The CED Command-line Utility
Quick Start Guide

By Christopher Slominski

Version 2.0 – December 2015

Introduction

The CED command line utility provides LINUX shell users access to the wealth of information in the CEBAF Element Database. To execute the program users simply enter '**ced**' and some arguments to the shell prompt. A good start is to enter '**ced -h**'; which provides you with a list of command line help topics. The program's command line help has a multitude of topics, each related to one set of command line options, which I refer to as a command signature. When you ask for help on a specific topic (example: **ced -h inventory**), you get brief information about what is being requested as well as the options and arguments available with that command line signature.

Although there are many command line signatures for performing CED access operations, not all are discussed in this manual. This manual focuses on the most common use case; which is extracting CEBAF element information from the production CED database.

The CED command line program serves as a user interface to the CED API. It just exposes a code library to the command line user. This document gives a brief introduction for the new command line utility

user. To understand the many facets of the API, and therefore the command line utility, you must refer to the CED API documentation.

Concepts

There are three items users must understand to interact with the CED; element types, element type properties, and elements.

- Types - These are categories of the physical devices that exist in the CEBAF environment; such as BPMs and Dipoles.
- Properties - Element types have properties, such as length, serial number, and location.
- Elements - These are individual instances of an element type, having specific values assigned to the various element type properties.

Users must understand the concept of different CED deployments and workspaces within a deployment. Without any direction from the user, the application defaults to using the production (pro) deployment and its operational (ops) workspace. Various signatures include *–ced* and *–wrkspc* options to choose alternatives. Users will seldom have need to specify an alternative CED deployment; however it is common for users to specify an alternative workspace. Users that modify CED property values will do so in a workspace; which may later be promoted to be operational. Use the *–workspaces* signature to view the choices.

Exploring the CED

An element

Users often approach the CED with an element they wish to examine. They want to get the value of some of the element's properties. If they know the name of the property they want it is as simple as;

ced -get -e <element> -p <property>

If the user isn't sure of the property name, or wants to get values for a number of properties on an element, the inventory signature is used as follows. Note the first example gets all property values and the second gets just those explicitly requested.

ced -inventory -e <element>

ced -inventory -e <element> -p <property> <property> …

Related sets of elements

The inventory signature is used to get a set of elements from the CED. How these elements are related depends on the 'filters' provided on the command line. Filters restrict which elements are selected from the CED. When you provide no filters, and don't explicitly name the elements of interest as shown in the previous section; all elements are extracted from the CED. Filters limit the contents of your inventory by

element type, name pattern, geographic zone, or property values. Combinations of filters may be used to extract only the element set in which you are interested. The following example gets all dipole elements in the injector having their resistance property value greater than ten.

```
ced -inventory -t Dipole -a A_Injector -Ex 'resistance > 10'
```

As shown in the previous section, you may state the properties of interest; otherwise you will get all property values. Refer to command line help to get information about the filters available.

Specifying properties of interest

As seen above, the inventory signature accepts a *–p* option to specify properties of interest. When not provided, all properties are extracted for the elements in the inventory. You may list the properties of interest after the option; however providing the option with no named properties following means do not include any element properties in the inventory.

Property expressions

The required command line input format may be confusing in a couple of situations. The value in a property expression needs to be quoted for 'string' data types, since the API pastes the expression directly into SQL. The shell command line processor strips quotes, so double quoting is necessary. If you don't want to be aware of the property data types, you may use the double quoting for non-string types as well. A command above is repeated in this format as an example.

```
ced -inventory -p -t ioc -Ex "owner='kjeldsen'"
ced -inventory -t Dipole -a A_Injector -Ex "resistance > '10'"
```

When just the property name is provided for a property expression, the filter is on the existence of that property; only elements having a value assigned for that property are included in the inventory. Prefixing a '!' character negates the logic, meaning only elements that do not have a value for the property. The negation requires escaping because of the command shell parser.

```
ced -inventory -p -Ex powered_by
ced -inventory -p -Ex '\!powered_by'
```

