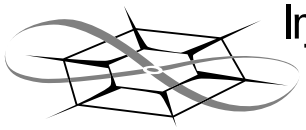
The University of Kansas



Information and Telecommunication Technology Center

Technical Report

ACE Project Service Command Language Specifications Version 1.0

Renzo Hayashi, Leon Searl, and Gary Minden

ITTC-FY2001-TR-23150-04

July 2000

Project Sponsor:
U.S. Air Force and the Defense Advanced Research Projects Agency under contract no. F30602-00-2-0581
and The National Science Foundation
under grant EIA-9972843

Table of Contents

1.0 Introduction

This document shall fully describe and set ground rules for the syntax of the language we shall create for ACE device/service control. Syntax development, rules, and usage should follow these guidelines for the development of an all-encompassing language parser and control API's.

Such a language must be produced to give our development team the freedom to mold the ACE environment as is seen fit. This language, based on the known capabilities of the ACE world, is described below.

2.0 Specifications

2.1 - Syntax Specifications

The general syntax of our ACE device command language is as follows:

<CMND><space>[[<ARGNAME>]'='<ARGVALUE>]...';'

This structure shall allow us to assign a specific value to a given argument name (this can refer to a specific device or hardware component). The way in which this value is assigned to a specific device shall depend solely on the nature of the <CMND> portion of the syntax.

2.1.1 - Breaking it Down

Each sub-component of this language structure is as follows:

<CMND> := <CMNDNAME><space>[<ARGLIST>];

<CMNDNAME> := <WORD>;

<ARGLIST> := | <ARGUMENT> | <ARGUMENT><space><ARGLIST>

<ARGUMENT> := <ARGNAME>'='<ARGVALUE>;

<ARGNAME> := <WORD>;

<ARGVALUE> := <INTEGER> | <FLOAT> | <WORD> | <STRING> | <ARRAY>;

<ARRAY> := {<ARGVALUE LIST>};

<ARGVALUE LIST> := <ARGVALUE> | <ARGVALUE>','<ARGVALUE LIST> |
<ARRAY> | <ARRAY>','<ARRAY>;

<WORD> := <contiguous alphanumeric & underscore>;

<STRING> := <WORD> | "<contiguous printable characters>";

S-2.1.1.1 <ARGUMENT>'s are position independent. That is, there is no specific and correct order in which to specify arguments to the command. An example is given below:

"MoveTo x=0.0 y=1.0 z=0.5" is equivalent to "MoveTo z=0.5 x=0.0 y=1.0"

S-2.1.1.3 <STRING> values with space characters within them shall necessarily be enclosed within double-quotation marks. If the need to express double-quotation marks arises within this string, then a back-slash (\ ) shall precede this double-quotation mark in the string.

S-2.1.1.4 The syntax does not require <ARGUMENT>'s. That is, some device commands may not need any values given to them since they are self-explanatory in nature (e.g. reportStatus).

S-2.1.1.6 A single command line may have more than one argument name (further detailed in the ACE Project Service Command Language Parser Specifications document).

S-2.1.1.8 A single command line shall be terminated by a semicolon. These also serve as command line separators.

> Rationale: Since the SSL layer does not separate individual commands into unique packets, a separating character is necessary in order to distinguish between two or more commands.

S-2.1.1.9 The ACE service language shall be case insensitive to simplify the issuing of commands. Exception is S-2.1.1.14.

S-2.1.1.10 A single argument name may take one or more values. In order to specify more

than one value for a single argument, these values shall be separated by commas. Whether or not spaces come after or before these commas should not affect parsing.

S-2.1.1.12 If a command only needs a single argument, that argument name is optional. In the case of two or more needed arguments, the argument names are mandatory.

> Rationale: The command name itself should be self-explanatory as to the nature of the value needed for the command.

S-2.1.1.13 All command and argument names developed in this syntax should be full words or well known abbreviations.

S-2.1.1.14 <STRING>'s are case sensitive with the exception of <WORD>'s which shall be case insensitive.

2.2 - Semantic Specifications

S-2.2.1 <ARGNAME> and <ARGVALUE> must have same data type. For example, if <ARGNAME> is an argument that expects a value of type CMDVAL_INT, then <ARGVALUE> must also be of type CMDVAL_INT.

S-2.2.2 <CMND> must receive the appropriate arguments depending on the type of operation performed and the parameters it requires.

S-2.2.3 Some arguments specified may be mutually exclusive. That is, in specifying one argument the possibility of specifying another mutually exclusive one cannot be done.

S-2.2.4 It is possible for an <ARGNAME> to receive multiple values and store these into an array of <ARGVALUE>'s.

S-2.2.5 The parser shall recognize data types from a list of enumerated data types specific to the ACE environment.

S-2.2.6 The order of the arguments must be preserved after the command line is parsed.

> Rationale: This allowa the calling function to know in which order arguments were specified in the case when a single argument is given multiple values. See S-2.1.1.5

S-2.2.7 The return string of the parser function call (above) shall be a null value if no parsing errors are present in the input command line.

3.0 - Deprecated Specifications

S-2.1.1.2 A single command line (such as described above) shall be sent in a single packet. We do this so that we can keep communication and computation simple and so that there is no need to assemble multiple packets before transmission.

S-2.1.1.7 A single argument name may take on more than one value or a set of values.

S-2.1.1.11 A continuation character (\) can be used to split the command line into two or more rows of text.

S-2.1.1.5 The syntax does not require a value to be given to an argument. For instance, some commands may require the argument value to be returned, therefore, no value needs to be supplied. **(This is a duplicate specification)**

4.0 - Glossary

(TBS)

5.0 - Change Log

Version	Date	Changes
0.1	June 19, 2000	Original
0.2	July 5, 2000	Added commands to 2.1.2
0.3	July 20, 2000	Made corrections to command language and added more examples for each device.
1.0	July 22, 2000	Initial working version.

6.0 Notes

- Let's consider possibly nesting <CMND>'s within <ARGUMENT>'s so we may implement logic elements such as if-then-else statements in the following manner:
- if <predicate>=<CMND> <then CMND> <else CMND>