Configuration File Syntax

6 April 2022
Release 5.5+
Andrew Hanushevsky

1 Introduction 5
2 Specifying Conditional Directives 7
3 Configuration File Continuation 11
4 Using set Variables 13
4.1 Assigning Variable Values 13
4.2 Substituting Variables 15
4.3 Specifying set Options 17
4.4 Assigning Environmental Variable Values 18
5 Document Change History 21

1 Introduction

This document describes the syntax used in the configuration file for **xrootd**, **cmsd,** and all other related components**.** Refer to the respective configuration documents on directive details.

All configuration directives start with a prefix identifying the system component to which the directive applies. The prefix is separated by the actual directive keyword by a single period. This allows configuration of all aspects of a system using a single configuration file. The following table lists valid prefixes

Prefix	System Component
acc	**Acc**ess control (i.e., authorization)
cms	**C**luster **M**anagement **S**ervices
dig	The **digFS** built-in file system
frm	**F**ile **R**esidency **M**anager
http	HTTP protocol plug-in.
ofs	**O**pen **f**ile **s**ystem coordinating **acc**, **cms** & **oss** components
oss	**O**pen **s**torage **s**ystem (i.e., file system implementation)
pfc	**P**roxy **F**ile **C**ache plug-in
pss	**P**roxy **S**torage **S**ervice plug-in
sec	**Sec**urity authentication
xrd	E**x**tended **R**equest **D**aemon
xrootd	The **xrootd** protocol implementation.
all	Applies the directive to all of the above components.

Records that do not start with a recognized identifier are ignored. This includes blank record, comment lines (i.e., lines starting with a pound sign, #), and prefixes not immediately followed by a single period. Because each component has a unique prefix, a common configuration file can be used for the whole system. The location of the configuration file is specified on the command line. Refer to the reference manuals for each component on how it locates the configuration files.

This guide documents the basic directive syntax and describes the use of conditional statements and set variables within the configuration file.

2 Specifying Conditional Directives

The **if-fi** directives are used to allow you to optionally include directives based on host and instance name. The syntax for this directive pair is:

```
  if [ hostpat [. . .] ] [ conds ]

     [ directives when if is true ]

[ else if [ hostpat [. . .] ] [ conds ]

     [ directives when all previous if's are false
                          and this if   is  true ]
]
●
●    [ additional "else if" clauses, as desired ]
●
[ else

     [directives when all previous if's are false ]
]

  fi

hostpat: host | host+ | pfx* | *sfx | pfx*sfx]

conds:cond1 | cond2 | cond3

cond1:defined var [. . .] [&& {cond1 | cond2 | cond3}]

cond2:exec pgm [. . .] [&& cond3]

cond3:named name [. . .]

var:  ?varname | ?~varname
```

Function

Specify the conditions under which subsequent directives are to be used.

Parameters

hostpat

The pattern of the host to which subsequent directive applies. All non-applicable hosts ignore all directives until the next **else** or **fi**. Host patterns are:

host	Any host that matches the specified **DNS** name.
*host***+**	Any host that has an address that matches any of the addresses assigned to host.
*pfx******	Any host starting with *pfx*.
*****sfx	Any host ending with *sfx*.
*pfx******sfx	Any host beginning with *pfx* and ending with *sfx*.

name An instance name (i.e., a name that can be specified using the **–n** command line option). All directives until the next **else** or **fi** are ignored unless the executable has been given one of the instance names in the list of names.

pgm The prefix-name of the executable. The prefix-name is defined to be all of the characters in the base filename (i.e., the directory path removed) up to but not including the first dot in the name, if any. If the name starts with a dot, the prefix-name is the complete base filename. All directives until the next **else** or **fi** are ignored unless the executable has the given name.

var A set variable name or an environmental variable name, *varname*. "**?***varname*" refers to set variable; while "**?~***varname*" refers to an environmental variable. All directives until the next **else** or **fi** are ignored unless one of the specified variables in the list of variables is defined.

Defaults

None. At least one *hostpat*, **defined**, **exec** or the **named** keyword must be specified.

