Software requirements for caTools

1 Introduction

caTools represent a set of tools used to read, write and display information about EPICS channels. Two sets of tools currently exist. First set is written in TCL programming language and have been in use at PSI for many years. Later a second set of tools was added to EPICS base. The basic functionalities of both are equal, but each has different additional features. Furthermore, the input and output of both tools is not compatible.

The new version of the TCL programming language is not compatible to the previous ones, so the existing PSI tools cannot be used with the new TCL version. Thus a new set of caTools is needed, that will be backwards compatible with the existing PSI tools and that will have additional features from EPICS base tool set added.

An analysis of the current tools is available inside a Jira ticket [1]. A preliminary list of command line arguments for the new tools is available inside a Jira ticket [2] as `prelimenaryArgs.pdf` attachment. The rest of this document describes the requirements for the new set of caTools.

2 Requirements

2.1 General

2.1.1. The following tools shall be available:

caget Reads and formats a value from each channel specified by the user.

cagets Writes 1 to PROC field of a channel and reads it back after channel has finished processing.

caput Writes a value to a channel, waits for the processing to finish and reads back the new value.

caputq Writes a value to a channel, but does not wait for the processing to finish. Does not have any output (except if an error occurs).

camon Monitors a set of channels and outputs their values on each change.

cado Writes 1 to a channel, but does not wait for the processing to finish. Does not have any output (except if an error occurs).

cawait Monitors a channel and waits until specified conditions for the channel match. Then the value is displayed.

cainfo Displays detailed information about a channel. Has custom output, so the formatting options do not apply.

2.1.2. caTools shall be written in C programming language.

2.1.3. caTools shall compile to run on all terminal stations in PSI. At least the following platforms shall be supported:

- SL6-x86 SL6-x86_64,
- SL5-x86 SL5-x86_64,
- eldk52-e500v2,
- eldk42-ppc4xxFP,
- Windows (exact build system is not yet determined).

2.1.4. Input and expected output shall be backwards compatible with existing PSI tools, except for **caInfo** tool and array handling (output will be without curly braces).

2.1.5. Date and time of the record processing shall be available (CA server date/time).

2.1.6. Date and time when the record data was received on the client side shall be available (client local date/time).

2.2 Channel access

2.2.1. The user shall be able to request specified DBR type (eg. `DBR_TIME_CHAR`).

2.2.2. Default requested DBR type shall be matching record's field DBF type:

- By default, `GR` detail level (`DBR_GR_[native DBF]`) shall be used.

- When only record processing time-stamp (CA server date/time) and/or severity and status are requested, `TIME` detail level shall be used (`DBR_TIME_[native DBF]`).
- When more detailed information than severity, status and record processing time-stamp (CA server date/time) is requested, two requests shall be made. First request shall be using `TIME` detail level and second using `GR` detail level.
- When less detailed information is requested, the detail level shall be set accordingly.

2.2.3. The user shall be able to specify channel access timeout (eg. how long to wait for channel to connect).

2.3 Tool specifics

2.3.1. **camon** shall have options to select incremental time-stamp:

- Time elapsed since start of program.
- Time elapsed since last update.
- Time elapsed since last update, by channel.

2.3.2. **camon** shall have an option to exit monitoring after user specified number of updates.

2.3.3. **cawait** shall have an option to exit after a user specified timeout.

2.3.4. **caInfo** tool shall display all the information available in `DBR_CTRL_XXX` DBR type, and more:

- native DBF type, array size, IOC name, access rights,
- record type, description, severities assigned to alarm limits.

2.4 Value formatting

2.4.1. The format of the `decimal` values shall be settable:

- Round the value to closest integer, round to next integer, round to previous integer.
- Scientific notation (mantissa/exponent) with specified precision - %e format.

- Decimal floating point notation with specified precision - %f format.
- Use the shortest representation between %e and %f.
- Override the PREC field defined in the record.

2.4.2. The user shall be able to format enum or char value to integer (and vice-versa).

2.4.3. The user shall be able to set the format of the `integer` values to hex, binary or octal.

2.4.4. The user shall be able to force interpretation of `enum` and `char` values as integers or characters / array of characters.

2.5 Array handling

2.5.1. Arrays shall be printed as a list of values, where the formatting shall be based on the record's field native DBF type (eg. char type is displayed as an ASCII character, double type is displayed as a decimal number). Default behavior can be overridden by requirement 2.4.4.

2.5.2. Array input and output values separator shall be configurable.

2.5.3. The user shall be able to set the requested amount of array elements to read or write.

2.5.4. The user shall be able to print the number of elements in an array before printing out array values.

2.6 Output formatting

2.6.1. The default output format shall be in the following form:

```
[date] [time] [channel name] [value] [EGU] [severity and status]
```

2.6.2. Severity and status shall be displayed when not in `NO_ALARM` state.

2.6.3. The user shall be able to force the severity and status to always be displayed.

2.6.4. The user shall be able to force the severity and status to never be displayed.

2.6.5. The user shall be able to hide channel name from the output.

2.6.6. The user shall be able to hide engineering units (EGU) from displaying next to values.

2.6.7. The value and output formatting shall be applied to all tools that honor the default output format equally.

2.7 Input formatting

2.7.1. The input format for **caget**, **cagets**, **camon** and **cado** shall be in the following form:

```
[tool] [arguments] [channel] [channel] ... [channel]
```

2.7.2. The input format for **cawait** shall be in the following form:

```
[cawait] [arguments] [[channel] [condition]]
                                  ...
                     [[channel] [condition]]
```

2.7.3. The input format for **caput**, **caputq** shall be in the following form:

```
[tool] [arguments] [[channel] [value]] ... [[channel] [value]]
```

2.7.4. The input format for **caInfo** shall be in the following form:

```
cainfo [channel] [channel] ... [channel]
```

References

[1] Sašo Skube. Analyze and document current tools - Jira issue. `https://tracker.psi.ch/jira/browse/COSYLAB-256`.

[2] Sašo Skube. Collect requirements - Jira issue. `https://tracker.psi.ch/jira/browse/COSYLAB-310`.