
list ?<arg> <arg> …?	**FCT statement**		<arg> STRING ALS	1- Number of arguments is limited to10	set <varname> [list ?<arg> … <arg>?]
llength <list>	**FCT statement**		<list> STRING ALS		set <varname> [llength <list>]
lockshared ?-timeout <time>? <name>	**FCT statement**		<time> STRING ALS <name> STRING ALS	1- <name> and <time> must not be double quoted 2- A set varname must be placed before the function	set <varname> [unlockedshared ?-timeout <time>? <name>]
lrange <list> <first> <last>	**FCT statement**		<list> STRING ALS <first> STRING ALS <last> STRING ALS		set <varname> [lrange <list> <first> <last>]
lreplace <list> <first> <last> ?<element> …?	**FCT statement**		<list> STRING ALS <first> STRING ALS <last> STRING ALS <element> STRING ALS		set <varname> [lreplace <list> <first> <last> ?<element>…<element>?]

Tcl/TOPE statements	MOIS implementation	Mechanism	Arguments	Rules	Recognized formulation
lsearch ?<mode>? <list> <pattern>	**FCT statement**		<mode> STRING ALS <list> STRING ALS <pattern> STRING ALS	1- Mode is a required argument	set <varname> [lsearch <mode> <list> <pattern>]
lsort ?<options>? <list>	**FCT statement**		<options> STRING ALS <list> STRING ALS		set <varname> [lsort ?<options>? <list>]
memory	**n/a**				
namespace	**n/a**				
nametospid <pktname>	**FCT statement**		<pktname> STRING STR	1-Double quotes must be present around the packet name. 2- A set varname must be placed before the function	set <varname> [namespid <pktname>]
newtmdumpfile <vcid> ?<dumpname>?	**DIR statement**		<vcid> INTEGER ALS <dumpname> STIRNG ALS **Optional**		newtmdumpfile <vcid> ?<dumpname>?
open <filename> open <filename> <access> open <filename> <access> <permissions>	**FCT statement except the last format.**		<filename> STRING ALS <access> STRING ALS <permissions> STRING ALS	1- If there are no <access> parameters then <permissions> is ignored (i.e. the open <filename> <access> formulation is not supported) .	set <varname> [open <filename> ?<access>? ?<permissions>?]
patchlocation <param-name> <spid> <byteoffset> <bitoffset>	**DIR statement**		<param-name> STRING ALS <spid> INTEGER DEC <byteoffset> INTEGER DEC <bitoffset> INTEGER DEC	1- Each argument must be separated by a space character. 2- No double quotes can be supplied for any argument.	patchlocation <param-name> <spid> <byteoffset> <bitoffset>
patchscript <param-name> <script>	**DIR statement**		<param-name> STRING ALS <script> STRING STR	1- <Script> must be double quoted and <param-name> not. 2- Each argument must be separated by a space.	patchscript <param-name> <script>
patchnumericalcalcurve <calibcurveid> <pointid> <newXval> <newYval>	**DIR statement**		<calibcurveid> INTEGER DEC <pointid> INTEGER DEC <newXVal> STRING ALS <newYVal> STRING ALS	1- Each argument must be separated by a space character.	patchnumericalcalcurve <calibcurveid> <pointid> <newXval> <newYval>
patchtextualcurve <calibcurveid> <pointid> <newfrom> <newto> <newtext>	**DIR statement**		<calibcurveid>INTEGER DEC <pointid> INTEGER DEC <newfrom> INTEGER DEC <newto> INTEGER DEC <newtext> STRING STR	1- Each argument must be separated by a space character. 2- <newtext> argument must be double quoted.	patchtextualcurve <calibcurveid> <pointid> <newfrom> <newto> <newtext>
patchpolynomialcurve <calibcurveid> <coefficientid> <newcoeffvalue>	**DIR statement**		<calibcurveid> INTEGER DEC <coefficientid> INTEGER DEC <newcoeffvalue> REAL DEC	1- Each argument must be separated by a space character.	patchpolynomialcurve <calibcurveid> <coefficientid> <newcoeffvalue>

