XMLAccess Frequently Asked Questions.

What is XMLAccess?

The Digital Experience XML Configuration Interface, also known as XMLAccess, is a command line tool or utility that takes an XML file as input. The XML file contains statements or instructions to modify a Digital Experience configuration.

For example, changing the access controls on several DX Portal pages can be a time consuming task using the DX Portal administrative UI. The changes can be made quickly using XMLAccess. Similarly, you can deploy new or updated Portlet applications using XMLAccess. Virtually any task that can be accomplished using the UI can be performed via XMLAccess.

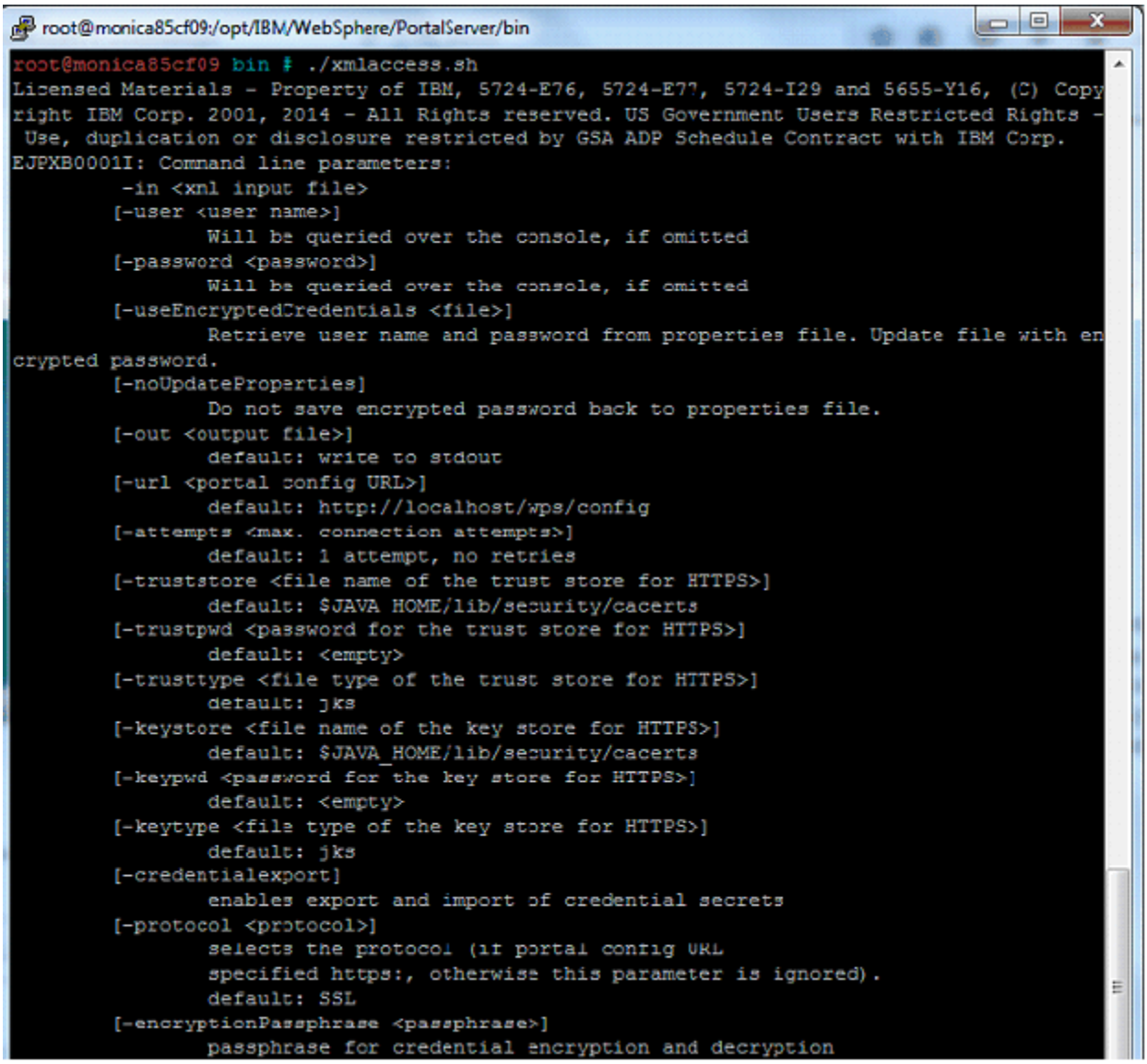
The XMLAccess client starts a Java process that reads an XML input file and prepares to write an output XML file. The script sends the commands in the input XML file, through network, to a special servlet installed in the DX Portal called the "config" servlet. It's running in the Portal continuously and listening to and waiting for client to send requests. For this reason, the DX Portal server must be running to run XMLAccess commands. That is, XMLAccess cannot be run against a stopped DX Portal server.