Notes

1) All specified conditions must be true (i.e., *hostpat*, **defined**, **exec**, and **named**) for the subsequent directives to be used.
2) A double ampersand (**&&**) is used to "and" two or more *named* tests. Be aware that the specified *named* tests must appear in the specific order (i.e. **defined** before **exec** and **exec** before **named**).
3) A qualified if is an **if** that is preceded by an else on the same line. An unqualified if is an **if** that appears first on a line.

4) Every unqualified **if** must be followed by a **fi**. Every **fi** must be preceded by a qualified or unqualified **if**.
5) Every **else** must be preceded by a qualified or unqualified **if**.
6) Nested unqualified **if** directives are not allowed.
7) The name **anon** refers to servers that were not given a name via **–n**.
8) Some directives allow the "if" to be placed as the rightmost tokens on the associated directive line. For these directives, no "fi" is required as the end of the line determines the **if**'s scope.

Examples

```
if *slac.stanford.edu named anon
xrd.port 9999
fi

if named public
xrd.port 8888
else
xrd.port 9999
fi

if exec cmsd && named public
xrd.port 2131
else
xrd.port 1094
fi

if exec cmsd && named public
xrd.port 2131
else if exec cmsd && named private
xrd.port 3121
else
xrd.port 1094
fi

if defined ?~EXPORTPATH
set exportpath = $EXPORTPATH
else
set exportpath = /tmp
fi
all.export $exportpath
```

3 Configuration File Continuation

```
continue [[?]{dirpath | filepath]] [sfx] [if spec]

sfx: *txt [sfx]
```

Function

Specify the file(s) to continue the current configuration file.

Parameters

dirpath

the path to a directory holding additional configuration files. The directory is scanned for applicable files and each such file, in lexical order, is used to extend the current configuration file. Applicable files are those that

a) do not start with a dot (.),
b) are not marked as executable files, and
c) do not end with the historical suffixes of ".cfsaved", ".rpmsave", ".rpmnew", ".dpkg-old", ".dpkg-dist", or "~" (i.e. tilde).

filepath

the path to a file holding additional configuration directives. The file is used as an immediate continuation to the current configuration file. The file is acceptable as long as it does not have the execute bit set.

sfx one or more file suffixes that are allowed to be continuations (files marked as executable are still ignored). This option is only meaningful when *dirpath* is specified. When *filepath* is specified, any specified valid suffix specifications are ignored.

txt the characters that must appear at the end of the filename for the file to be considered applicable.

spec a valid **if** directive clause. The **continue** statement only applies if the clause evaluates to true.

Notes

1) A **continue** directive with no arguments simply continues the processing of the current configuration file. An empty argument list may occur due to variable substitution.
2) The **continue** directive is not allowed in a continuation (i.e., a continuation may not continue to another file).
3) The **continue** directive may not appear in an **if-else-fi** clause. Use an inline **if** to control its applicability.
4) Continuations are particularly useful for defining a base configuration that allows site, VM or container specific augmentation.
5) Be very aware that component directives may be cumulative or replaceable. Refer to the specific directive that you wish to alter should it appear in an antecedent configuration file.

Example

```
continue /etc/morecfg .cf .cfg .conf if named foobar
```

4 Using set Variables

4.1 Assigning Variable Values

```
set var { = { value | varname } | < path }

varname: $envvar | $(envvar) | ${envvar} | $[envvar]
```

Function

Specify the value a set variable must have.

Parameters

var The name of a variable. Variable names may only contain letters and digits and should start with a letter. Case is significant. Variable names may not be longer than 63 characters.

value The value to be assigned to the variable. It must consist of a single non-blank text token no longer than 511 characters.

path The path to the file that contains the value to be assigned to the variable. The file must not be longer than 1023 characters.

varname

The value comes from an environmental variable named *envvar*. The environmental variable must not be longer than 511 characters. In most cases the environmental variable must be defined, as explained below.

Specification	**Defined** *envvar*	**Undefined** *envvar*
$*envvar*	Definition substituted	Fatal error
$(*envvar***)**	Definition substituted	Fatal error
${*envvar***}**	Definition substituted	Fatal error
$[*envvar***]**	Definition substituted	Null string substituted

Notes

1) Unless **$[***envvar***]** notation is used; use of an environmental variable that has not been set is considered to be a fatal error.