Tcl/TOPE statements	MOIS implementation	Mechanism	Arguments	Rules	Recognized formulation
patchcurveused <param-name> <pos> <newcurveid>	**DIR statement**		<param-name> STRING ALS <pos> STRING ALS <newcurveid> STRING ALS	1- Each argument must be separated by a space character. 2- The position must be supplied every time to be properly implemented in the reverse generation.	patchcurveused <param-name> <pos> <newcurveid>
patchlimit <param-name> <type> <pos> <lowvalue> ?<highvalue>?	**DIR statement**		<param-name> STRING ALS <type> STRING ALS <pos> STRING ALS <lowvalue> STRING ALS <highvalue> STRING ALS **Optional**	Each argument must be separated by a space character.	patchlimit <param-name> <type> <pos> <lowvalue> ?<highvalue>?
pid ?<fileId>?	**FCT statement**		<fileId> STRING ALS		set <varname> [pid <fileId>]
prompt ?<type>? <message>	**DIR statement**		<type> STRING ALS <message> STRING STR	1- <message> must be double quoted. 2- Each argument must be delimited by a space character.	prompt ?<type>? <message>
putlog ?-error? ?-warn? ?--? <expr>	**DIR statement**		<expr> STRING STR	1- <expr> must be double quoted. 2- Arguments must be separated by a space character.	putlog ?-error? ?-warn? ?--? <expr>
puts ?-nonewline? ?<channelId>? <string>	**DIR statement**		<channelId> STRING ALS <string> STRING ALS	Only 1 optional argument is allowed	puts ?-nonewline? ?<channelId>? <string>
pwd	**FCT statement**				set <varname> [pwd]
read ?-nonewline? <channelId> read <channelId> <numChars>	**DIR statement**		<channelId> STRING ALS <numChars> INTEGER DEC		1- read ?-nonewline? <channelId> 2- read <channelId> <numChars>
regexp ?<switches>? <exp> <string> ?<matchVar>? ?<subMatchVar> <subMatchVar> …?	**FCT statement**		<switches> STRING ALS <exp> STRING ALS <string> STRING ALS <matchvar> STRING ALS <submatchvar> STRING ALS	The number of <submatchvar> arguments is limited to 4.	set <varname> [regexp <switches> <exp> <string> <matchvar> ?<submatchvar> … <submatchvar>?]
regsub ?<switches>? <exp> <string> <subSpec> <varName>	**FCT statement**		<switches> STRING ALS <exp> STRING ALS <string> STRING ALS <subSpec> STRING ALS <varName> STRING ALS		set <varname> [regsub ?<switches>? <exp> <string> <subSpec> <varName>]
rename	**n/a**				
scan <string> <format> ?<varName> <varName> …? (normal) scan <string> <format> (inline form)	**DIR statement (normal form)** **FCT statement (inline form)**		<string> STRING STR <format> STRING ALS <varname> STIRNG ALS	The number of <varname> arguments is limited to 5.	1- set <varname> [scan <string> <format> ?<varname> … <varname>?] 2- scan <string> <format>

Tcl/TOPE statements	MOIS implementation	Mechanism	Arguments	Rules	Recognized formulation
seek <channelId> <offset> ?<origin>?	**DIR statement**		<channelId> STRING ALS <offset> INTEGER DEC <origin> STRING ALS		seek <channelId> <offset> ?<origin>?
set <varname> ?<value>?	**SET statement**				
setparameter ?-raw? <param-name> <value>	**DIR statement**		raw **Optional** STIRNG ALS <param-name> STRING ALS <value> STRING ALS	<param-name> must be double quoted	setparameter ?-raw? <param-name> <value>
setshared <name> <value>	**DIR statement**		<name> STRING ALS <value> STRING AL:S	<name> must not be double quoted	setshared <name> <value>
setrevision <expr>	**DIR statement**		<expr> STRING ALS	<expr> must be delimited by braces	setrevision {<expr>}
socket	**n/a**				
source	**n/a**				
spidtoname <spid>	**FCT statement**		<spid> STRING ALS	1-No Double quotes must be present around the spid. 2- A set varname must be placed before the function	set <varname> [spidtoname <spid>]
split <string> ?<splitchars>?	**FCT statement**		<string> STRING STR <splitcahrs> STRING STR		set <varname> [split <string> ?<splitchars>?]
string <option> <arg> ?<arg> …?	**FCT statement**		<option> STRING ALS <arg> STRING ALS	The number of arguments is limited to 10.	set <varname> [string <option> <arg> ?<arg> … <arg>?]
subst ?-nobackslashes? ?-nocommands? ?-novariables? <string>	**FCT statement**		<string> STRING ALS		set <varname>[subst ?-nobackslashes? ?-nocommands? ?-novariables? <string>]
subscribe <par-name> referby <var>	**DIR statement**		<par-name> STRING ALS <var> STRING ALS	No double quotes must be present around the parameter name.	subscribe <param-name> referby <var>
subcribeset {<par-list>} referby <var>	**DIR statement**		<par-name> STRING ALS <var> STRING ALS	1- No double quotes must be present around the parameter list. 2- Parameters in the <param-list> must be separated by a space	subscribeset {<param-list>} referby <var>
subscribepacket <spid> referby <var>	**DIR statement/PKT statement**		<spid> STRING ALS <var> STRING ALS	1- The first formulation will be intepreted as a directive (DIR) statement. 2- The second formulation will be interpreted as a wait for packet (PKT) statement. This second formulation must consist of 3 lines in the Tcl/Tope script (see section 4.2)	1- subscribepacket <spid> referby <var> 2- subscribepacket <pktid> referby <varname> waitfor -timeout <time><varname> unsubscribepacket <pktid>
suspend	**DIR statement**		n/a		suspend
syslog ?-error? ?-warn? ?--? <expr>	**DIR statement**		<expr> STRING STR	1- <expr> must be double quoted. 2- Arguments must be separated by a space character.	syslog ?-error? ?-warn? ?--? <expr>

