
iocdoc Documentation

Release 0.0.2+0.gc458a41.dirty

Pete R. Jemian

Sep 27, 2017

Contents

1	Overview	1
2	Indices and tables	11
	Python Module Index	13

version 0.0.2+0.gc458a41.dirty

published Sep 27, 2017

IOCDoc: Document the configuration of an EPICS IOC

In EPICS, an IOC is the server, coordinating hardware actions with software configuration, command, and control.

This project arose from a user request: *What does my EPICS system provide? Tell me about all the PVs.*

Operation of an IOC begins with a command to a executable, compiled for the host computer architecture, and a startup script, often called `st.cmd` by convention. All necessary configuration information is provided in the startup script or through the environment variables from the host operating system when the startup script is launched.

It is possible, by parsing the startup script, to document the implementation of the IOC and that is the goal of this package. Reference to well-known IOC commands (such as `dbLoadRecords`), packages (such as `synApps`), and the chosen EPICS base version will be provided.

Output is in a set of restructured text files for compilation by Sphinx into HTML and/or PDF. Reference to any additional HTML documentation provided in the IOC's directory structure will be included.

Contents:

IOCDoc: Document the configuration of an EPICS IOC

iocdoc: Document the configuration of an EPICS IOC

docs <http://iocdoc.readthedocs.org>

git <https://github.com/prjemian/iocdoc/>

The user will provide the directory path (absolute or relative) to an EPICS IOC startup command file. The IOC startup file will specify the starting point to discover the configuration of each IOC.

This code will parse the startup command file and discover the commands and databases used to start the IOC. An attempt is made to read the database files. This program will report if a file or database is not found.

The goal is to generate a set of WWW pages with appropriate links to standard documentation for all the commands and other common support. Links will be made to any local documentation provided in the `./documentation` directory of the top level directory.

Techniques used in coding this package

Text file parsing relies on the Python `tokenize` package ...

Sphinx is used to prepare documentation ...

The `lxml` package ...

module: `command_file`

module: `database`

EPICS database file analysis

class `iocdoc.database.Database` (*parent, dbFileName, ref, **env*)

Bases: `object`

call for one EPICS database file with a given environment

makeProcessVariables ()

make the EPICS PVs from the record definitions

parse ()

interpret records for PV declarations

module: `ioc`

module: `macros`

support for macro substitutions

From the EPICS Application Developer's Guide

see <http://www.aps.anl.gov/epics/base/R3-14/12-docs/AppDevGuide/node7.html>

6.3.2 Unquoted Strings

In the summary section, some values are shown as quoted strings and some unquoted. The actual rule is that any string consisting of only the following characters does not have to be quoted unless it contains one of the above keywords:

```
a-z A-Z 0-9 _ -- : . [ ] < > ;
```

```
my regexp: [\w_\-:.\[\]<>;]+([\.[A-Z0-9])?
```

These are also the legal characters for process variable names. Thus in many cases quotes are not needed.

6.3.3 Quoted Strings

A quoted string can contain any ascii character except the quote character `"`. The quote character itself can be given by using `\"` as an escape. For example `\"\"` is a quoted string containing the single character `"`.

6.3.4 Macro Substitution

Macro substitutions are permitted inside quoted strings. Macro instances take the form:

```
$ (name)
```

or

```
${name}
```

There is no distinction between the use of parentheses or braces for delimiters, although the two must match for a given macro instance. The macro name can be made up from other macros, for example:

```
$(name_$(sel))
```

A macro instance can also provide a default value that is used when no macro with the given name is defined. The default value can be defined in terms of other macros if desired, but cannot contain any unescaped comma characters. The syntax for specifying a default value is as follows:

```
$(name=default)
```

Finally macro instances can also contain definitions of other macros, which can (temporarily) override any existing values for those macros but are in scope only for the duration of the expansion of this macro instance. These definitions consist of `name=value` sequences separated by commas, for example:

```
$(abcd=$(a) $(b) $(c) $(d) , a=A, b=B, c=C, d=D)
```

class `iocdoc.macros.KVpair` (*parent*, *key*, *value*, *ref=None*)

Bases: `object`

any *single* defined key:value pair in an EPICS IOC command file

- PV field
- Record field
- Macro
- Symbol

class `iocdoc.macros.Macros` (***env*)

Bases: `object`

manage a set of macros (keys, substitutions)

exists (*key*)

is there such a *key*?

get (*key*, *missing=None*)

find the *key* macro, if not found, return *missing*

items ()

get the full database, like `dictionary.items()`

keys ()

get the list of macros, like `dictionary.keys()`

replace (*text*)

