Kikupd

Version : 1.0
Date : 19/08/2006
Author : ¨Patrick Germain Placidoux

SUMMARY

1 INTRODUCTION

The kikupd command provides an administrative interface on the kikonf Action files.

2 OVERVIEW

2.1 INFORMATION ABOUT PLUGINS

The kikupd command provides information about intalled Actions using the -p (--plugins), -P (aplugin) or --info options.

2.1.1 The -p (--plugins) option

The -p (--plugins) option provides informations about all installed Actions for this kikonf installation.

>kikupd -p

```
-----------------
| Category: was |
-----------------

Plugin> was.cache
Category: was
Name: cache
License: Modified BSD License
Multiple: false
By: kikonf
Version: 5.0

Plugin> was.chgclonid
Category: was
Name: chgclonid
License: Modified BSD License
Multiple: false
By: kikonf
Version: 5.0
...
```

2.1.2 The -P (--aplugin) option

The -P (--aplugin) option provides informations about a specific Action.

>kikupd -P was.cache

Plugin> was.cache

Category: was
Name: cache
License: Modified BSD License
License file:
E:\Projets\DEPLOYMENT\kikonf\plugins\actions\was\cache\by\kikonf\ACT_INF\action.attrs\COPYING
Multiple: false
By: kikonf
Version: 5.0

Note:
The notation used to design a specific Action is the same as the one in use within the kikact command which syntax is: <BAL>
For more information about the BAL (Basic Action Locator) syntax see Annex 1.

2.1.3 The –info option

The –info option provides explicits informations about an Action:

>kikupd --info was.cache

```
-------------------------------------------------------------------------------------------------
|Descriptor: E:\Projets\DEPLOYMENT\kikonf\plugins\actions\was\cache\by\kikonf\ACT_INF\action.xml|
-------------------------------------------------------------------------------------------------

/cache>                                  A Cache is a memory space where to store objects
_____________________________________________________________________________________
Name            Default   Type          Required   Denied  Help
_____________________________________________________________________________________
name                                    True
jndi_name                               True
size            50        int           True
prefix                                  Using prefix for configuration resource names is a generic way to define them
desc                      str

/cache/scope>             True          A scope refers to a Name Space where to create the resource
_____________________________________________________________________________________
Name            Default   Type          Required   Denied  Help
_____________________________________________________________________________________
cell            false     (false, true) True
node                      str
server                    str
cluster                   str

--------
|Sample|
--------

<cache type='action' bal='None' sub_type='configuration' softwares='None' name='mycache' jndi_name='cache/mycache2'
size='100' prefix='None' desc='None'>
   <scope cell='false' node='localhostNode01' server='server1' cluster='None'/>
</cache>
```

2.2 THE KIKUPD MODES

The kikupd command works in two modes:
the **console mode** and the **request mode**.

Running kikupd with the -o (--console) option, turns it into the console mode

e.g.:
>kikupd --kikact was.cache -o

```
[serenity]     CACHE

 ____________________
|                    |
| ACTION: CACHE      |
|____________________|


 1/ cache>             [A Cache is a memory space where to store objects]
   name: mycache
   jndi_name: cache/mycache2
   size: 100
   prefix: None
   desc: None

 2/ cache/scope>       [A scope refers to a Name Space where to create the resource]
   cell: false
   node: localhostNode01
   server: server1
   cluster: None

 3) Show Action file

 4) Show Descriptor file

 5) Show Help

 6) Check All

 7) Print

 8) Save


   ( Exit:0)

         Choice ?
```

Running kikupd **without** the -o (--console) option, turns it into the request mode

e.g.:
>kikupd --kikact was.cache aa
EEpicxml: td:Request Error: First Tag is not:aa, but:cache.
Your Top Down PxQuery request:aa.

>kikupd --kikact was.cache cache
/cache> type:action bal:None sub_type:configuration softwares:None name:mycache jndi_name:cache/mycache2 size:100 prefix:None desc:None
scope

2.3 KIKUPD INPUTS

kikupd inputs can be one of the three followings.

