

Repoz xpc commands

Version : 1.0

Date : 19/08/2006

Author : Patrick Germain Placidoux

Copyright (c) 2007-2008, Patrick Germain Placidoux

All rights reserved.

SUMMARY

1	1 Objective	3
2	2 Introduction	4
3	3 Getting Help	5
4	4 The xpc processor	6
5	5 The xpath mode	7
5.1	5.1 The Picpath Expression	8
5.1.1	5.1.1 Working with text	8
5.2	5.2 The list command	9
5.3	5.3 The Change Directory command	11
5.4	5.4 The Update command	13
5.5	5.5 The Create command	16
5.6	5.6 The Remove command	18
5.7	5.7 The Copy command	19
5.8	5.8 The Mode command	21
5.9	5.9 The set command	23
5.10	5.10 The Show and Save command	24
5.11	5.11 Annex / the xpath commands generic options	25
6	6 The Xql mode	27
6.1	6.1 The Request path	29
6.2	6.2 Select operations	30
6.3	6.3 Create operation	38
6.4	6.4 Update operation	41
6.5	6.5 Delete operation	43
6.6	6.6 Duplicate operation	45
6.7	6.7 Insert operation	49

1 OBJECTIVE

This document describes the xpc processor commands

2 INTRODUCTION

The set of available commands for a processor depends on the processor and the current mode.

The samples below use the files into the directory: <REPOZ_INSTALLATION_DIR>/samples

Note about the sample directory:

To run the Repoz samples of this documentation from a Kikonf installation , replace the directory :
<repoz_intallation_dir>/samples by: **<kikonf_intsallation_dir>/samples/repoz.**

3 GETTING HELP

The following help commands may be type in any mode.

help shows this help: a long help for the commands available for the mounted processor for the current mode.

h (or help) <command> shows help for this commands available for the mounted processor for the current mode.

h show a short summary of the commands available for the mounted processor for the current mode.

H shows a short summary of the globally available commands (for all mode, all processor).

HELP shows a long help for the globally available commands.

H (or HELP) <command> shows help for this globally available commands.

4 THE XPC PROCESSOR

The xpc processor is used to work with xml files.

An xpc processor is mounted using the xpc command like this :

```
:-> xpc -F <repoz_intallation_dir>/samples/test.xml
```

When an xml (or xpc) processor is mounted, the path is set to the first tag of the xml file.

e.g.:

```
:test:/tag1>
```

The ":" at the beginning of the prompt shows that the first mode by default is the picpath mode.

5 THE XPATH MODE

The RepoZ xpath is an xpath like implementation.
 The xpath mode is the default mode for the xpc processor.
 This xpath like implementation supports 8 commands:

ls, cd, upd, new, rm, cp, mv, set

These commands accept one or more **picpath** arguments.

To swich to the xpath mode :

- Switch to the xpath mode
e.g. (from the python : ? mode):
?test:/sometag>: (or type "mode xpath")
 :test:/sometag>
- From any mode Create and Mount an xpc processors
e.g.:
:test:/sometag>:xpc -F <REPOZ_INSTALLATION_DIR>/samples/test.xml -a mypc
New processor with alias:mypc, created. (use mount mypc, to mount it)
 :test:/tag1>
- Mount your new processor (if not already mounted)
e.g.:
:test:/sometag>:mount mypc
 :mypc:/tag1>

5.1 THE PICPATH EXPRESSION

This chapter explains the picpath arguments syntax.

The picpath syntax is:

TAG[@ATTR=VALUE[,@ATTR=VALUE]],@ATTR Tag sep:/
 TAG[@ATTR=VALUE[,@ATTR=VALUE]],@ATTR[,@ATTR]
 TAG[@ATTR=VALUE[,@ATTR=VALUE]],@*

e.g. 1:

>ls tag2 # Lists the content of tag2 and its attributes.

>ls tag2,@attr1,@attr2 # Lists the attr1 and attr2 attributes of tag2.

>ls tag2@attr2=i/tag4/tag5 # Selects for the tag2 only the node(s) with attribute attr2 equal to i,
 # the sub child node(s) tag5.

e.g. 2:

>ls tag2@attr2=i/tag4/tag5@attr1=True,@attr1,@attr3

Selects for the tag2, node(s) with attribute attr2 equal to i,
 # the sub child node(s) tag5 with attribute attr1 equal to True,
 # and shows only the attributes attr1 and attr3 of this last.

5.1.1 Working with text

The picpath syntax is:

TAG[@ATTR=VALUE[,@ATTR=VALUE]],@%text

e.g.:

>ls tag3@%text # shows the text content of tag3.

Note:

For more information about picpath check out the documentation of the picxml project on www.sourceforge.net. or into the directory <REPOZ_INSTALL_PATH>/doc, or into the directory <KIKONF_INSTALL_PATH>/doc if you are using Kikonf, or on the www.kikonf.com site.

5.2 THE LIST COMMAND

Command: **ls**

The ls command, lists one or more picpath(s).

Syntax:

ls <picpath> [<picpath>]

- With no picpath at all, ls list the current Node content

e.g.:

:test:/tag1>ls

/tag1> attr1:a attr2:b

tag2

tag3

tag2

tag2

- Listing child nodes

e.g. 1:

:test:/tag1>ls 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

e.g. 2:

:test:/tag1>ls tag3

/tag1/tag3> attr1:f attr2:g attr3:A value with spaces !

- Listing a set of attribute for more multiples nodes.

e.g. 3:

:test:/tag1>ls tag2@attr2

d,i,p

- Listing a set of picpath

e.g. 4:

```
:test:/tag1>ls tag2 tag2@attr2 tag3  
/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
```

d,i,p

```
/tag1/tag3> attr1:f attr2:g attr3:A value with spaces !
```

- Listing a node at a specific index

e.g. 5:

```
:test:/tag1>ls 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
```

```
:test:/tag1>ls tag2[0]
```

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

```
:test:/tag1>ls tag2[1]
```

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

```
:test:/tag1>ls tag2[2]
```

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

- Listing a node text

e.g.:

```
:test:/tag1>ls tag3@%text  
aaaa1;bbbbbbbbb2
```

5.3 THE CHANGE DIRECTORY COMMAND

Command: **cd**

The cd command aka changes directory command allows to switch from a node to another.