Tcl/TOPE statements	MOIS implementation	Mechanism	Arguments	Rules	Recognized formulation
tcsend <command-name> ?referby <varname>? ?<options>…??<Parameterise>…?	**CMD statement/FCT statement**	Either tcsend will be implemented as CMD statement or it will be implemented as FCT statement. The following options will not be supported by CMD statement: checks, ackflags, id, parameterize, patch, option1, option2. One of these options make it be reversed as FCT statement.	<command-name> STRING ALS <varname> STRING ALS <releasetime> STRING ALS <executiontime> STRING ALS <checks> STRING ALS <ackflags> STRING ALS <id> INTEGER DEC <parameterise> STRING ALS <patch> STRING ALS <option1> STRING ALS <option2> STRING ALS	See section 4.1	1- tcsend <command-name> ?referby <varname>? …..
tell <channelId>	**FCT statement**		<channelId> STRING ALS		set <varname> [tell <channelId>]
tellsequence <id> <action>	**DIR statement**		<id> STRING ALS <action> STRING STR	1- <id> must not be double quoted. 2- <action> must be double quoted. 3- Each argument must be separate by a space character.	tellsequence <id> <action>
time	**n/a**				
trace	**n/a**				
unknown	**n/a**				
unlockshared <name>	**DIR statement**		<name> STRING ALS	1- <name> must not be double quoted to be recognized.	unlockshared <name>
upvar	**n/a**				
unset	**n/a**				
unsubscribe <par-name>	**DIR statement**		<par-name> STRING ALIAS	1- No double quotes must be present arount the parameter list. 2- Only one parameter name will be supported. Parameter list handling is not supported	unsubscribe <par-name>
unsubscribe -all	**DIR statement**		n/a		unsubscribe -all
unsubscribepacket <spid>	**DIR statement**		<spid> STRING ALS		unsubscribepacket <spid>
verified ?-timestamp <time>? -tc <cmd> ?<param>…? verified ?-timestamp <time>? -tm <parameter>	**DIR statement**		<time> STRING ALS **Optional** -tc and -tm switches <cmd> STRING ALS <param> STRING ALS <parameter> STRING ALS	1- For a telecommand, the number of parameters is limited to 10 2- No double quotes must be used for arguments. 3- Each argument must be delimited by space characters.	1- verified ?-timestamp <time>? -tc <cmd> ?<param>…? 2- verified ?-timestamp <time>? -tm <parameter>
variable	**n/a**				
waittime <time>	**CTL/PSE Statement**		n/a		waittime <time>

Tcl/TOPE statements	MOIS implementation	Mechanism	Arguments	Rules	Recognized formulation
waitfor ?-timeout <time>? <var-list> -until ?<condition>? Waitfor ?-timeout <time>? -all <var-list>	**DIR statement/CTL statement**	In case of use with tcsend command, the waitfor can be implemented as a CTL statement.	<time> STRING ALS <var-list> STRING ALS <condition> STRING ALS	1- <var-list> Each var defined in the list must be delimeted by a space character and the list must be delimited by square brackets. 2- Condition must be delimited by braces. 3- Each argument must be separated by a space character. 4- Note the particular case of tcsend + waitfor as described in section 4.1.2	waitfor ?-timeout <time>? <var-list> -until ?<condition>? Waitfor ?-timeout <time>? -all <var-list>