2.3.1 Using –kikact <BAL>, a specific Action

e.g. in console mode:

>kikupd --kikact was.jdbc –console

```
[serenity]     JDBC

 ___________________
|                   |
| ACTION: JDBC      |
|___________________|


 1/ jdbc>              [Create one JDBC provider]
    name: myprovider
    path: /my/database/jdbc/path
    prefix: None
    desc: None

 2/ jdbc/scope>        [A scope refers to a Name Space where to create the resource]
    cell: false
    node: localhostNode01
    server: server1
    cluster: None

 3/ New jdbc/oracle>

 4/ jdbc/db2>
    xa: true
    jars: db2jcc.jar;db2jcc_license_cu.jar;Another_path

 5/ New jdbc/msql>
```

6) Show Action file

7) Show Descriptor file

8) Show Help

9) Check All

10) Print

(Exit:0 Up:+)

Choice ?

e.g. in request mode:

>kikupd --kikact was.jdbc jdbc@path
/my/database/jdbc/path

or

>kikupd --kikact was.jdbc -u jdbc[@name=newPovider,@path=/new/path]
updated node:/jdbc>
From path:/my/database/jdbc/path To path:/new/path
From name:myprovider To name:newPovider

2.3.2 Using –kikarc <ACTION_FILE>, a custom xml file

The following c.xml file sample comes from the **<kikonf_install_root>/tests** directory.
You can execute: cd **<kikonf_install_root>/tests** then run the following commands.

e.g. in console mode:

>kikupd --kikarc c.xml --console

```
[serenity]     C.XML

 _____________________
|                     |
| CUSTOM FILE:C.XML |
|_____________________|

 1/ Action: clsloader    [Change a given Classloader Mode and Policy]
   type: action
   bal: was.clsloader
   sub_type: configuration
   softwares: None
   policy: MULTIPLE
   mode: PARENT_FIRST

 2/ Action: clsloader    [Change a given Classloader Mode and Policy]
   type: action
   bal: was.clsloader
   sub_type: configuration
   softwares: None
   policy: MULTIPLE
   mode: PARENT_FIRST

 3/ Action: crtcluster   [Creates a new Cluster]
   type: action
   bal: was.crtcluster
   sub_type: configuration
   softwares: None
   name: cluster1
   enable_ha: false
   node_group: DefaultNodeGroup
   core_group: DefaultCoreGroup
   desc: None

   ( Exit:0  Up:+)

         Choice ?
```

e.g. in request mode:

>kikupd --kikarc c.xml application
/application> name:myapp
clsloader
modules
clusters
servers

>kikupd --kikarc c.xml application/servers/server@name=server2/crtserver/chgports
/application/servers/server/crtserver/chgports> type:action bal:was.chgports sub_type:configuration softwares:None starting_port:None
scope
was6Ports

or

>kikupd --kikarc c.xml application/servers/server@name=server2/crtserver/chgports[@starting_port=10000] -u
updated node:/application/servers/server/crtserver/chgports>
From starting_port:None To starting_port:10000

2.3.3 Using -F -D and/or -R, any described xml file

The following samples comes from the **<kikonf_install_root>/tests** directory.
You can execute: cd **<kikonf_install_root>/tests** then run the following commands.

e.g. in console mode:
(todo)

e.g. in request mode:
>kikupd -F test.xml -D test.desc.xml tag1/tag3
/tag1/tag3> attr1:f attr2:g attr3:A value with spaces !

or

>kikupd -F test.xml -D test.desc.xml -n "tag1/tag3[@attr3=A new spaced value]"
Created node: /tag1/tag3>
updated node:/tag1/tag3>
From attr3:None To attr3:A new spaced value

Note:
Check out the **<kikonf_install_root>/tests** directory for samples using the kikupd (and kikact or kikarc) commands.

3 THE REQUEST MODE

The request mode allows to run epicxml request to a guiven kikupd input.
The resulting in memory xml file can then be saved using the -f (to_file) option or –overwrite option.

Note:
Be aware not to overwrite the default provided Action file samples under
<kikonf_install_root>/actions using the –overwrite option.

The kikupd command is based upon the epicxml parser which itself implements the picpath syntax from the picxml parser.

To understand the picpath syntax please consult the picxml documentation under the <kikonf_install_root>/doc directory or the picxml project at sourceforge.net.

The epicxml parser extends the picxml parser in the way it implements update and remove operations.

Here are provided samples using the kikupd command in request mode.
Those samples come from the **<kikonf_instalml_root>/tests** directory.
You can execute: cd **<kikonf_instalml_root>/tests** and run each of the following commands.