Replace macro parameters in source string

set (*key*, *value*, *parent=None*, *ref=None*)

define the *key* macro

setMany (***env*)

define several macros

`iocdoc.macros.identifyEpicsMacros` (*source*)

Identify any EPICS macro substitutions in the source string. Multiple entries of the same substitution (redundancies) are ignored. Does not include nested macros such as:

```
$ (P=$ (S) ) ${S_}$ (P) }
$ (PJ=$ (P) ) ${S_}$ (P) }
```

For these, only the innermost are returned:

```
[ '$ (S) ', '$ (P) ' ]
[ '$ (P) ' ]
```

Note This routine will also properly identify command shell macro substitutions.

Parameters **source** – string with possible (EPICS) macro substitution expressions

Returns list of macro substitutions found

module: record

EPICS record

class `iocdoc.record.PV` (*record_object*, *ref*, ***env*)

Bases: `object`

single instance of an EPICS record, will expand all macros as provided

class `iocdoc.record.Record` (*parent*, *rtype*, *rname*, *ref*, ***env*)

Bases: `object`

definition of an EPICS record, macros are not expanded

module: reports

module: template

module: text_file

`text_file.py` - describe any text file used by an EPICS IOC

code	description
<code>read()</code>	get a file either from the cache or from storage
<code>items()</code>	get the cache as a set of dictionary items
<code>keys()</code>	get the names of files in the cache
<code>values()</code>	get the Python objects of items in the cache
<code>remove_comments()</code>	strip out a C-style comment
<code>FileNotFound</code>	Exception: raised when <code>filename</code> does not exist
<code>_FileCache</code>	(internal) supports “load each file only once”
<code>_TextFile</code>	(internal) superclass: common handling of text file

Example:

```
# filename must have all macros expanded
file_object = text_file.read(filename)
```

`iocdoc.text_file.items()`
get the cache as a set of dictionary items

`iocdoc.text_file.keys()`
get the names of files in the cache

`iocdoc.text_file.read(filename)`
get a file either from the cache or from storage

Parameters `filename` (*str*) – relative or absolute path to file

Always use filenames with all macros expanded.

Ok?	filename
Ok	<code>st.cmd</code>
Ok	<code>./testfiles/templates/example.template</code>
not Ok	<code>\$(SSCAN) /sscanApp/Db/scanParms.db</code>

`iocdoc.text_file.values()`
get the Python objects of items in the cache

module: token_support

Applies Python `tokenize` analysis to each line of a text file.

class `iocdoc.token_support.TokenLog`

Applies the Python `<code>tokenize</code>` analysis to each line of a file. This allows a lexical analysis of the file, line-by-line. This is powerful and makes some complex analyses more simple but it assumes the file resembles Python source code.

:note The `<code>tokenize</code>` analysis is not robust. Some files will cause exceptions for various reasons.

:see <http://docs.python.org/library/tokenize.html> :see <http://docs.python.org/library/token.html>

get (*index*)
retrieve the indexed token from the list

getCrossReferences ()

Returns dictionary of token cross-references

getFullWord ()
parse the token stream for a contiguous word and return it as str

Some words in template files might not be enclosed in quotes and thus the whole word is broken into several tokens. This command rebuilds the word, without stripping quotes (if provided).

getKeyValueSet ()

parse a token sequence as a list of macro definitions into a dictionary

example:

```
{ P=12ida1:,SCANREC=12ida1:scan1 }
{P=12ida1:,SCANREC=12ida1:scan1,Q=m1,POS="$ (Q) .VAL",RDBK="$ (Q) .RBV" }
```

getTokenList ()

Returns list of token dictionaries

lineAnalysis ()

analyze the tokens by line

:return dictionary with all the lines, including tokenized form

next ()

return the next token or raise a `StopIteration` exception upon reaching the end of the sequence

Returns token object

Raises `StopIteration` – reached the end of the sequence

nextActionable (*skip_list=None*)

walk through the tokens and find the next actionable token

Parameters `skip_list` (*(str)*) – list of tokens to ignore

default list: ('COMMENT', 'NEWLINE', 'ENDMARKER', 'ERRORTOKEN', 'INDENT', 'DEDENT')

Returns token object or None if no more tokens

previous ()

return the previous token

Returns token object

Raises `StopIteration` – reached the beginning of the sequence

processFile (*filename*)

process just one file

report ()

prints (to stdout) results contained in tokenList list and xref dictionary

setTokenPointer (*position=None*)

set the token pointer to the given position

Parameters `position` – index position within list of tokens

Raises `Exception` – token pointer position errors

summary (*alsoPrint=False*)

Summarizes the xref dictionary contents. Reports number of each different token name (type).