Note:

This command in fact operates a ls command in background on the picpath argument, and then cd into the returned node.

This requires that at least one and only one node is returned by the request.

Syntax:

cd <picpath>

- Cding a simple node

:test:/tag1>cd tag3

/tag1/tag3> attr1:f attr2:g attr3:A value with spaces !

:test:/tag1/tag3>ls

/tag1/tag3> attr1:f attr2:g attr3:A value with spaces !

- Using .. to move back

:test:/tag1/tag3>cd ..

/tag1> attr1:a attr2:b

tag2

tag3

tag2

tag2

- Cding a complex request

:test:/tag1>cd tag2@attr2=i

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

tag4

- Using .. to ls back

```
:test:/tag1/tag2>ls ..  
/tag1> attr1:a attr2:b  
tag2  
tag3  
tag2  
tag2
```

```
:test:/tag1/tag2>ls ..@attr2  
b
```

```
:test:/tag1/tag2>ls ../tag2@attr2  
d,i,p
```

- Cding sub child nodes

```
e.g.:  
:test:/tag1/tag2>ls  
/tag1/tag2> attr1:h attr2:i attr3:j attr4:k  
tag4
```

```
:test:/tag1/tag2>cd tag4/tag5  
/tag1/tag2/tag4/tag5> attr1:True attr2:m attr3:n
```

```
e.g.:  
:test:/tag1/tag2/tag4/tag5>ls  
/tag1/tag2/tag4/tag5> attr1:True attr2:m attr3:n
```

```
e.g.:  
:test:/tag1/tag2/tag4/tag5>ls ../tag5@attr1  
True
```

5.4 THE UPDATE COMMAND

Command: **upd**

The update command, updates one or more picpath(s).

Syntax:

```
upd <picpath>[@attr=value,@attr=value,...] [...]
```

- Simple update

e.g.:

```
:test:/tag1>ls tag3
```

```
/tag1/tag3> attr1:f attr2:some values with spaces attr3:A value with spaces !
```

```
:test:/tag1>upd tag3[@attr1=bbb]
```

```
updated node:/tag1/tag3>
```

```
From attr1:f To attr1:bbb
```

- Updating more than one attributes using values with spaces

e.g.:

```
:test:/tag1>ls tag3
```

```
/tag1/tag3> attr1:bbb attr2:some values with spaces attr3:A value with spaces !
```

```
:test:/tag1>upd 'tag3[@attr1=c,@attr2=some values with spaces]'
```

```
updated node:/tag1/tag3>
```

```
From attr2:some values with spaces To attr2:some values with spaces
```

```
From attr1:bbb To attr1:c
```

```
:test:/tag1>ls tag3
```

```
/tag1/tag3> attr1:c attr2:some values with spaces attr3:A value with spaces !
```

- Updating using more than one picpath

e.g.:

```
:test:/tag1>upd 'tag3[@attr1=a,@attr2=b]' tag3[@attr3=c]
```

```
updated node:/tag1/tag3>
```

```
From attr2:some values with spaces To attr2:b
```

```
From attr1:c To attr1:a
```

updated node:/tag1/tag3>
 From attr3:A value with spaces ! To attr3:c

:test:/tag1>ls tag3
 /tag1/tag3> attr1:**a** attr2:**b** attr3:**c**

- Updating using over multiple nodes

e.g.:
:test:/tag1>upd tag2[@attr2=aaa]
 updated node:/tag1/tag2>
 From attr2:d To attr2:aaa

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

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

:test:/tag1>ls tag2
 /tag1/tag2> attr1:b attr2:aaa attr3:e attr4:None

/tag1/tag2> attr1:b attr2:aaa attr3:j attr4:k
 tag4

/tag1/tag2> attr1:b attr2:aaa attr3:None attr4:None

- Updating using seach criterias

e.g.:
:test:/tag1>ls tag2@attr2=i/tag4/tag5
 /tag1/tag2/tag4/tag5> attr1:True attr2:m attr3:n

:test:/tag1>upd tag2@attr2=i/tag4/tag5[@attr1=False]
 updated node:/tag1/tag2/tag4/tag5>
 From attr1:True To attr1:False

:test:/tag1>ls tag2@attr2=i/tag4/tag5
 /tag1/tag2/tag4/tag5> attr1:**False** attr2:m attr3:n

- Updating a 's text

e.g.:

```
:test:/tag1>ls tag3@%text  
aaaa1;bbbbbbb2
```

```
:test:/tag1>upd 'tag3[@%text=new value for;this;texts]'  
updated node:/tag1/tag3>
```

```
From %text:
```

```
    aaaa1  
    bbbbbbbb2
```

```
To %text:
```

```
    new value for  
    this  
    texts
```

```
:test:/tag1>ls tag3@%text  
new value for;this;texts
```

Please note that working with text Carriage Returns are replaced by ";" in both: upd and ls command.

5.5 THE CREATE COMMAND

Command: **new**

The new command, creates one or more new nodes.

Syntax:

new <picpath> [<picpath>]

- Simple node creation

e.g.:

:test:/tag1>ls 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

:test:/tag1>new tag2[@attr1=a,@attr3=b]

Created node: /tag1/tag2>

updated node:/tag1/tag2>

From attr3:None To attr3:b

From attr1:None To attr1:a

:test:/tag1>ls 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

/tag1/tag2> attr1:a attr2:None attr3:b attr4:None <---

- Node creation with complex search criteria

e.g.:

:test:/tag1>ls tag2@attr2=i/tag4/tag5

/tag1/tag2/tag4/tag5> attr1:True attr2:m attr3:n


```
:test:/tag1>new tag2@attr2=i/tag4/tag5[@attr1=False,@attr3=b]
```

```
Created node: /tag1/tag2/tag4/tag5>
```

```
updated node:/tag1/tag2/tag4/tag5>
```

```
From attr3:None To attr3:b
```

```
From attr1:None To attr1:False
```

```
:test:/tag1>ls tag2@attr2=i/tag4/tag5
```

```
/tag1/tag2/tag4/tag5> attr1:True attr2:m attr3:n
```

```
/tag1/tag2/tag4/tag5> attr1:False attr2:None attr3:b <--
```