3.1 USING (--KIKACT) AN ACTION AS INPUT

3.1.1 With the cache Action

>kikupd --kikact was.cache cache
/cache> type:action bal:None sub_type:configuration softwares:None name:mycache jndi_name:cache/mycache2 size:100 prefix:None desc:None
scope

>kikupd --kikact was.cache cache/scope
/cache/scope> cell:false node:localhostNode01 server:server1 cluster:None

>kikupd --kikact was.cache --update cache/scope[@node=node2,@server=server2]
updated node:/cache/scope>
From node:localhostNode01 To node:node2
From server:server1 To server:server2

>kikupd --kikact was.cache --update cache/scope[@node=node2,@server=server2] --print
updated node:/cache/scope>
From node:localhostNode01 To node:node2
From server:server1 To server:server2

<cache type='action' bal='None' sub_type='configuration' softwares='None' name='mycache' jndi_name='cache/mycache2' size='100' prefix='None' desc='None'>
<scope cell='false' node='node2' server='server2' cluster='None'/>
</cache>

3.1.2 With the Jvm Action

>kikupd --kikact was.jvm jvm
/jvm> type:action bal:None sub_type:configuration softwares:None xms:512 xmx:1024 run_hprof:true disable_jit:true temp_dir:/dir/to/my/temp_dir ha:true plugin_ma
x_con:52 user:myuser group:mygroup
scope
verbose
classpath
boot_classpath
debug_args
generic_jvm_args
hprof_args
system_props
timouts
extlibs
gc
shared_classes

>kikupd --kikact was.jvm jvm/verbose
/jvm/verbose> cl:true gc:true jni:true

>kikupd --kikact was.jvm --update jvm[@user=user1,@group=group2] jvm/verbose[@gc=true]
updated node:/jvm>
From group:mygroup To group:group2
From user:myuser To user:user1

updated node:/jvm/verbose>
From gc:true To gc:true

>kikupd --kikact was.jvm jvm/gc/policy/gencon
/jvm/gc/policy/gencon> xmns:128 xmnx:256 xmos:128 xmox:768

>kikupd --kikact was.jvm --update jvm[@user=user1,@group=group2] jvm/verbose[@gc=true] jvm/gc/policy/gencon[@xmns=64,@xmnx=128]

updated node:/jvm>
From group:mygroup To group:group2
From user:myuser To user:user1

updated node:/jvm/verbose>
From gc:true To gc:true

updated node:/jvm/gc/policy/gencon>
From xmns:128 To xmns:64
From xmnx:256 To xmnx:128

3.2 USING (--KIKARC) A CUSTOM FILE AS INPUT

3.2.1 With the Clsloader Action

>kikupd --kikarc c.xml application
/application> name:myapp
clsloader
modules
clusters
servers

>kikupd --kikarc c.xml application/modules
/application/modules>
module
module

>kikupd --kikarc c.xml application/modules/module
/application/modules/module> name:Increment Enterprise Java Bean

/application/modules/module> name:Default Web Application
clsloader

>kikupd --kikarc c.xml "application/modules/module@name=Default Web Application"
/application/modules/module> name:Default Web Application
clsloader

>kikupd --kikarc c.xml --update "application/modules/module@name=Default Web
Application/clsloader[@policy=SINGLE]"

3.2.2 With the Crtserver Action

>kikupd --kikarc c.xml aaa
EEpicxml: td:Request Error: First Tag is not:aaa, but:application.
Your Top Down PxQuery request:aaa.

>kikupd --kikarc c.xml application
/application> name:myapp
clsloader
modules

clusters
servers

>kikupd --kikarc c.xml application/servers
/application/servers>
server
server

>kikupd --kikarc c.xml application/servers/server
/application/servers/server> name:server1
crtserver

/application/servers/server> name:server2
crtserver

>kikupd --kikarc c.xml application/servers/server@name=server2
/application/servers/server> name:server2
crtserver

>kikupd --kikarc c.xml application/servers/server@name=server2/crtserver
/application/servers/server/crtserver> type:action bal:was.crtserver sub_type:configuration softwares:None template:default weight:2 desc:None
scope
chgports
logs
tp
tp
tp
tp
tp
tp
tp
clsloader
session
resources

>kikupd --kikarc c.xml application/servers/server@name=server2/crtserver/chgports
/application/servers/server/crtserver/chgports> type:action bal:was.chgports sub_type:configuration softwares:None starting_port:None
scope
was6Ports

>kikupd --kikarc c.xml
application/servers/server@name=server2/crtserver/chgports[@starting_port=10000] -u
updated node:/application/servers/server/crtserver/chgports>
From starting_port:None To starting_port:10000

>kikupd --kikarc c.xml application/servers/server@name=server2/crtserver/chgports
/application/servers/server/crtserver/chgports> type:action bal:was.chgports sub_type:configuration softwares:None starting_port:None
scope
was6Ports

>kikupd --kikarc c.xml application/servers/server@name=server2/crtserver/chgports/was6Ports -e

Removed: /application/servers/server/crtserver/chgports/was6Ports<

3.3 USING (-F, -D) ANY DESCRIBED XML FILE AS INPUT

3.3.1 List operations

>kikupd -F test.xml -D test.desc.xml tag1
/tag1> attr1:a attr2:b
tag2
tag3
tag2
tag2