Once the commands in the XML file has been received by the servlet, the client XMLAccess Java process will wait for the results coming back, report status in the command or terminal window, and write the output XML file. The XML input file is processed by the server and, in many cases, the XML is "converted" into a series of database insert, update or delete statements depending on the directives in the input XML. If a web or portlet application is installed or updated, the XMLAccess code will call the appropriate WebSphere Application Server API to deploy or update the application.

Questions:

HOW DO I INVOKE OR RUN XMLACCESS?

XMLAccess command is located in directory <PortalServer_root>/bin.
If you run XMLAccess without any command line parameters, a list of available parameters will be shown, as follows:



Here is the short description of these command line arguments:

-**in** this is the only required parameter to specify the input XML file name;
-user this is the user id to run this command.
-password (or **-pwd**) this is the password of the user running the command.
-useEncryptedCredentials Retrieve user name and password (encrypted or unencrypted) from the properties file.
-noUpdateProperties Do not save encrypted password back to properties file.
-out specify the output XML file. If omitted, write output to the command window or terminal.
-url specify the Portal “config” URL. It's defaulted to http://localhost/wps/config

-attempts specify how many times the command will be tried. The default is 1, no retries.
-truststore When HTTPS is used in -url parameter, this specify the trust certificate store.
If omitted, $JAVA_HOME/lib/security/cacerts will be assumed.
-trustpwd the password to open the truststore file.
-trusttype specify the type of the truststore file. If omitted, JKS is assumed.
-keystore specify the keystore file when client certificate is required for client authentication on server.
If omitted, $JAVA_HOME/lib/security/cacerts is assumed
-keypwd The password to open the keystore files
-keytype The keystore file type. If omitted, JKS is assumed.
-credentialexport This parameter enables export and import of credential secrets from or to the credential vaults.
-protocol selects the protocol (if the Portal config URL specified https) otherwise this parameter is ignored.
-encryptionPassphrase The passphrase used to encrypt and decrypt the password to be imported or exported.

DOES XMLACCESS HAVE TO BE RUN DIRECTLY ON THE PORTAL SERVER HOST?

The answer is no. It is perfectly fine to run XMLAccess from one system against a remote DX Portal server. XMLAccess can be run from any host as long as a Java Runtime Environment (JRE) is installed and the DX Portal server can be contacted over the network. HCL Digital Experience does not have to be installed to run XMLAccess. As a matter of fact, it is recommended that XMLAccess not be run on a production server, because the JAVA process which invokes XMLAccess requires memory to run, in some cases, it would use a lot, especially with a lot of updates.

To run XMLAccess on a remote server, do the following:

1) Copy the files listed below from any Portal system to a remote system with a JRE:

a) <PortalServer_root>/base/wp.xml.client/bin/wp.xml.client.jar
b) <PortalServer_root>/base/wp.base/shared/app/wp.base.jar
c) <AppServer_root>/lib/j2ee.jar
d) <wp_profile_root>/bin/xmlaccess.sh or <wp_profile_root>\bin\xmlaccess.bat

2) Edit the xmlaccess batch or shell script file and adjust the paths to the jar files as appropriate.

All of the files can be copied to the same directory and the directory can have any name. If your DX Portal is upgraded it is a good idea to copy the updated versions of the files to the remote system.

HOW ARE THE OBJECTIDS IN XML EXPORT FILE CONSTRUCTED?