- Node creation with multiple picpaths

e.g.:

```
:test:/tag1>new tag2[@attr1=a,@attr3=b] tag2@attr2=i/tag4/tag5[@attr1=False,@attr3=b]
```

```
Created node: /tag1/tag2>
```

```
updated node:/tag1/tag2>
```

```
From attr3:None To attr3:b
```

```
From attr1:None To attr1:a
```

```
Created node: /tag1/tag2/tag4/tag5>
```

```
updated node:/tag1/tag2/tag4/tag5>
```

```
From attr3:None To attr3:b
```

```
From attr1:None To attr1:False
```

```
:test:/tag1>ls tag2 tag2@attr2=i/tag4/tag5
```

```
/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
```

```
/tag1/tag2> attr1:a attr2:None attr3:b attr4:None <---
```

```
/tag1/tag2/tag4/tag5> attr1:True attr2:m attr3:n
```

```
/tag1/tag2/tag4/tag5> attr1:False attr2:None attr3:b <---
```

```
/tag1/tag2>
```

5.6 THE REMOVE COMMAND

Command: **rm**

The remove command, removes one or more nodes.

Syntax:

rm <picpath> [<picpath>]

- Simple remove

e.g.:

:test:/tag1>ls

/tag1> attr1:a attr2:b

tag2

tag3

tag2

tag2

:test:/tag1>rm tag3

Removed: /tag1/tag3<

:test:/tag1>ls

/tag1> attr1:a attr2:b

tag2

tag2

tag2

- Removing more than one node

e.g.:

:test:/tag1>rm tag2

Removed: /tag1/tag2<

Removed: /tag1/tag2<

Removed: /tag1/tag2<

:test:/tag1>ls

/tag1> attr1:a attr2:b

:test:/tag1>

5.7 THE COPY COMMAND

Command: **cp**

The cp command, copies one or more source node(s) to one or more targets nodes.

Syntax:

cp <picpath_source> <picpath_destination>

- Simple copy

:test:/tag1>ls 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

:test:/tag1>cp tag2@attr2=i ..

Adding child:/tag1/tag2

:test:/tag1>ls 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

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

:test:/tag1>

- Copy resulting in multiple new nodes

e.g.:

:test:/tag1>ls 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
```

```
:test:/tag1>cp tag2 ..
```

```
Adding child:/tag1/tag2
```

```
Adding child:/tag1/tag2
```

```
Adding child:/tag1/tag2
```

```
:test:/tag1>ls 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
```

```
/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 <---
```

5.8 THE MODE COMMAND

Command: **mv**

The mv ommnad copies one or more source node(s) to one or more targets nodes and deletes all the source nodes.

Syntax:

mv <picpath_source> <picpath_destination>

- Move resulting in multiple new nodes

e.g.:

:test:/tag1>ls

/tag1> attr1:a attr2:b

tag2

tag3

tag2

tag2

:test:/tag1>ls 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

:test:/tag1>mv tag3 tag2

EEpicxml: Not allowed child:tag3 for Node:tag2. Allowed children are:'tag4', or use force.

:test:/tag1>mv tag3 tag2 -X

Adding child:/tag1/tag2/tag3

Adding child:/tag1/tag2/tag3

Adding child:/tag1/tag2/tag3

:test:/tag1>ls tag2

/tag1/tag2> attr1:c attr2:d attr3:e

tag3 <---

/tag1/tag2> attr1:h attr2:i attr3:j

tag4

tag3 <---

/tag1/tag2> attr1:o attr2:p attr3:None
tag3 <---

5.9 THE SET COMMAND

Command: set

The set comand just set a value to a guiven attribute for the current path.

Syntax:

set <attr> = <value>

- Simple sample

e.g.:

:test:/tag1>ls

/tag1> attr1:a attr2:b

tag2

tag3

tag2

tag2

:test:/tag1>set attr1 = b

:test:/tag1>ls

/tag1> attr1:b attr2:b

tag2

tag3

tag2

tag2

- Sample with spaces

e.g.:

:test:/tag1>set attr1 = 'a new value'

:test:/tag1>ls

/tag1> attr1:a new value attr2:b

tag2

tag3

tag2

tag2

5.10 THE SHOW AND SAVE COMMAND

The show (or save) command, show (or save) the content of the file managed by the current processor.

e.g.:

```
:>xpc -F <REPOZ_INSTALLATION_DIR>samples/test.xml
```

New processor with alias:test, created and mounted.

```
:test:/tag1>show
```

```
<tag1 attr1='a' attr2='b'>
  <tag2 attr1='c' attr2='d' attr3='e'>
    <tag3 attr1='f' attr2='g' attr3='A value with spaces !'>
      aaaa1
      bbbbbbbb2
    </tag3>
  <tag2 attr1='h' attr2='i' attr3='j' attr4='k'>
    <tag4>
      <tag5 attr1='True' attr2='m' attr3='n'>
    </tag4>
  </tag2>
  <tag2 attr1='o' attr2='p'>
</tag1>
```

The command save **-all (-a)** save all processors.

e.g.:

```
:>xpc -F <REPOZ_INSTALLATION_DIR>samples/test.xml -s -f c:\temp\mytest.xml
```

New processor with alias:mytest, created and mounted.

File:c:\temp\mytest.xml saved !

```
:mytest:/tag1>save -a
```

File:c:\temp\mytest.xml saved !