>kikupd -F test.xml -D test.desc.xml tag1@attr1
a

>kikupd -F test.xml -D test.desc.xml tag1@attr1,@attr2
attr1:a attr2:b

>kikupd -F test.xml -D test.desc.xml tag1/tag2
/tag1/tag2> attr1:c attr2:d attr3:e attr4:None

/tag1/tag2> attr1:h attr2:i attr3:j attr4:k
tag4

/tag1/tag2> attr1:o attr2:p attr3:None attr4:None

>kikupd -F test.xml -D test.desc.xml tag1/tag2@attr1,@attr3
attr1:c attr3:e,attr1:h attr3:j,attr1:o attr3:None

>kikupd -F test.xml -D test.desc.xml tag1/tag2@attr3
e,j,None

>kikupd -F test.xml -D test.desc.xml tag1/tag2@attr2=d,@attr3
e

>kikupd -F test.xml -D test.desc.xml tag1/tag2@attr2=i,@attr3
j

>kikupd -F test.xml -D test.desc.xml tag1/tag2@attr2=p,@attr3
None

>kikupd -F test.xml -D test.desc.xml tag1/tag2@attr2=d,@attr1,@attr3
attr1:c attr3:e

>kikupd -F test.xml -D test.desc.xml tag1 tag1/tag2
/tag1> attr1:a attr2:b
tag2
tag3
tag2
tag2

/tag1/tag2> attr1:c attr2:d attr3:e attr4:None

/tag1/tag2> attr1:h attr2:i attr3:j attr4:k
tag4

/tag1/tag2> attr1:o attr2:p attr3:None attr4:None

>kikupd -F test.xml -D test.desc.xml tag1/tag2/tag4/tag5
EEpicxml: tdc:Request Error: More than one node found for Sub Level:tag2.
Your Top Down Complete PxQuery request:tag1/tag2/tag4/tag5.

>kikupd -F test.xml -D test.desc.xml tag1/tag2@attr2=i/tag4/tag5
/tag1/tag2/tag4/tag5> attr1:0 attr2:m attr3:n

>kikupd -F test.xml -D test.desc.xml tag1/tag2@attr2=i/tag4/tag5@attr1
0

>kikupd -F test.xml -D test.desc.xml tag1/tag2@attr2=i/tag4/tag5@attr1,@attr3
attr1:0 attr3:n

3.3.2 Update operations

>kikupd -F test.xml -D test.desc.xml tag1/tag3
/tag1/tag3> attr1:f attr2:g attr3:A value with spaces !

>kikupd -F test.xml -D test.desc.xml "tag1/tag3@attr3=A value with spaces !"
/tag1/tag3> attr1:f attr2:g attr3:A value with spaces !

>kikupd -F test.xml -D test.desc.xml -n "tag1/tag3[@attr3=A new spaced value]"
Created node: /tag1/tag3>
updated node:/tag1/tag3>
From attr3:None To attr3:A new spaced value

>kikupd -F test.xml -D test.desc.xml tag1/tag2
/tag1/tag2> attr1:c attr2:d attr3:e attr4:None

/tag1/tag2> attr1:h attr2:i attr3:j attr4:k
tag4

/tag1/tag2> attr1:o attr2:p attr3:None attr4:None

>kikupd -F test.xml -D test.desc.xml -u tag1/tag2[@attr2=a]
updated node:/tag1/tag2>
From attr2:d To attr2:a

updated node:/tag1/tag2>
From attr2:i To attr2:a

updated node:/tag1/tag2>
From attr2:p To attr2:a

>kikupd -F test.xml -D test.desc.xml -u tag1/tag2[@attr1=2,@attr2=2,@attr3=3]
updated node:/tag1/tag2>
From attr2:d To attr2:2
From attr3:e To attr3:3
From attr1:c To attr1:2

updated node:/tag1/tag2>
From attr2:i To attr2:2
From attr3:j To attr3:3
From attr1:h To attr1:2

updated node:/tag1/tag2>
From attr2:p To attr2:2
From attr3:None To attr3:3
From attr1:o To attr1:2

>kikupd -F test.xml -D test.desc.xml -n tag1/tag2[@attr1=2,@attr2=2,@attr3=3]

```
Created node: /tag1/tag2>
updated node:/tag1/tag2>
  From attr2:None To attr2:2
  From attr3:None To attr3:3
  From attr1:None To attr1:2
```