Example

```
set myVar = myToken
set yourVar=$EnvVar
```

4.2 Substituting Variables

```
1st_token subs

subs:    [vname][text][subs]

vname:   $var | ${var} | $(var) | $[var]
```

Function

Specify a variable to be substituted by its set value.

Parameters

1st_token

The first token in any line of a configuration file. The first token may never specify a variable and is one of the following:

- A prefixed directive
- **if**, **else**, or the token **fi**
- **set**
- # (indicating a comment)

text Any text.

vname The name of a set variable. The variable name ends when an non-alphanumeric character is encountered; including the of the line. The variable's value replaces name, as follows:

Specification	**Defined** *var*	**Undefined** *var*
$*var*	Definition substituted	Fatal error
$(*var***)**	Definition substituted	Fatal error
${*var***}**	Definition substituted	Fatal error
$[*var***]**	Definition substituted	Null string substituted

Notes

1) Except for **$[***var***]**; use of a variable that has not been set is considered to be a fatal error.
2) Variables may be used in any text line other than a **set** statement.
3) Substitution occurs only once. Substituted lines are never rescanned.

Example

```
set myHost = io.slac.stanford.edu
set myPath = /foo/fum/fi/

all.role manager if $myHost
ofs.fslib $(myPath)libXrdOfs.so
```

4.3 Specifying set Options

```
set { -q | -v | -V }
```

Function

Specify the level of substitution detail.

Parameters

-q Enables quiet mode. Neither substitutions nor substituted lines are displayed.

-v Enables verbose mode. While substitutions are not displayed; substituted lines are displayed. This is the default.

-V Enables very verbose mode. Both substitutions and substituted lines are displayed.

Defaults

set -v

Notes

1) Configuration files are processed by multiple components. Every time a component scans through a configuration file and "**-v**" is in effect, substituted lines used by that component are displayed.
2) When "**-V**" is in effect, every time a variable is given a value the assignment is displayed. This means that each component scanning through the configuration file will generate a display of all **set** statements.

Example

```
set -V
```

4.4 Assigning Environmental Variable Values

```
setenv envvar { = { value | varname } | < path }

varname: $var | $(var) | ${var} | $[var]
```

Function

Specify the value a environmental variable must have.

Parameters

envvar The name of an environmental variable. Variable names may only contain letters and digits and should start with a letter. Case is significant. Environmental variable names may not be longer than 63 characters and may not start with XRD.

value The value to be assigned to the environmental variable. It must consist of a single non-blank text token no longer than 511 characters.

varname

The value comes from a set variable named *var*. In most cases the set variable must be defined, as explained below.

path The path to the file that contains the value to be assigned to the variable. The file must not be longer than 1023 characters.

Specification	**Defined** *envvar*	**Undefined** *envvar*
$*var*	Definition substituted	Fatal error
$(*var***)**	Definition substituted	Fatal error
${*var***}**	Definition substituted	Fatal error
$[*var***]**	Definition substituted	Null string substituted

Notes

1) Unless **$[***var***]** notation is used; use of an environmental variable that has not been set is considered to be a fatal error.

Example

```
setenv EnvVar = myToken
set myPath = /foo/fum/fi/
```

```
setenv EnvPath = $[myPath]
```

5 Document Change History

29 March 2007

- Manual introduced.

8 January 2008

- Deprecate the **odc** and **olb** components.

23 June 2009

- Document the new "if/else if/else/fi" construct.

13 November 2010

- Document the new "**setenv**" construct.
- Allow undefined variables to be used via the **$[]** construct.
- Remove references to the **odc** and **olb** components.

4 May 2014

- Document the new **defined** if-test construct.
- Add **http** as a component name.

14 August 2015

- Add missing components to the component table (i.e. **dig**, **frm**, **pfc**, and **pss**).
- Explain the use of the double ampersand.

27 July 2018

- Document the **continue** directive.

6 April 2022

- Document that the **set** and **setenv** constructs can obtain the value from a file.