Parameters `alsoPrint` – boolean to enable print to stdout

Returns dictionary of token frequencies

tokenName (*tokType*)

convert token number to a useful string

tokenReceiver (*tokType, tokStr, start, end, tokLine*)
called by tokenize package, logs tokens as they are called

tokens_to_list ()
parse an enclosed list of tokens into a list
Assume token_pointer is pointing at start terminator

examples:

```
(DESC, "motor $(P)$ (M)" ) --> ['DESC', 'motor $(P)$ (M) ']  
{P,      S, BL,      T1, T2, A} --> ['P', 'S', 'BL', 'T1', 'T2', 'A']  
{12ida1: A  "##ID" 1   2   1} --> ['12ida1:', 'A', '##ID', '1', '2', '1']  
  
TODO: alias $(IOC):IOC_CPU_LOAD, "$ (IOC):load"
```

iocdoc.token_support.**parse_bracketed_macro_definitions** (*tokenLog*)
walk through a bracketed string, keeping track of delimiters
verify we start on an opening delimiter

iocdoc.token_support.**reconstruct_line** (*tokens=[], firstIndex=1*)
reconstruct the line from the list of tokens presented

Parameters

- **tokens** – as used throughout this module
- **firstIndex** – first index in tokens list to use

Returns reconstructed line

iocdoc.token_support.**token_key** (*tkn*)
developer use, short string identifying the type and text of this token

module: utils

common routines for many modules

support	description
<i>chdir()</i>	change current IOC shell directory
<i>datenow()</i>	get a file either from the cache or from storage
<i>detailedExceptionLog()</i>	log the details of an exception
<i>FileRef</i>	associate filename and line number of an object
<i>logMessage()</i>	log a message
<i>strip_outer_pair()</i>	remove outer symbols from text
<i>strip_outer_quotes()</i>	strip outer quotes (either single or double) from text
<i>remove_comments()</i>	strip out a C-style comment
LOG_FILE	default log file name
<i>strip_parentheses()</i>	remove outer parentheses from text
<i>strip_quotes()</i>	strip outer double quotes from text

class iocdoc.utils.**FileRef** (*filename, linenumber, colnumber, obj*)
Bases: object

associate filename and line number of an object

iocdoc.utils.**chdir** (*newDir, nfsMounts={}*)
change the current working directory for the IOC shell

Parameters `newDir` – name of new directory

Returns success (True) or failure (False)

`iocdoc.utils.datetime()`

return date and time now as a string

`iocdoc.utils.detailedExceptionLog(title='', print_traceback=True)`

enter details of an exception to the log (developer tool)

`iocdoc.utils.logMessage(text)`

log a message

`iocdoc.utils.remove_comments(text)`

strip out a C-style comment

```
/* such as this */
```

See <http://stackoverflow.com/questions/241327/python-snippet-to-remove-c-and-c-comments>

`iocdoc.utils.strip_outer_pair(text, left, right=None)`

remove outer symbols from text

Parameters

- **text** – string
- **left** – symbol on left side
- **right** – symbol on right side (default is left-side symbol)