>kikupd -F test.xml -D test.desc.xml -n tag1/tag2[@attr1=2,@attr2=2,@attr3=3] --print

```
Created node: /tag1/tag2>
updated node:/tag1/tag2>
  From attr2:None To attr2:2
  From attr3:None To attr3:3
  From attr1:None To attr1:2

<tag1 attr1='a' attr2='b'>
   <tag2 attr1='c' attr2='d' attr3='e' attr4='None'/>
   <tag3 attr1='f' attr2='g' attr3='A value with spaces !'/>
   <tag2 attr1='h' attr2='i' attr3='j' attr4='k'>
      <tag4>
         <tag5 attr1='0' attr2='m' attr3='n'/>
      </tag4>
   </tag2>
   <tag2 attr1='o' attr2='p' attr3='None' attr4='None'/>
   <tag2 attr1='2' attr2='2' attr3='3' attr4='None'/>
</tag1>
```

>kikupd -F test.xml -D test.desc.xml -e tag1/tag2

```
Removed: /tag1/tag2<
Removed: /tag1/tag2<
Removed: /tag1/tag2<
```

>kikupd -F test.xml -D test.desc.xml -e tag1/tag2 --print

```
Removed: /tag1/tag2<
Removed: /tag1/tag2<
Removed: /tag1/tag2<
<tag1 attr1='a' attr2='b'>
   <tag3 attr1='f' attr2='g' attr3='A value with spaces !'/>
</tag1>
```

>kikupd -F test.xml -D test.desc.xml -e tag1/tag2@attr2=d

```
Removed: /tag1/tag2<
```

>kikupd -F test.xml -D test.desc.xml -e tag1/tag2@attr2=d --print

```
Removed: /tag1/tag2<
<tag1 attr1='a' attr2='b'>
   <tag3 attr1='f' attr2='g' attr3='A value with spaces !'/>
   <tag2 attr1='h' attr2='i' attr3='j' attr4='k'>
      <tag4>
         <tag5 attr1='0' attr2='m' attr3='n'/>
      </tag4>
   </tag2>
   <tag2 attr1='o' attr2='p' attr3='None' attr4='None'/>
</tag1>
```

4 KIKUPD OPTIONS

4.1 USAGE

Usage show a short view of kikact.

>kikupd

Usage:

1) EpicQuery operations

1.1) List operation

Syntax:
<picpath> [<picpath>]

Notes:
The only allowed arguments are picpaths, the other type of parameters are options.
For more informations about picpaths or epicpaths see the picxml or the epicxml documentation.

> kikupd -F /where/is/my.xml -D /where/is/my.desc.xml tag1/tag2=attr1=val1@attr2
val2
or
> kikupd --kikarc c.xml appli/servers/server
This list the node server its attributes and its child names.

1.2) Update operations

Syntax:
[--update|--create|--remove] <picpath> [<picpath>] [--force]

1.3) Update:

> kikupd --kikarc c.xml --update appli/servers/server@[name=server1]
"appli@name=incomes[@comment=my new comment !]"
Udpate: Attributes are updated but all nodes must pre-exist.

1.4) Create:

> kikupd --kikarc c.xml --create appli/servers/server[@name=newServer2,@host=myhost2]
appli/servers/server@name=newServer2/jvm[@xms=256,@xmx=1024]
This creates a new node: server with attributes: "name=newServer2" and "host=myhost2", under appli/servers.
And creates another node: jvm with attributes "xms=256" and "xmx=1024", under the newly created

node "newServer2".
Note: in this request an unique instance of the node servers must pre-exist.

1.5) Remove:

> kikupd --kikarc c.xml --remove appli/servers/server@[name=server1]
This removes the node: "server" and with attribute: "name=server1" under appli/servers.

1.6) EpicQuery on action files (using –kikact)

The same operations than previously except that we use --kikact instead of --kikarc.

e.g. was.cache:
> kikupd --kikact was.cache --update cache@name=newName
cache/scope@node=newNode,@server=newServer

1.7) EpicQuery On trivial xml files (using --file_user (-F))

The same operations than previously except that we use -F and -D options instead of --kikact or --kikarc.

> kikupd --update -F /where/is/my.xml -D /where/is/my.desc.xml tag1/tag2=attr1=val1@attr2=newVal

2) console

> kikupd --kikact /where/is/c.xml --console
or
> kikupd --kikarc was.jvm --console

This generates a console pimenu to allow Actions management.

3) Massive pipe operations

3.1) With custom action files

text.txt:
/where/is/c1.xml
/where/is/c2.xml
/where/is/c3.xml

On unixes:
cat text.txt | kikupd --kikarc --update appli/servers/server@[name=server1]
"appli@name=incomes[@comment=my new comment !]"
On windows:
type text.txt | kikupd --kikarc --update appli/servers/server@[name=server1]
"appli@name=incomes[@comment=my new comment !]"

