APPROVAL SHEET

TITLE :  
TCS Server SCL Syntax Specification

DOCUMENT NUMBER :  
1741AS0005 ISSUE: 2

SYNOPSIS :  
This document describes the SALT Command Language syntax.

KEYWORDS :  

PREPARED BY :  
Janus Brink

APPROVED :  
Gerhard Swart
SALT SYSTEMS ENGINEER

DATE :  
November 2003
This issue is only valid when the above signatures are present.
Printed: 21/11/03 13:42
**ACRONYMS AND ABBREVIATIONS**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BNF</td>
<td>Backus-Naur Form</td>
</tr>
<tr>
<td>ABNF</td>
<td>Augmented BNF (as defined in RFC 2234)</td>
</tr>
<tr>
<td>SALT</td>
<td>Southern African Large Telescope</td>
</tr>
<tr>
<td>SCL</td>
<td>SALT Command Language</td>
</tr>
<tr>
<td>SOMMI</td>
<td>SALT Operator Man-Machine Interface</td>
</tr>
</tbody>
</table>

**TABLE OF CONTENTS**

1. Scope ........................................................................................................................................... 4
2. Referenced Documents .................................................................................................................. 4
3. Context .......................................................................................................................................... 4
4. Basic SCL syntax ........................................................................................................................... 4
   4.1 Subsystem name ....................................................................................................................... 4
   4.2 Parameter name ....................................................................................................................... 5
   4.3 Parameter value ....................................................................................................................... 5
   4.4 Flag .......................................................................................................................................... 5
5. Implementation specific notes ...................................................................................................... 6
   5.1 Command parsing ..................................................................................................................... 6
   5.2 Script parsing ............................................................................................................................ 6
6. Examples ......................................................................................................................................... 7
7. Detailed SCL Syntax Definition .................................................................................................... 9
1 Scope

This document describes the SALT Command Language (SCL) syntax to be used in the interface from SOMMI, SAMMMI and the various instrument computers to the TCS Server.

2 Referenced Documents

The following documents are referenced in this specification.

<table>
<thead>
<tr>
<th>Doc No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1741AE0004</td>
<td>TCS Server Software Design Document</td>
</tr>
</tbody>
</table>

3 Context

The SALT Command Language is used to transfer commands from the SOMMI to the TSC Server in the form of text-based messages. An SCL command is contained within a single line of text and multiple commands can be grouped as a script. Commands are separated with the new-line character (\n). Comments can be included in a script by preceding the commented line with the ‘%’ character (comments are not allowed on the same line as commands).

The basic structure of an SCL command is described below; some examples are given in Section 6 followed by a detailed ABNF syntax definition in Section 7.

4 Basic SCL syntax

The high-level syntax of an SCL command is as follows:

<subsystem name>.<parameter name> = <parameter value>[&flag]

where the delimiters (‘.’ and ‘=’) and items enclosed in ‘<’ are mandatory, while items enclosed in ‘[ ]’ are optional. Any leading or trailing white space around keywords and parameters are ignored.

Each of these high-level tokens is discussed in more detail below.

4.1 Subsystem name

The subsystem name must match one of the defined subsystem names. This token is not case sensitive and all white space is ignored when finding a match. The valid subsystem names are derived directly from the defined ICD TCS control clusters as follows:

Given a control cluster named “TCS xxx.ctl”, the subsystem name is defined to be “xxx”.

Examples from the ICD of 12.02.2003 are:
- bms control
- dome control
- guidance control
- payload control
- pmas align control
- pmas control info
- structure control
- tracker control
• trajectory control

4.2 Parameter name

The parameter name must match one of the defined parameters for the specified subsystem. This token is not case sensitive and all white space is ignored when finding a match. The defined parameter names for a subsystem are directly derived from element names of the TCS subsystem control ICD clusters.

Examples (including subsystem names) from the ICD of 12.02.2003 are:
- dome control.dome mode
- dome control.authorise movement
- dome control.shutter command
- dome control.dome az angle
- structure control.structure mode
- structure control.structure az angle
- structure control.enable mcp
- trajectory control.x
- trajectory control.phi
- trajectory control.T0
- trajectory control.deltaT

4.3 Parameter value

The parameter value is defined as one of the following types:

- **Numeric**: A numeric value, which may be a floating point value. A numeric may not contain white space.
- **Boolean**: A boolean value written as “true” or “false”. A boolean is treated non-case-sensitive and may not contain white space.
- **String**: A string constant. String constants are not modified in any way – they are treated case sensitively and white space contained within the constant is left intact.
- **Enum**: Defines the value of an enumerated type (named numeric in LabVIEW) in string form. The string must match one of the defined enum values. Enum values are not case sensitive and all white space is ignored when finding a match.
- **Array**: The values in an array can be set by:
  - listing all the element values of a 1-dimensional array,
  - listing the element values of selected elements in an n-dimensional array,
  - setting the elements in a subset of an n-dimensional array to the same value.
- **Variable**: A variable name is given from which the parameter value is derived at run-time.

4.4 Flag

The flag parameter is optional and is used to denote one of the following:

- **Wait (W [duration])**: Waits until execution of the previous commands to this subsystem is complete before proceeding with the current command. If followed by a number, this represents the number of seconds to wait after finishing the previous commands.
- **Delay (T <duration>)**: Do not implement this command until the specified number of seconds has passed since issuing this command.
- **Abort (A)**: Abort all unfinished commands to this subsystem and implement this one.
- **Priority (P)**: Priority commands hare handled by a parallel processing engine and are passed to the subsystem controller verbatim as soon as they are received.

