IT:
IDL TCL/TK Interface

Bob Daly
Argonne National Laboratory
Advanced Photon Source
Accelerator Systems Division/Controls Group
February 1995

1. Introduction

This document describes the `tcl` command and command types which are used to communicate with `IDL` via the IDL Remote Procedure Call Interface. IDL is a program for the analysis and visualization of scientific and engineering data. It is supplied by Research Systems, Inc., Boulder, Colorado. In addition to the `tcl` application libraries necessary to build `et`, the IDL RPC library, which is part of the IDL distribution is needed.

The reader of this document is assumed to be familiar with the `et` `tcl` implementation. See EPICS document *ET: EPICS TCL/TK Interface* by Bob Daly, for further information.

2. Running it

Prior to starting `it`, `idl` should be started in server mode using default settings:

```
idl -server
```

Start `it` by typing:

```
it_wish
```

In the extensions source directory for `it` (i.e. `extensions/src/tcl_it`) is a subdirectory `examples` containing examples using `it`. These examples together with the following description of the `idl` command types should be used when learning to use `it`.

3. Command Syntax

The command syntax consists of three or more words, with the first three words having the same meaning in each command. The words after the first three words are dependent on the type of command issued. A command example follows:

```
idl link ai1tcl ai1idl D 10000
```

- First word: the `tcl` `idl` command
- Second word: the **command type** i.e. `link`.
- Third word: **the name of a tcl variable** i.e. `ai1` or `ai1tcl` (command type **cmd** is exception)
- Other words: **depend on the type** i.e. for type `link` the fourth word consists of a name of an `idl` variable, the fifth word is the variable type, the sixth and last word is the number of elements in the variable.

Examples of all supported types:

```
idl link tcl1 idl1 S 512
idl link tcl3 idl3 D 1
idl linkpv wf wfidl
idl get tcl1
idl put ai1
idl cmd plot,idl1
idl cmd {plot,wf,color=12,yrange=[2000,3000]}
<<<<start multi-element (vector) record oriented types>>>>>>>>>
idl vdef tcl1 {0 256 5}
idl vset tcl1
idl vdis wf $graph line1
```

4. idl Command Types

link

Syntax:

```
idl link tclVar idlVar typeVar nelementsVar
```

Description: Establish a link between a `tcl` variable and an `idl` variable. The type of variable (`typeVar`) i.e Byte (`B`), Int (`I`), Long (`L`), Float (`F`), or Double (`D`) and the number of elements (`nelements`) of the defined type are also specified. Both the `tcl` and `idl` variable are created. For scalar variables the `tcl` variable can be set or read directly; however, for vector variables the ‘`vdef`’ and ‘`vset`’ command types are used. Strings are not yet supported. All `tcl` values used at the script level are treated as doubles.

Return:

- 0 successful
- 1 unsuccessful and error is described in `tcl` variable “`errorCode`”

Examples of correct forms of `link`(w):

```
idl link tclvar1 idlvar1 F 1024
idl link tclai1 idlai1 F 1
set tclai1 11.1
```

linkpv

Syntax:

```
idl linkpv tclVar idlVars
```

Description: Establish a link between `tcl` variable (which has previously been linked with an IOC process variable via a '`pv link`' command) and an `idl` variable. The type of variable i.e Byte (`B`), Int (`I`), Long (`L`), Float (`F`), or Double (`D`) and the number of elements are determined by the IOC process variable. Strings are not yet supported. All values used at the `tcl` script level are treated as doubles.

Return:

- 0 successful
- 1 unsuccessful and error is described in `tcl` variable "`errorCode`"

Examples of correct forms of `linkpv`:

```
pv linkw wf vong:xy566WaveformCh0
pv linkw ai1 T:ai1
idl linkpv wf wfidl
idl linkpv ai aiidl
```

cmd

Syntax:

```
idl cmd IDLcommand
```

Description: Execute the `idl` command.

Return:

- 0 successful
- 1 unsuccessful and error is described in `tcl` variable "`errorCode`"

Examples of correct forms of `cmd`:

```
idl cmd plot,wf
idl cmd {plot_io,freq,abs(fft(x,1)),title="TEST",ytitle="POWER"}
```

get

Syntax:

```
idl get tclvar
```

Description: Update the linked `tcl` variable to the current values stored in the `idl` variable.

Return:

- 0 successful
- 1 unsuccessful and error is described in `tcl` variable "`errorCode`"

Examples of correct forms of `get`(w):

```
pv linkw ai T:ai1
pv linkw wf vong:xy566WaveformCh0
idl link ai aitcl
idl link wf wfidl
idl get aitcl
idl get wf
```

put

Syntax:

```
idl put tclvar
```

Description: Update the value of the linked `idl` variable to the `tcl` variable value.

Return:

- 0 successful
- 1 unsuccessful and error is identified in `tcl` variable "`errorCode`"

Examples of correct forms of `put`:

```
pv linkw ai T:ai1
pv linkw wf vong:xy566WaveformCh0
idl link ai aitcl
idl link wf wfidl
idl put wf
idl put aitcl
set aitcl 11.1
idl put aitcl
```

vdef, vset, vdis

Syntax:

```
idl vdef tclVar {base size precision}
idl vset tclVar {list of values to be written to tclVars}
idl vdis tclVar graph   graph_element
```

Description: All three of these command types are used to deal with multi-element and waveform records. For multi-element and single element records, the `link`, `get`, and `put` command types operate the same. However, for multi-element process variables, the linked `tcl` variable only reflects the value of the first element. The user reads/writes/displays multi-element data stored in the interface buffer using a "`view`" mechanism. A view is a defined subset of a multi-element record which will subsequently be used for accessing data from the record.

- **VDEF** defines a view which specifies the base, size and number of significant digits. `VDEF` can be defined for each linked `tcl` variable.
- **VSET** is used either to return to `tcl` a list consisting of the values in the view with the number of significant digits defined by `VSET` or to write into the interface the values defined by the final word (normally a list of values) to `VSET` command type. The base and maximum number of values to be written are determined by the `VSET` command type.
- **VDIS** is used with the `BLT` graph widget to display the process variable view as an "`blt graph element`" on a "`blt graph`".

Return:

- 0 successful*
- 1 unsuccessful and error is identified in `tcl` variable "`errorCode`"

---

* When a list of values to be written is not included as the final word of the command, `VSET` returns a list of values

Examples of correct forms of `vdef`, `vset`, and `vdis`:

```
set graph .g
blt_graph $graph -width 500
$graph element create line1 -symbol circle -bg red
pack append . .g {}
pv link {wf1 wf} {vong:xy566WaveformCh0 vong:xy566WaveformCh1}
pv get {wf1 wf}
idl linkpv wf wfidl

idl vdef wf {0 256 5}
idl vset wf
idl vset wf {1 2 33 56.4 987.56 1.01}
idl vdis wf $graph line1
```