3.1) With trivial xml files

The text.txt content:
/where/is/my1.xml
/where/is/my2.xml
/where/is/my3.xml

Or with trivial xml files with descriptors:
The text.txt content:
/where/is/my1.xml /where/is/my1.desc.xml
/where/is/my2.xml /where/is/my1.desc.xml
/where/is/my3.xml /where/is/my1.desc.xml

On unixes:
cat text.txt | kikupd --update tag1/tag2=attr1=val1@[attr2=newValue]
"tag1/tag2=attr1=val1@[comment=This is my new comment !]"
On windows:
type text.txt | kikupd --update tag1/tag2=attr1=val1@[attr2=newValue]
"tag1/tag2=attr1=val1@[comment=This is my new comment !]"

3) Kikarc/kikact common operation

The following options are the ones supported when using one of the
--kikact or --kikarc options.

-C KIKONF_ATTRS, --cattrs=KIKONF_ATTRS
(optional) The path to a custom kikonf.attrs file.
When you don't want to use the default one into the
<KIKONF_INSTALL_DIR>/conf directory. Note: If this
option is not set, kikact tries to retreive its value
from an environment variable named KIKONF_CATTRS. If
neither the option or the environment variable are set
kikact use the default kikonf.attrs file into the
<KIKONF_INSTALL_DIR>/conf directory.
-c ACTION_DIR, --cxml=ACTION_DIR
(optional) The path to the directory of the action
files. If not given, sample action files are
retreived from the <KIKONF_INSTALL_DIR>/actions
directory. Note: If this option is not set, kikact
tries to retreive its value from an environment
variable named KIKONF_CXML. If neither the option or
the environment variable are set kikact use the
default <KIKONF_INSTALL_DIR>/actions directory.
-r RESTRICTOR_DIR, --crst=RESTRICTOR_DIR
(optional) The path to a directory of the action's
restrictor files. Note: If this option is not set,

kikact tries to retreive its value from an environment variable named KIKONF_RST.

4.2 HELP AND LOGS

>kikact -h

-h, --help *show this help message and exit.*

>kikact -h –kikact *or*
>kikact -h –kikact*, shows kikact (or kikarc), extended help.*

This option shows help on all options.

-H HELP, --HELP=HELP Extended help.

This options shows the complete kikact document ation.

-v VERBOSE, --verbose=VERBOSE *The verbose level. A number from 0 to 30.*

The highter this number is the more verbose is shown.
For instance -v3 shows a block of text per Action injected or extracted.
Example :
>kikupd --kikarc c.xml application -v3 or
>kikupd --kikarc c.xml application -v 3

4.3 PATHS

Using the options: –kikact and –kikarc with the kikupd command,
the same path resolution phylosophy as the one implemented for the kikact and kikarc commands applies.

-C KIKONF_ATTRS, --cattrs=KIKONF_ATTRS
(optional) The path to a custom kikonf.attrs file.
When you don't want to use the default one into the <KIKONF_INSTALL_DIR>/conf directory.
Note: If this option is not set, kikact tries to retreive its value from an environment variable named KIKONF_CATTRS.
If neither of the option or the environment variable are set kikact use the default kikonf.attrs file into the <KIKONF_INSTALL_DIR>/conf directory.

You may use this option if you want to specify a custom kikonf.attrs file, covering another scope of softwares. This would be the case if you supports more than one binary for the same software.

For more information about the kikonf.attrs file see the kikonf core documentation.

-c ACTION_DIR, --cxml=ACTION_DIR
(optional) The path to the directory of the action files. If not given, sample action files are retreived from the <KIKONF_INSTALL_DIR>/actions directory.
Note: If this option is not set, kikact tries to retreive its value from an environment variable named KIKONF_CXML.
If neither the option or the environment variable are set kikact use the default <KIKONF_INSTALL_DIR>/actions directory.

An Action file is an xml file wich defines the settings for a singular Action.
Actually it is where you put your setup for a singular Action.
For more information about Action files see (chapter above "overview" or) the kikonf core documentation.

The default <KIKONF_INSTALL_DIR>/actions directory contains a sample for each existing Actions.
Beware that if no custom directory is found these samples are run for each corresponding Actions.

Note:
The structue of a custom action directory is:

```
<MY_CUSTOM_ACTION_DIR>
        <BAL1>.xml
        <BAL2>.xml
        <BALn>.xml
    ...
```

For more information about custom Action directory see the kikonf core documentation.

-r RESTRICTOR_DIR, --crst=RESTRICTOR_DIR
(optional) The path to a directory of the action's restrictor files.
Note: If this option is not set, kikact tries to retreive its value from an environment variable named KIKONF_RST.