```
:test:/tag1>
```


5.11 ANNEX / THE XPATH COMMANDS GENERIC OPTIONS

The xpath commands support the following options.

Usage:

Supported xpath commands are: cd, ls, upd, new, rm, cp, mv, set, save, show

For help on command type: h (or help) <command>

Options:

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

-v VERBOSE, --verbose=VERBOSE
The verbose level.

-H, --HELP Shows the processor extended options.

-X, --force (default False) In conjunction with the new command option will allow the creation of nodes with unchecked attribute values.

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

-t TEXT_SEPARATOR, --text_separator=TEXT_SEPARATOR
Separator when multiple lines of text are returned (default: ;).
Option --text_separator (-t) 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
Attribute Separator but for the picpath expression (default: ;).
Option --picpath_attr_separator (-z) is allowed when not using: --console (-o), --update (-u), --create (-n) and --remove (-e) options.

-T PICPATH_TEXT_SEPARATOR, --picpath_text_separator=PICPATH_TEXT_SEPARATOR
Text item Separator but for the picpath expression (default: ;). Option --picpath_text_separator (-T) is allowed when not using: --console (-o), --update

- print** (-u), --create (-n) and --remove (-e) options.
(default False) If used the resulting xml file is printed to the output.
- x, --xforce** (default False) force writing with no check and regardless to descriptor file. BE CAUTIOUS !

6 THE XQL MODE

The Repoz xql is an xql like implementation.

To switch to the xql mode :

- Switch to the xql mode
e.g. (from the python : ? mode):
?test:/sometag>% (or type "mode ql")
 %test:/sometag>
- From any mode Create and Mount an xpc processors
e.g.:
%test:/sometag>:xpc -F <REPOZ_INSTALLATION_DIR>/samples/test.xml -a mypc
New processor with alias:mypc, created. (use mount mypc, to mount it)
 %test:/tag1>
- Mount your new processor (if not already mounted)
e.g.:
%test:/sometag>:mount mypc
 %mypc:/tag1>

This xql like implementation supports **5 operations:**
select, create, update, delete, duplicate, insert

Foreach operation there is one or more commands.

Select operation :

Main commands: **select, xselect**

Advanced commands: cselect, ccselect, rselect, crselect

Create operation :

Main command: **create**

Advanced commands: ccreate, rcreate

Delete operation :

Main command: **delete**

Update operation :

Main command: **update**

Duplicate operation :

Main command: **duplicate**

Insert operation :

Main command: **insert**

Each command supports a line argument called the **xql request**.

6.1 THE REQUEST PATH

All operations are requested to the **current node**.

e.g.:

```
:test:/tag1>% # Entering xql mode.
```

```
%test:/tag1>select * at tag1
```

	<i>*tun</i>	<i>attr2</i>	<i>attr1</i>	<i>*text</i>
<i>/tag1</i>	<i>I1</i>	<i>b</i>	<i>a</i>	<i>[]</i>

```
%test:/tag1>: # Changind path from the xpath mode.
```

```
:test:/tag1>cd tag2@attr2=i
```

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

```
:test:/tag1/tag2>% # Going back into xql mode.
```

```
%test:/tag1/tag2>select * at tag1 # Shows nothing because there are no tag1 nodes below tag2 nodes.
```

```
%test:/tag1/tag2>select * at tag2
```

	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>
<i>/tag1/tag2</i>	<i>I4</i>	<i>k</i>	<i>i</i>	<i>j</i>	<i>h</i>

6.2 SELECT OPERATIONS

The select operation, selects one or more node(s) at a set of tag(s) and regarding an optional where clause.

Syntax:

select O_WHAT at F_TAGS where F_ATTRS

- simple select

%test:/tag1>select * at tag1

	<i>*tun</i>	<i>attr2</i>	<i>attr1</i>	<i>*text</i>
<i>/tag1</i>	<i>11</i>	<i>b</i>	<i>a</i>	<i>[]</i>

- The select deep search behaviour, comparing to the Repos xpath ls implementation

e.g. 1:

:test:/tag1>ls

/tag1> attr1:a attr2:b

tag2

tag3

tag2

tag2

:test:/tag1>%

%test:/tag1>xselect * at tag3 *# Because xql will search the whole descendant node from the current # path, the tag patern do not have to be successive (tag1/tga3) like in # the Repos xpath implemenation.*

/tag1/tag3

**tun:13*

<tag3 attr3="A value with spaces !" attr1="f" attr2="g">

aaaa1

bbbbbb2

</tag3>

e.g. 2:

```
%test:/tag1>xselect * at tag5
```

```
/tag1/tag2/tag4/tag5
```

```
*tun:16
```

```
<tag5 attr2="m" attr3="n" attr1="True">
</tag5>
```

- Selecting more than one tag

e.g.:

```
:test:/tag1>%
```

```
%test:/tag1>xselect * at tag3,tag2
```

```
/tag1/tag2
```

```
*tun:12
```

```
<tag2 attr4="None" attr3="e" attr1="c" attr2="d">
</tag2>
```

```
/tag1/tag3
```

```
*tun:13
```

```
<tag3 attr3="A value with spaces !" attr1="f" attr2="g">
  aaaa1
  bbbbbbbb2
</tag3>
```

```
/tag1/tag2
```

```
*tun:14
```

```
<tag2 attr4="k" attr3="j" attr1="h" attr2="i">
  <tag4 >
    <tag5 attr2="m" attr3="n" attr1="True">
    </tag5>
  </tag4>
</tag2>
```

```
/tag1/tag2
```

```
*tun:17
```

```
<tag2 attr4="None" attr3="None" attr1="o" attr2="p">
</tag2>
```

- Straightly jumping to a tag unique name (*tun)

As you can see, the symbol ***tun** demonstrate that each node has a tag unique name, this is the unique identifier of the node through the xml file.

e.g.:

%test:/tag1>select * at *tun=I7

	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>ttr1</i>
<i>/tag1/tag2</i>	<i>I7</i>	<i>None</i>	<i>p</i>	<i>None</i>	<i>o</i>

e.g.:

%test:/tag1>select * at tag3,*tun=I7

	<i>*tun</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>	<i>*text</i>	
<i>/tag1/tag3</i>	<i>I3</i>	<i>g</i>	<i>A value with spaces !f</i>		<i>[aaaa1,bbbbbbbb2]</i>	
	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>	<i>text</i>
<i>/tag1/tag2</i>	<i>I7</i>	<i>None</i>	<i>p</i>	<i>None</i>	<i>o</i>	<i>[]</i>

- Selecting a set of attributes

e.g.:

%test:/tag1>select attr1,attr3 at tag2

	<i>*tun</i>	<i>attr1</i>	<i>attr3</i>	<i>*text</i>
<i>/tag1/tag2</i>	<i>I2</i>	<i>c</i>	<i>e</i>	<i>[]</i>
<i>/tag1/tag2</i>	<i>I4</i>	<i>h</i>	<i>j</i>	<i>[]</i>
<i>/tag1/tag2</i>	<i>I7</i>	<i>o</i>	<i>None</i>	<i>[]</i>

- Selecting using the where close

e.g.:

%test:/tag1>select * at tag2

	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>
<i>/tag1/tag2</i>	<i>I2</i>	<i>None</i>	<i>d</i>	<i>e</i>	<i>c</i>
	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>