This token is not case sensitive.
5 Implementation specific notes

5.1 Command parsing

1. The variable parameter value type is only to be used in the SOMMI. All variables are pre-compiled to absolute values before passing SCL commands to the TCSS.
2. Ideally commented lines should be stripped from the SCL script by the SOMMI before transmission to the TCSS.
3. Arrays:
   a. Arrays of up-to 3 dimensions are supported.
   b. Matrix algebra conventions are used when specifying n-dimensional array indices: row | column | page (as also defined in LabVIEW).
   c. When an array is equated to a single parameter value (of the correct type) each dimension of the array is set to a size of 1 and the single element set to the parameter value.
   d. When an array is equated with an empty parameter value the array is cleared (i.e. dimension size(s) set to zero).
   e. Setting array elements (or a sub-array) at indices larger than the current array size will cause the array to be re-sized (retaining the current information).
   f. Resizing an array of strings to a single element implies the given string constant may not contain array operators e.g. ":", "[" or "]".
4. Only strings and arrays can be set to an empty parameter value.

5.2 Script parsing

When a list of commands are combined to form an SCL script the following should be taken into account: Commands to the same subsystem are grouped into a single cluster command before transmission to the subsystem. This implies that the commands are not necessarily executed in the order they appear in the script. Use the w or T flag to enforce specific execution order. Commands preceding and including the line where the flag appears are grouped and transmitted together.

For instance the commands

```
dome control.dome mode = ready
dome control.dome az angle = 10.0
dome control.dome shutter command = open
```

are grouped as a single cluster command to the subsystem. If the shutter is to be opened after the dome has rotated, the script should be adapted as follows:

```
dome control.dome mode = ready
dome control.dome az angle = 10.0 &w
    dome control.dome shutter command = open
```

This will split the script into two cluster commands, waiting for the dome rotation to complete before executing the shutter open command.
6 Examples

Some examples of valid SCL commands are shown below:

Setting a numeric value:
```
Dome control.dome az angle = 12.3
Domecontrol.domeazangle=1.23e1
Domecontrol.domeazangle=1.23e+1
DomeControl.DomeAzAngle = 123e-1
```

Setting a boolean value:
```
DomeControl.EnableMCP=true
Dome control.enable mcp = false
```

Setting a string value:
```
subsystem.stringval = hello world
subsystem.STRINGVAL = hello world
subsystem.stringval = 0:1;foo   -> treated as the string “0:1;foo”
subsystem.stringval = 0|5;bar   -> treated as the string “0|5;bar”
subsystem.stringval = 0,1,2,3,4,5 -> treated as the string “0,1,2,3,4,5”
subsystem.stringval =           -> treated as the empty string
```

Setting an enumerated value:
```
dome control.dome mode = ready
DOME CONTROL. domemode     = READY
```

Setting 1-dimensional array values:
```
subsystem.array = 1,2,3,4.5
subsystem.array = true, true, false, TRUE
subsystem.array = hello, world
subsystem.array = OFF, off, ready
```

Setting a subset of values of an n-dimensional array:
```
subsystem.array1d = 0:5 ; 1.0
subsystem.array2d = 0:5 | 0:3 ; 1.0
subsystem.array3d = 1:5 | 2:3 | 1:3 ; 1.0
```

Setting the values of specific elements in an n-dimensional array:
```
subsystem.array1d = 1; true
subsystem.array1d = 1;true, 5;false
subsystem.array3d = 1|2|2;false, 12|15|35;true
```

Resizing and clearing an n-dimensional array:
```
subsystem.array3d = 1.0          -> results in a 1x1x1 array.
subsystem.array =              -> results in a 0x0x0 array.
subsystem.array3d = 5|5|5;hello -> updates element 5,5,5 to “hello”.
  Array operators take precedence over
  a string value of “5|5|5;hello”. If the
  array was smaller than 6x6x6 it will be resized.
```

Using flags:
```
dome control.dome mode = ready &w
Dome control.dome mode = ready &w 10
Dome control .dome mode = ready &T10
Dome control.dome mode = off &a
```

Using variables (defined in SOMMI):
```
dome control.dome mode = <current mode>
```
DomeControl.DomeMode = <CurrentMode>
subsystem.arrayId = <a>, 15, <b>
7 Detailed SCL Syntax Definition

This section provides an ABNF definition of the SALT Command Language syntax:

script-line = comment / scl-command LF

comment = "WSP "%" *ANYCHAR

scl-command = subsystem-name "." parameter-name "=" parameter-value ["&" flag]

subsystem-name = KEYWORD

parameter-name = KEYWORD

parameter-value = value / array

flag = ("w" [duration]) / ("t" duration) / "a" / "p"

duration = numeric

value = absvalue / variable

absvalue = numeric / enum / boolean / string

variable = "<" KEYWORD ">"

numeric = float [exponent]

float = ["+" / "]" 1*DIGIT ["." 1*DIGIT]

exponent = ["e" ["+" / "]" 1*DIGIT]

enum = KEYWORD

boolean = "true" / "false"

string = *(ECHAR / DIGIT)

array = 1d-array / elements / sub-array / single-value / "" ; note can be empty

1d-array = value *("," value)

elements = element *("," element)

element = index *2( "]" index ";" value

sub-array = index "." index *2( ["l" index ";" index) ";" value

index = 1*DIGIT
single-value = numeric / enum / boolean / rstring / variable

rstring = *(RECHAR / DIGIT)

KEYWORD = 1*(ECHAR / DIGIT)

ANYCHAR = ECHAR / “&” / “<” / “>” / DIGIT / WSP

; extended character – note “&”, “<” and “>” are excluded
ECHAR = RECHAR / “;” / “:” / “,” / “|”

; restricted extended character – array operators are excluded

ALPHA = (“a” – “z”) / (“A” – “Z”)

WSP = %x20 ; white space

DIGIT = “0” – “9”

LF = %x0A ; line-feed character