Generally you very less likely need to use restritor directory than you may need to use custom action directory.

You may want to use restrictor directories when occures the need to restrict a singular user's (or group) rights over a singular action.

This would restrict his access to (whole or) a specific tag(s) or Attribute(s).
You can see a restrictor file as a mask within a few fields are allowed.

Actually restrictor file are Action descriptor file, customized .

You can take a descriptor file for a given Action from:
<PLUGINS_DIR>/actions/<CATEGORY>/<ACTION_NAME>/by/WHO/ACT_INF/action.xml
and place it into your restrictor directory.
Obviously if not customized it would restrict nothing.

Note:
As you can see in the path (<PLUGINS_DIR>/actions) above have a certain structure , your restrictor directory must reflect this structure.

<MY_RESTRICTOR_DIR>
<CATEGORY>
<ACTION_NAME>
by
<WHO>
action.xml
For more information about how to make restrictor files see the kikonf core documentation.

5 ANNEX 1 THE BAL (BASIC ACTION LOCATOR)

5.1 SYNTAX

BAL (Basic Action Locator)	IN COMMANDS (kikact, kikarc,...)	MAPPED ACTION FILES	MAPPED ACTION RESTRICTOR FILES	ACTION TAG IN CUSTOM XML FILE
CATEGORY.ACTION [.by.WHO]	BAL[.$LABEL]	BAL[.$LABEL].xml **Top tag:** ACTION	BAL.xml **Top tag:** ACTION	**Action tag:** ACTION **Bal attribute:** BAL

5.2 SAMPLES

BAL	IN COMMANDS	SAMPLES	MAPPED ACTION CONFIGURATION FILES	MAPPED ACTION RESTRICTOR FILES	ACTION TAG IN CUSTOM XML FILES
CATEGORY.ACTION	BAL	was.datasrc	was.datasrc.xml or was.datasrc.$LABEL.xml (all labelized action files are treated as well) **Top tag:** datasrc	was.datasrc.xml **Top tag:** datasrc	**Action tag:**datasrc **Bal attribute:** was.datasrc
	BAL.$LABEL	was.datasrc.$1 (or tmc.datasrc. $invoices	was.datasrc.$1.xml **Top tag:** datasrc		
Advanced (use it if you know what you are doing):					
CATEGORY.ACTION .by.WHO	BAL	was.datasrc.by.shandra	was.datasrc.by.shandra.xml or was.datasrc.by.shandra. $LABEL.xml (all labelized action files are treated as well)	was.datasrc.(+) by.shandra.xml **Top tag:** datasrc	**Action tag:**datasrc **Bal attribute:** was.datasrc.(+) by.shandra
	BAL.$LABEL	was.datasrc.by.shandra.$1	was.datasrc.by.shandra. $1.xml		

Note:

If you launch a command (ex: kikact) with the -g (or --category) CATEGORY option (ex: -g was), you do not need to prefix each BAL with CATEGORY.
kikact -g was jvm,datasrc,vhost instead of
kikact was.jvm,was.datasrc,was.vhost

Advanced trick:

In the advanced part, the clause "by" in CATEGORY.ACTION.by.WHO is used when one action has multiple providers (or authors).
ex: The action was.jvm may have been writen by kikonf, but another version more adapted for specific needs may have also been provided by shandra.

WHO is the name of this provider.
Action plugins are clasified per category/action/provider.
The kikact option -P shows a detailled list of all plugins per providers.

A directory entitled by WHO exists at
<KIKONF_INSTALL_DIR>/plugins/actions/<CATEGORY>/<ACTION>/by/<WHO>.

If the file <KIKONF_INSTALL_DIR>/plugins/actions/<CATEGORY>/<ACTION>/default.txt is filled with WHO.
WHO becomes the default plugin for this action and the clause "by" is no more needed.

On delivry of the kikonf binary, **kikonf** is this default provider.

6 ANNEXE 2: KIKUPD ALL OPTIONS

>kikact -h

Usage:

...