```

/tag1/tag2
      I4          k          i          j          h
      *tun        attr4      attr2      attr3      attr1
/tag1/tag2
      I7          None      p          None      o
    
```

%test:/tag1>select * at tag2 where attr2=i

```

      *tun        attr4      attr2      attr3      attr1
/tag1/tag2
      I4          k          i          j          k          h
    
```

- Selecting using the where clause in conjunction with Hierarchical Attributes (aka Hierarchical where clause)

e.g.:

%test:/tag1>select * at tag2,tag3 where attr2=i or attr2=g

A where clause is processed in two steps:

- 1st step : The set of nodes matching the "at" clause is retrieved (e.g.: at tag2,tag3).
- 2nd step : This set of nodes is checked again upon the "where" clause conditions (e.g.: where attr2=i or attr2=g).

When the Attributes of the where clause are not Hierarchical all the nodes found in the first step are checked for the requested attributes.

e.g.:

%test:/tag1>select * at tag2,tag3 where attr2=i or attr2=g

```

      *tun attr2      attr3      attr1      *text
/tag1/tag3
      I3  g          A value with spaces !f      [aaaa1,bbbbbbbb2]

      *tun attr4      attr2      attr3      attr1      *text
/tag1/tag2
      I4  k          i          j          h          []
    
```

In this selection the nodes tag3 and tag2 are retrieved because both have an Attribute named attr2 satisfying the where condition.

Now using Hierarchical Attributes will allow to accept from the first extraction, only the nodes who match one or more ancestors defined by the Hierarchical Attributes.

e.g.:

```
%test:/tag1>select * at tag5 where tag2@attr2=i
```

	<i>*tun</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>	<i>*text</i>
<i>/tag1/tag2/tag4/tag5</i>	<i>I6</i>	<i>m</i>	<i>n</i>	<i>True</i>	<i>[]</i>

Here the tag5 node found for the "at" clause is accepted because it has an ancestor:tag2, (actually its grand father: tag2/tag4/tag5) and this ancestor'Attribute attr2 satisfies the request (=i).

Just for information, this is the content under tag1:

```
%test:/tag1>xselect * at tag1
```

```
/tag1
```

```
*tun:11
```

```
<tag1 attr2="b" attr1="a">
  <tag2 attr2="d" attr3="e" attr1="c">
    </tag2>
    <tag3 attr3="A value with spaces !" attr1="f" attr2="g">
      aaaa1
      bbbbbbbb2
    </tag3>
    <tag2 attr4="k" attr3="j" attr1="h" attr2="i">
      <tag4 >
        <tag5 attr2="m" attr3="n" attr1="True">
          </tag5>
        </tag4>
      </tag2>
    <tag2 attr4="None" attr3="None" attr1="o" attr2="p">
    </tag2>
  </tag1>
```

- Selecting using complex multiple where closes

```
%test:/tag1>select * at tag2
```

	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>	<i>*text</i>
<i>/tag1/tag2</i>	<i>I2</i>	<i>None</i>	<i>d</i>	<i>e</i>	<i>c</i>	<i>[]</i>

	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>	<i>*text</i>
<i>/tag1/tag2</i>	<i>I4</i>	<i>k</i>	<i>i</i>	<i>j</i>	<i>h</i>	<i>[]</i>

```

      *tun      attr4      attr2      attr3      attr1      *text
/tag1/tag2
      I7       None       p         None       o         []

```

%test:/tag1>select * at tag2 where attr2=i or attr2=p

```

      *tun      attr4      attr2      attr3      attr1      *text
/tag1/tag2
      I4       k         i         j         h         []

```

```

      *tun      attr4      attr2      attr3      attr1      *text
/tag1/tag2
      I7       None       p         None       o         []

```

%test:/tag1>select * at tag2 where (attr2=i or attr2=p) and attr1=o

```

      *tun      attr4      attr2      attr3      attr1      *text
/tag1/tag2
      I7       None       p         None       o         []

```

%test:/tag1>select * at tag2 where ((attr2=i or attr2=p) and attr1=o) and attr4=None

```

      *tun      attr4      attr2      attr3      attr1      *text
/tag1/tag2
      I7       None       p         None o         []

```

As you can see there is no limit to parenthesis imbrication.

- Selecting using complex where close with other operators than "="

e.g. 1:

%test:/tag1>select * at tag2 where attr2 > i # Using > operator

```

      *tun      attr4      attr2      attr3      attr1      *text
/tag1/tag2
      I7       None       p         None       o         []

```

e.g. 2:

%test:/tag1>select * at tag2 where attr2 >= i # Using >= operator

```

      *tun      attr4      attr2      attr3      attr1      *text
/tag1/tag2

```

```

      I4      k      i      j      h      []
      *tun    attr4   attr2   attr3   attr1   *text
/tag1/tag2
      I7      None    p      None    o      []

```

e.g.:

%test:/tag1>select * at tag2 where attr2 <> i # Using <> operator

```

      *tun    attr4   attr2   attr3   attr1
/tag1/tag2
      I2      None    d      e      c
      *tun    attr4   attr2   attr3   attr1
/tag1/tag2
      I7      None    p      None    o

```

e.g.:

%test:/tag1>select * at tag2 where attr2 *in [i,p] # Using *in operator