Returns modified string

Raises **Exception** – left and right must have `len(..) == 1`

`iocdoc.utils.strip_outer_quotes(text)`

strip outer quotes (either single or double) from text

Returns text without comments

`iocdoc.utils.strip_parentheses(text)`

remove outer parentheses from text

Parameters **text** – string

Returns modified string

`iocdoc.utils.strip_quotes(text, quote="")`

strip outer double quotes from text

Returns text without comments

CHANGES

2016-03-24 0.0.2 pre-release

2016-01-29 0.0.1 - project to be refactored from topdoc

2014 TopDoc - <https://subversion.xray.aps.anl.gov/trac/bcdaext/browser/topdoc>

Source Code License

```

1 Copyright (c) 2011-2016, UChicago Argonne, LLC
2
3 All Rights Reserved
4
5 IOCDOC
6
7 Advanced Photon Source, Argonne National Laboratory
8
9
10 OPEN SOURCE LICENSE
11
12 Redistribution and use in source and binary forms, with or without
13 modification, are permitted provided that the following conditions are met:
14
15 1. Redistributions of source code must retain the above copyright notice,
16    this list of conditions and the following disclaimer. Software changes,
17    modifications, or derivative works, should be noted with comments and
18    the author and organization's name.
19
20 2. Redistributions in binary form must reproduce the above copyright notice,
21    this list of conditions and the following disclaimer in the documentation
22    and/or other materials provided with the distribution.
23
24 3. Neither the names of UChicago Argonne, LLC or the Department of Energy
25    nor the names of its contributors may be used to endorse or promote
26    products derived from this software without specific prior written
27    permission.
28
29 4. The software and the end-user documentation included with the
30    redistribution, if any, must include the following acknowledgment:
31
32    "This product includes software produced by UChicago Argonne, LLC
33    under Contract No. DE-AC02-06CH11357 with the Department of Energy."
34
35 *****
36
37 DISCLAIMER
38
39 THE SOFTWARE IS SUPPLIED "AS IS" WITHOUT WARRANTY OF ANY KIND.
40
41 Neither the United States GOVERNMENT, nor the United States Department
42 of Energy, NOR UChicago Argonne, LLC, nor any of their employees, makes
43 any warranty, express or implied, or assumes any legal liability or
44 responsibility for the accuracy, completeness, or usefulness of any
45 information, data, apparatus, product, or process disclosed, or
46 represents that its use would not infringe privately owned rights.
47
48 *****

```


CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`

i

- `iocdoc.database`, 2
- `iocdoc.macros`, 2
- `iocdoc.record`, 4
- `iocdoc.text_file`, 4
- `iocdoc.token_support`, 5
- `iocdoc.utils`, 7

C

chdir() (in module iocdoc.utils), 7

D

Database (class in iocdoc.database), 2

datenow() (in module iocdoc.utils), 8

detailedExceptionLog() (in module iocdoc.utils), 8

E

exists() (iocdoc.macros.Macros method), 3

F

FileRef (class in iocdoc.utils), 7

G

get() (iocdoc.macros.Macros method), 3

get() (iocdoc.token_support.TokenLog method), 5

getCrossReferences() (iocdoc.token_support.TokenLog method), 5

getFullWord() (iocdoc.token_support.TokenLog method), 5

getKeyValueSet() (iocdoc.token_support.TokenLog method), 6

getTokenList() (iocdoc.token_support.TokenLog method), 6

I

identifyEpicsMacros() (in module iocdoc.macros), 4

iocdoc.database (module), 2

iocdoc.macros (module), 2

iocdoc.record (module), 4

iocdoc.text_file (module), 4

iocdoc.token_support (module), 5

iocdoc.utils (module), 7

items() (in module iocdoc.text_file), 5

items() (iocdoc.macros.Macros method), 3

K

keys() (in module iocdoc.text_file), 5

keys() (iocdoc.macros.Macros method), 3

KVpair (class in iocdoc.macros), 3

L

lineAnalysis() (iocdoc.token_support.TokenLog method), 6

logMessage() (in module iocdoc.utils), 8

M

Macros (class in iocdoc.macros), 3

makeProcessVariables() (iocdoc.database.Database method), 2

N

next() (iocdoc.token_support.TokenLog method), 6

nextActionable() (iocdoc.token_support.TokenLog method), 6

P

parse() (iocdoc.database.Database method), 2

parse_bracketed_macro_definitions() (in module iocdoc.token_support), 7

previous() (iocdoc.token_support.TokenLog method), 6

processFile() (iocdoc.token_support.TokenLog method), 6

PV (class in iocdoc.record), 4

R

read() (in module iocdoc.text_file), 5

reconstruct_line() (in module iocdoc.token_support), 7

Record (class in iocdoc.record), 4

remove_comments() (in module iocdoc.utils), 8

replace() (iocdoc.macros.Macros method), 3

report() (iocdoc.token_support.TokenLog method), 6

S

set() (iocdoc.macros.Macros method), 4

setMany() (iocdoc.macros.Macros method), 4

setTokenPointer() (iocdoc.token_support.TokenLog
method), 6
strip_outer_pair() (in module iocdoc.utils), 8
strip_outer_quotes() (in module iocdoc.utils), 8
strip_parentheses() (in module iocdoc.utils), 8
strip_quotes() (in module iocdoc.utils), 8
summary() (iocdoc.token_support.TokenLog method), 6

T

techniques, 2
token_key() (in module iocdoc.token_support), 7
TokenLog (class in iocdoc.token_support), 5
tokenName() (iocdoc.token_support.TokenLog method),
6
tokenReceiver() (iocdoc.token_support.TokenLog
method), 6
tokens_to_list() (iocdoc.token_support.TokenLog
method), 7

V

values() (in module iocdoc.text_file), 5