Output formatting

You may choose between various inventory formatting options using both the '-f' and '-x' command line options. The commands below write an inventory to *stdout* in two different formats. In this case the '-f' option selects the "standard" or "aligned columns table" format and the '-x' options excludes type, count summary, and units. You could also suppy 'l' the excluded table attributes to get rid of the information labels.

```
ced -inventory -t BA -p TurnsPerMag HGap -f0 -x tcu
MBA1C05
  TurnsPerMag: 40
```

```
    HGAP: 0 meters
  MBA1C06
    TurnsPerMag: 40
    HGAP: 0 meters
  MBA1C07
    TurnsPerMag: 40
    HGAP: 0 meters
  MBA1C08
    TurnsPerMag: 40
    HGAP: 0 meters
  MBA1C09
    TurnsPerMag: 40
    HGAP: 0 meters
  MBA1C10
    TurnsPerMag: 40
    HGAP: 0 meters
  MBA1C11
    TurnsPerMag: 40
    HGAP: 0 meters
  MBA1C12
    TurnsPerMag: 40
    HGAP: 0 meters
```

```
ced -inventory -t BA -p TurnsPerMag HGap -f1 -x tcu
  Name  TurnsPerMag HGap
MBA1C05          40    0
MBA1C06          40    0
MBA1C07          40    0
MBA1C08          40    0
MBA1C09          40    0
MBA1C10          40    0
MBA1C11          40    0
MBA1C12          40    0
```

The '-f2' format is similar to the '-f1' format, in that it creates a table. However the output is more suitable for programmatic parsing than human reading. With this format, the columns of information are separated by a single tab character instead of a variable number of space characters; therefore columns won't necessarily align in a simple text display. The parsing advantage comes when an element does not have a value for a property. The single character delimiter makes identification of values and nulls easier.

Types of elements

Information about CED element types may be obtained using the command line utility. Element types are related hierarchically; with types inheriting properties from the types from which they are derived. You may view the CED element type tree as follows. The first shows the entire tree, which can be rather large. The second shows the branch starting at a specific element type, and stopping at a depth of some number of tree nodes.

ced -hierarchy

ced -hierarchy -b <type> -d <depth>

You get information about an element type with the catalog request. How much information you get about each property depends on one of the signature's command line arguments (-f). The following command gets full information about all properties of a specific element type.

ced -catalog -t <type> -f

Property value history

Users may see the history of an element's property value. This feature is only available when working with the operational workspace of a CED deployment.

ced -history -e <element> -p <property>

Inventory signature examples

The inventory signature is the most commonly used; however its complex set of options can be confusing. This section provides explanation and examples to help the application user.

Database wildcard characters

The application uses SQL database wildcard notation when appropriate. The '%' symbol means zero or more characters and the '_' mean any single character.

Property type exclusion

The simple type filter is the most commonly used way to limit the elements in an inventory. User may also request that particular types be excluded when specifying this filter. Use the '!' prefix on type names that should be excluded; remembering that mode command shells will required that the symbol be escaped. The behavior of exclusions needs to be explained, as they are only applied to the types that are requested to be included. Specifying a type filter with only an exclusion type will not provide all elements except the exclusion; but will provide no elements. The following example gets all IOC elements excluding the *SoftIOC* and *ExternIOC* sub-types.

ced -inventory -p -t IOC \!SoftIOC \!ExternIOC

The relational filter

Relational filters are a very powerful way to specify elements of interest, but their use is quit complex. The following examples show two uses that actually give the same result. The first usage indicates to include all BPM elements that are controlled by BPM crates that are controlled by the element *iocea2*. Note the element type and property are separated by a period. Due to BPM creates having properties (IFChannel<n>) that point back to the BPMs in the crate, the second usage may be used to get the desired result. It states to include all elements (not just BPMs) that are on any of the channel properties of BPM crates; which are controlled by the element *iocea2*.

ced -inventory -p -R 'BPM.Controlled_by>BPMCrates.Controlled_by>iocea2'
ced -inventory -p -R 'IFChannel%<BPMCrates.Controlled_by>iocea2'

The aggregate expression filter

This is the same as the more commonly used property expression filter, but allows the use of wildcard characters in the property name. Remember that the application uses SQL database wildcard characters described at the beginning of this section. The following example lists BPM crates that have the element *IPM2R01* on any of its channels.

ced -inventory -p -t BPMCrates -Ea 'IFChannel%=IPM2R01'

Element name filters

Element name filters are fairly straight forward, but users must remember to use SQL wildcard characters in glob style expressions. The first example below gets all elements whose names begin with "IPMSL" using a glob style name filter. The second example performs the same task using a regular expression style name filter.

ced -inventory -p -ng 'IPMSL%'
ced -inventory -p -nx '^IPMSL'

Sorting results

Inventory results may be sorted in a specified order. The *–s* option specifies the sort property. Note that the sort property must be scalar and be included in the set of properties to be included in the inventory. If the sort property is actually multi-pass, make sure to use the *–r* option to repeat each pass of the element in the inventory as individual elements.

ced -inventory -t BPM -p S -s S -r -f1

Specifying properties by category

The properties of interest may be specified by a property category. Note that property categories are in a hierarchy and the full path to the category must be specified using the ':' delimiter.

ced -inventory -t CryoCavity -C Main:LEM