```

      *tun attr4      attr2      attr3      attr1      *text
/tag1/tag2
      I4  k      i      j      h      []
      *tun attr4      attr2      attr3      attr1      *text
/tag1/tag2
      I7  None    p      None    o      []

```

e.g.:

%test:/tag1>select * at tag2 where attr2 *between [i,p] # Using *between operator

```

      *tun    attr4   attr2   attr3   attr1   *text
/tag1/tag2
      I4      k      i      j      h      []
      *tun    attr4   attr2   attr3   attr1   *text
/tag1/tag2
      I7      None    p      None    o      []

```

- Selecting a guiven count of nodes

e.g.:

%test:/tag1>select * at tag2

	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>
<i>/tag1/tag2</i>	<i>I2</i>	<i>None</i>	<i>d</i>	<i>e</i>	<i>c</i>
	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>
<i>/tag1/tag2</i>	<i>I4</i>	<i>k</i>	<i>i</i>	<i>j</i>	<i>h</i>
	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>
<i>/tag1/tag2</i>	<i>I7</i>	<i>None</i>	<i>p</i>	<i>None</i>	<i>o</i>

e.g.:

%test:/tag1>select * at tag2[1]

	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>
<i>/tag1/tag2</i>	<i>I2</i>	<i>None</i>	<i>d</i>	<i>e</i>	<i>c</i>

e.g.:

%test:/tag1>select * at tag2[2]

	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>
<i>/tag1/tag2</i>	<i>I2</i>	<i>None</i>	<i>d</i>	<i>e</i>	<i>c</i>
	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>
<i>/tag1/tag2</i>	<i>I4</i>	<i>k</i>	<i>i</i>	<i>j</i>	<i>h</i>

6.3 CREATE OPERATION

The create operation, creates one or more nodes to one or more guiven targets.

Syntax:

create O_WHAT O_SET at F_TAGS where F_ATTRS

- Simple create

e.g.:

:>xpc -F E:\Projets\REPOSITORY\repoz\samples\test.xml -X

New processor with alias:test, created and mounted.

:test:/tag1>%

%test:/tag1>select * at tag2

	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>
<i>/tag1/tag2</i>	<i>I2</i>	<i>None</i>	<i>d</i>	<i>e</i>	<i>c</i>
<i>/tag1/tag2</i>	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>
<i>/tag1/tag2</i>	<i>I4</i>	<i>k</i>	<i>i</i>	<i>j</i>	<i>h</i>
<i>/tag1/tag2</i>	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>
<i>/tag1/tag2</i>	<i>I7</i>	<i>None</i>	<i>p</i>	<i>None</i>	<i>o</i>

%test:/tag1>create tag2 set attr1=a;attr2=b;attr3=c at tag1

/tag1

*Creating Node:tag2 *tun:I8, on Node:/tag1.*

%test:/tag1>select * at tag2

	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>
<i>/tag1/tag2</i>	<i>I2</i>	<i>None</i>	<i>d</i>	<i>e</i>	<i>c</i>
<i>/tag1/tag2</i>	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>
<i>/tag1/tag2</i>	<i>I4</i>	<i>k</i>	<i>i</i>	<i>j</i>	<i>h</i>

	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>ztr1</i>
<i>/tag1/tag2</i>	<i>I7</i>	<i>None</i>	<i>p</i>	<i>None</i>	<i>o</i>
	<i>*tun</i>	<i>attr4</i>	<i>attr2</i>	<i>attr3</i>	<i>attr1</i>
<i>/tag1/tag2</i>	<i>I8</i>	<i>None</i>	<i>b</i>	<i>c</i>	<i>a <---</i>

- Creating using a where clause

e.g.:

```
%test:/tag1>create tag4 at tag2 where attr2=p
```

```
/tag1/tag2
```

```
Creating Node:tag4 *tun:I17, on Node:/tag1/tag2.
```

```
%test:/tag1>xselect * at tag1
```

```
/tag1
```

```
*tun:I1
```

```
<tag1 attr2="b" attr1="a">
  <tag2 attr2="d" attr3="e" attr1="c">
    </tag2>
    <tag3 attr3="A value with spaces !" attr1="f" attr2="g">
      aaaa1
      bbbbbbbb2
    </tag3>
    <tag2 attr4="k" attr3="j" attr1="h" attr2="i">
      <tag4 >
        <tag5 attr2="m" attr3="n" attr1="True">
          </tag5>
        </tag4>
      </tag2>
    <tag2 attr4="None" attr3="None" attr1="o" attr2="p">
      <tag4 >
        </tag4>
      </tag2>
    </tag1>
```

- Creating using a Hierarchical where clause

e.g.:

```
%test:/tag1>create tag5 set attr1=False;attr2=aaa;attr3=bbb at tag4 where tag2@attr2=p
```

```
/tag1/tag2/tag4
```

```
Creating Node:tag5 *tun:I19, on Node:/tag1/tag2/tag4.
```

%test:/tag1>xselect * at tag1

/tag1

*tun:11

```

<tag1 attr2="b" attr1="a">
  <tag2 attr2="d" attr3="e" attr1="c">
    </tag2>
    <tag3 attr3="A value with spaces !" attr1="f" attr2="g">
      aaaa1
      bbbbbbbb2
    </tag3>
    <tag2 attr4="k" attr3="j" attr1="h" attr2="i">
      <tag4 >
        <tag5 attr2="m" attr3="n" attr1="True">
          </tag5>
        </tag4>
      </tag2>
    <tag2 attr4="None" attr3="None" attr1="o" attr2="p">
      <tag4 >
        <tag5 attr2="aaa" attr3="bbb" attr1="False">
          </tag5>
        </tag4>
      </tag2>
    </tag1>

```


6.4 UPDATE OPERATION

The Update operation, updates one or more node with a set of pair of Attribute/Values.

Syntax:

update F_TAGS where F_ATTRS O_SET

- Updating multiple nodes

e.g.:

```
%test:/tag1>update tag2 set attr1=bla bla
```

```
/tag1/tag2
```

```
Updating *tun:I2
```

```
/tag1/tag2
```

```
Updating *tun:I4
```

```
/tag1/tag2
```

```
Updating *tun:I7
```

```
%test:/tag1>xselect * at tag1
```

```
/tag1
```

```
*tun:I1
```

```
<tag1 attr2="b" attr1="a">
  <tag2 attr2="d" attr3="e" attr1="bla bla">
  </tag2>
  <tag3 attr3="A value with spaces !" attr1="f" attr2="g">
  aaaa1
  bbbbbbbb2
  </tag3>
  <tag2 attr4="k" attr3="j" attr1="bla bla" attr2="i">
  <tag4 >
  <tag5 attr2="m" attr3="New value" attr1="True">
  </tag5>
  </tag4>
  </tag2>
  <tag2 attr4="None" attr3="None" attr1="bla bla" attr2="p">
  </tag2>
</tag1>
```

- Updating using a Hierarchical where clause

e.g.:

```
%test:/tag1>update tag5 where tag2@attr2=i set attr3=New value
/tag1/tag2/tag4/tag5
Updating *tun:16
```

```
%test:/tag1>xselect * at tag1
```

```
/tag1
```

```
*tun:11
```

```
<tag1 attr2="b" attr1="a">
  <tag2 attr2="d" attr3="e" attr1="c">
    </tag2>
  <tag3 attr3="A value with spaces !" attr1="f" attr2="g">
    aaaa1
    bbbbbbbb2
  </tag3>
  <tag2 attr4="k" attr3="j" attr1="h" attr2="i">
    <tag4 >
      <tag5 attr2="m" attr3="New value" attr1="True">
        </tag5>
      </tag4>
    </tag2>
  <tag2 attr4="None" attr3="None" attr1="o" attr2="p">
    </tag2>
  </tag1>
```

6.5 DELETE OPERATION

The delete operation, deletes one or more node regarding an optional where clause.

Syntax:

delete F_TAGS where F_ATTRS

- Deleting multiple nodes

%test:/tag1>delete tag2,tag3

/tag1/tag2

*Deleting Node:tag2 *tun:I2, on Node:/tag1 *tun:I1.*

/tag1/tag3

*Deleting Node:tag3 *tun:I3, on Node:/tag1 *tun:I1.*

/tag1/tag2

*Deleting Node:tag2 *tun:I4, on Node:/tag1 *tun:I1.*

/tag1/tag2

*Deleting Node:tag2 *tun:I7, on Node:/tag1 *tun:I1.*

%test:/tag1>xselect * at tag1

/tag1

**tun:I1*

<tag1 attr2="b" attr1="a">

</tag1>

- Deleting using a Hierarchical where clause

e.g.:

%test:/tag1>delete tag5 where tag2@attr2=i

/tag1/tag2/tag4/tag5

*Deleting Node:tag5 *tun:I6, on Node:/tag1/tag2/tag4 *tun:I5.*

%test:/tag1>xselect * at tag1

/tag1

**tun:I1*

```
<tag1 attr2="b" attr1="a">  
  <tag2 attr2="d" attr3="e" attr1="c">  
    </tag2>  
    <tag3 attr3="A value with spaces !" attr1="f" attr2="g">  
      aaaa1  
      bbbbbbbb2  
    </tag3>  
  <tag2 attr4="k" attr3="j" attr1="h" attr2="i">  
    <tag4 >  
    </tag4>  
  </tag2>  
<tag2 attr4="None" attr3="None" attr1="o" attr2="p">  
</tag2>  
</tag1>
```

6.6 DUPLICATE OPERATION

The Duplicate operation, duplicates one node to one or more target.

Syntax:

duplicate F_TAGS where F_ATTRS at F_TAGS where F_ATTRS

- Duplicating to more than one node

e.g.:

:>xpc -F E:\Projets\REPOSITORY\repoz\samples\test.xml -X

New processor with alias:test, created and mounted.

:test:/tag1>%

%test:/tag1>xselect * at tag1

/tag1

**tun:11*

```

<tag1 attr2="b" attr1="a">
  <tag2 attr2="d" attr3="e" attr1="c">
    </tag2>
  <tag3 attr3="A value with spaces !" attr1="f" attr2="g">
    aaaa1
    bbbbbbbb2
  </tag3>
  <tag2 attr4="k" attr3="j" attr1="h" attr2="i">
    <tag4 >
      <tag5 attr2="m" attr3="n" attr1="True">
        </tag5>
      </tag4>
    </tag2>
  <tag2 attr4="None" attr3="None" attr1="o" attr2="p">
    </tag2>
  </tag1>

```

%test:/tag1>duplicate tag4 at tag2

/tag1/tag2

*Duplicating Node:tag4 *tun:18, on Node:/tag1/tag2.*

/tag1/tag2

*Duplicating Node:tag4 *tun:110, on Node:/tag1/tag2.*

/tag1/tag2

Duplicating Node:tag4 *tun:112, on Node:/tag1/tag2.

```
%test:/tag1>xselect * at tag1
```

```
/tag1
```

```
*tun:11
```

```
<tag1 attr2="b" attr1="a">
  <tag2 attr3="e" attr1="c" attr2="d">
    <tag4 > <--- Duplicated
      <tag5 attr2="m" attr3="n" attr1="True">
        </tag5>
      </tag4>
    </tag2>
  <tag3 attr3="A value with spaces !" attr1="f" attr2="g">
    aaaa1
    bbbbbbbb2
  </tag3>
  <tag2 attr4="k" attr3="j" attr1="h" attr2="i">
    <tag4 > (original)
      <tag5 attr2="m" attr3="n" attr1="True">
        </tag5>
      </tag4>
    <tag4 > <--- Duplicated
      <tag5 attr2="m" attr3="n" attr1="True">
        </tag5>
      </tag4>
    </tag2>
  <tag2 attr4="None" attr3="None" attr1="o" attr2="p">
    <tag4 > <--- Duplicated
      <tag5 attr2="m" attr3="n" attr1="True">
        </tag5>
      </tag4>
    </tag2>
  </tag1>
```

- Duplicating using a Hierarchical where clause for the source node

e.g.:

```
%test:/tag1>duplicate tag4 where tag2@attr2=i at tag2
```

```
/tag1/tag2
```

Duplicating Node:tag4 *tun:18, on Node:/tag1/tag2.