Options:

-h, --help
show this help message and exit
-v VERBOSE, --verbose=VERBOSE
The verbose level.
-H, --HELP
Shows kikonf extended options.
-o, --console
(default False) Launch an interactive terminal menu to provide interactive interface for the xml file. Note: --console (-o), --update (-u), --create (-n) and --remove (-e) options are mutually exclusives.
-u, --update
(default False) Given an picpaths list of arguments will update the required xml targets. Note: --console (-o), --update (-u), --create (-n) and --remove (-e) options are mutually exclusives.
-n, --create
(default False) Given an picpaths list of arguments will create the required xml targets. Note: --console (-o), --update (-u), --create (-n) and --remove (-e) options are mutually exclusives.
-x, --force (default False) In conjunction with the create option will allow the creation of nodes with unchecked attribute values.
-e, --remove
(default False) Given an picpaths list of arguments will destroy the required xml targets. Note: --console (-o), --update (-u), --create (-n) and --remove (-e) options are mutually exclusives.
--kikact=KIKACT
Allows the use of the kikact extended options. The expected value for this option is a BAL (Basic Action Locator, e.g.:wls.jvm). Note: --kikact, --kikarc, --file_user (-F) and --info (-i) are mutually exclusives.
--kikarc=KIKARC
Allows the use of the kikarc extended options. The

expected value for this option is a custom action file. Note: --kikact, --kikarc, --file_user (-F) and --info (-i) are mutually exclusives.

-F FILE_USER, --file_user=FILE_USER

The xml file when not using the --kikact,--kikarc options. Note: this option is not allowed using the --kikact,--kikarc extended options. Note: --kikact, --kikarc, --file_user (-F) and --info (-i) options are mutually exclusives.

-s ATTR_SEPARATOR, --attr_separator=ATTR_SEPARATOR

Separator when multiple Attributes are returned. Option --attr_separator (-s) is allowed when not using: --console (-o), --update (-u), --create (-n) and --remove (-e) options.

-z PICPATH_ATTR_SEPARATOR, --picpath_attr_separator=PICPATH_ATTR_SEPARATOR

Attribut Separator but for the picpath expression. Option --picpath_attr_separator (-z) is allowed when not using: --console (-o), --update (-u), --create (-n) and --remove (-e) options.

-S NODE_SEPARATOR, --node_separator=NODE_SEPARATOR

Separator when multiple nodes are returned.. Option --attr_separator (-s) is allowed when not using: --console (-o), --update (-u), --create (-n) and --remove (-e) options.

--print

(default False) If used the resulting xml file is printed to the output.

-i INFO, --info=INFO

The expected value for this option is a BAL (Basic Action Locator, e.g.:wls.jvm). Note: --kikact, --kikarc, --file_user (-F) and --info (-i) are mutually exclusives.

-p, --plugins

List all the plugin installed into this kikonf installation.

-P APLUGIN, --aplugin=APLUGIN

The expected value for this option is a BAL (Basic Action Locator, e.g.:wls.jvm). Shows the plugin information for this Action.

Advanced options:

--overwrite

Will overwrite the original files.

-f TO_FILE, --to_file=TO_FILE

A path to a file where to write the resulting xml file.

--no_dft

Attributes whom value matches to the res/descriptor's default value for this attribute are not shown !

-D FILE_DESC, --file_desc=FILE_DESC

In conjunction with the --file_user (-F) option.An xml descriptor file.

-R FILE_REST, --file_rest=FILE_REST

In conjunction with the --file_user (-F) and the --file_desc (-D) options. An xml restrictor file. (todo)

-l, --cap_sensitive

(default True) Is this xml fie cap sensitive. Not allowed with --kikact or --kikarc options.

-X, --xforce

(default False) force writing with no check and regardingless to descriptor file. BE CAUTIOUS !

--dont_show_files

In conjunction with --console option. If guiven wont show menu entries for Action, Descriptor and Restrictor file !

Kikonf extended generic options:

The following options are the ones supported when using one of the --kikact, --kikarc or --info options.

-C KIKONF_ATTRS, --cattrs=KIKONF_ATTRS

(optional) The path to a custom kikonf.attrs file. When you don't want to use the default one into the <KIKONF_INSTALL_DIR>/conf directory. Note: If this option is not set, kikact tries to retreive its value from an environment variable named KIKONF_CATTRS. If neither the option or the environment variable are set kikact use the default kikonf.attrs file into the <KIKONF_INSTALL_DIR>/conf directory.

-c ACTION_DIR, --cxml=ACTION_DIR

(optional) The path to the directory of the action files. If not given, sample action files are retreived from the <KIKONF_INSTALL_DIR>/actions directory. Note: If this option is not set, kikact tries to retreive its value from an environment variable named KIKONF_CXML. If neither the option or the environment variable are set kikact use the default <KIKONF_INSTALL_DIR>/actions directory.

-r RESTRICTOR_DIR, --crst=RESTRICTOR_DIR

(optional) The path to a directory of the action's restrictor files. Note: If this option is not set, kikact tries to retreive its value from an environment variable named KIKONF_RST.

7 TRADEMARKS:

Db2 is a registred trademark of International Business Machines Corporation.

Oracle and Java are registred trademarks of Oracle and/or its affiliates.

Other names may be trademarks of their respective owners.