```
/tag1/tag2
```

Duplicating Node:tag4 *tun:110, on Node:/tag1/tag2.

/tag1/tag2

Duplicating Node:tag4 *tun:112, on Node:/tag1/tag2.

This gives the same result as previous sample.

- Duplicating using a Hierarchical where clause for the source node and the target node

e.g.:

>xpc -F E:\Projets\REPOSITORY\repoz\samples\test.xml -X

New processor with alias:test, created and mounted.

:test:/tag1>%

*%test:/tag1>xselect * at tag1*

/tag1

*tun:11

```
<tag1 attr2="b" attr1="a">
  <tag2 attr2="d" attr3="e" attr1="c">
  </tag2>
  <tag3 attr3="A value with spaces !" attr1="f" attr2="g">
    aaaa1
    bbbbbbbb2
  </tag3>
  <tag2 attr4="k" attr3="j" attr1="h" attr2="i">
    <tag4 >
      <tag5 attr2="m" attr3="n" attr1="True">
      </tag5>
    </tag4>
  </tag2>
  <tag2 attr4="None" attr3="None" attr1="o" attr2="p">
  </tag2>
</tag1>
```

%test:/tag1>duplicate tag4 where tag2@attr2=i at tag2 where tag2@attr2=p

/tag1/tag2

Duplicating Node:tag4 *tun:18, on Node:/tag1/tag2.

%test:/tag1>xselect * at tag1

/tag1

*tun:11

```
<tag1 attr2="b" attr1="a">
  <tag2 attr2="d" attr3="e" attr1="c">
  </tag2>
  <tag3 attr3="A value with spaces !" attr1="f" attr2="g">
```

```
aaaa1  
bbbbbbbb2  
</tag3>  
<tag2 attr4="k" attr3="j" attr1="h" attr2="i">  
  <tag4 >  
    <tag5 attr2="m" attr3="n" attr1="True">  
      </tag5>  
    </tag4>  
  </tag2>  
<tag2 attr4="None" attr3="None" attr1="o" attr2="p">  
  <tag4 >  
    <tag5 attr2="m" attr3="n" attr1="True">  
      </tag5>  
    </tag4>  
  </tag2>  
</tag1>
```


6.7 INSERT OPERATION

The insert operation is a Repoz special operation that uses the Repoz Return cache capability, to allow insertion of the stored node(s) into another file or another part of the same xml file.

Syntax:

insert \$ro where F_ATTRS at F_TAGS where F_ATTRS

Here the original content of the file test.xml, we also open the **first processor**.

```
>xpc -F E:\Projets\REPOSITORY\repoz\samples\test.xml -X
```

New processor with alias:test, created and mounted.

```
:test:/tag1>%
```

```
%test:/tag1>xselect * at tag1
```

```
/tag1
```

```
*tun:11
```

```
<tag1 attr2="b" attr1="a">
  <tag2 attr2="d" attr3="e" attr1="c">
    </tag2>
  <tag3 attr3="A value with spaces !" attr1="f" attr2="g">
    aaaa1
    bbbbbbbb2
  </tag3>
  <tag2 attr4="k" attr3="j" attr1="h" attr2="i">
    <tag4 >
      <tag5 attr2="m" attr3="n" attr1="True">
        </tag5>
      </tag4>
    </tag2>
  <tag2 attr4="None" attr3="None" attr1="o" attr2="p">
    </tag2>
  </tag1>
```

We run a select clause.

In the Repoz scheme, a select always returns the set of found nodes into the Repoz Variable : ro.

%test:/tag1>select * at tag2

```

      *tun      attr2      attr3      attr1      *text
/tag1/tag2    I2          d          e          c          []

      *tun      attr4      attr2      attr3      attr1      *text
/tag1/tag2    I4          k          i          j          h          []

      *tun      attr4      attr2      attr3      attr1      *text
/tag1/tag2    I7          None       p          None       o          []

```

This is the content of the Repoz Variable ro (a list of nodes):

%test:/tag1>var ro

```
[<lib.epicxmpl.Node instance at 0x00BBF620>, <lib.epicxmpl.Node instance at 0x00BBF738>,
<lib.epicxmpl.Node instance at 0x00BBF670>]
```

Now we may create and **mount a second processor** where to make the insertion.

Note: It could be the same processor.

%test:/tag1>xpc -F E:\Projets\REPOSITORY\repoz\samples\test.xml -X -s

New processor with alias:test0, created. (use mount test0, to mount it)

%test:/tag1>mount test0

Now we run the insert command using the Repoz: ro Variable

%test0:/tag1>insert \$ro at tag1

```

Found target node at:/tag1
Adding child:/tag1/tag2
Adding child:/tag1/tag2
Adding child:/tag1/tag2

```

This is the result:

%test0:/tag1>xselect * at tag1

```
/tag1
```

*tun:l1

```

<tag1 attr2="b" attr1="a">
  <tag2 attr2="d" attr3="e" attr1="c">
    </tag2>
  <tag3 attr3="A value with spaces !" attr1="f" attr2="g">
    aaaa1
    bbbbbbbb2
  </tag3>
  <tag2 attr4="k" attr3="j" attr1="h" attr2="i">
    <tag4 >
      <tag5 attr2="m" attr3="n" attr1="True">
        </tag5>
      </tag4>
    </tag2>
  <tag2 attr4="None" attr3="None" attr1="o" attr2="p">
    </tag2>
  <tag2 attr2="d" attr3="e" attr1="c"> <-- New from insert
    </tag2>
  <tag2 attr4="k" attr3="j" attr1="h" attr2="i"> <-- New from insert
    <tag4 >
      <tag5 attr2="m" attr3="n" attr1="True">
        </tag5>
      </tag4>
    </tag2>
  <tag2 attr4="None" attr3="None" attr1="o" attr2="p"> <-- New from insert
    </tag2>
</tag1>

```