When a DX Portal artifact or resource is added to the database, either during the initial installation or later through development, a unique object identifier (ID) is then assigned to the resource. The object ID is generated by an internal algorithm and it is guaranteed to be unique for all portal installations. The main purpose of an object ID is to identify the object and provide an association when it is referenced elsewhere in the XML file generated by the export operation. When the artifact resides in the portal database, the object ID is fixed and will not change, until it is deleted. The normal way to obtain the objectID is to export the portal database content using XMLAccess. There are exceptions that some object IDs are intentionally kept the same for different installations. "wps.content.root" for the base portal is one such example. Some artifacts do not have object IDs, because they are not to be referenced or their "singleton" nature. For example, "portal", global settings, service settings, and policy-nodes.

The object IDs are categorized. The category of an object ID is identified by the first 1 or 2 characters.

Some commonly used ones are:
0 - VIRTUAL RESOURCE

1 - WEB_MODULE
2 - PORTLET_APPLICATION_DEFINITION
3 - PORTLET_DEFINITION
5 - PORTLET_ENTITY (PORTLETINSTANCE)
6 - CONTENT_NODE
7 - COMPONENT
8 - USER GROUP
9 - USER
10 - Cross-page-wire
13 - ACTION
15 - transformationinstance
16 - transformation
17 - transformation-app
18 - Virtual Portal
1I –Language

C - URL MAPPING
E - CREDENTIAL VAULT SEGMENT
F - CREDENTIAL SLOT
J - THEME
K - SKIN
L - MARKUP
M - CLIENT
R - EVENT-HANDLER
V –SERVLET_DEFINITION
CI –tag
CI - rating

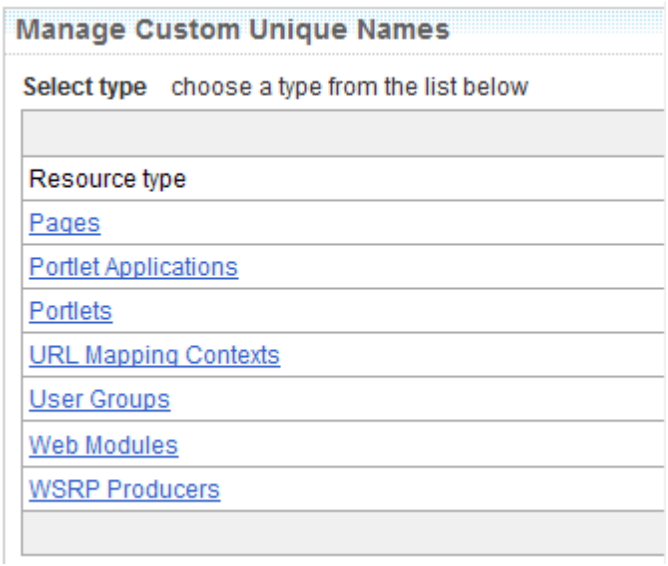
In DX Portal, a Z is added to the category identifier. For example, Z6 identifies a content node.

Since the object IDs are not meant to be manually handled by human, it is strongly recommended to create unique names for any artifacts to be referenced in your own XML script. Portal will guarantee no two resources will have the same unique name.

WHAT IS A UNIQUE NAME? HOW IS IT USED?

Unique names are identifiers given to DX Portal resources (pages, portlets, themes, skins, etc) by DX Portal administrators to uniquely identify them during deployment and administration, especially when XMLAccess is used. These names are alphanumerical strings, plus . (period), - (hyphen), and _ (underscore). Any other special characters are not recommended.

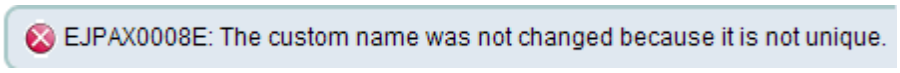
In HCL Digital Experience Graphic User Interface (GUI), you can set unique names to resources using Manage Custom Unique Names portlet. You can, however, only set unique names to a few categories, as shown below,



If you want to set unique names to other types of resources not in the list above, you can use XMLAccess.

When creating a page, if you try to assign a unique name to the page, but the unique name is already assigned to an existing resource (not necessarily a page), no errors would be generated, and the page will be created, but the unique name would not be shown for the page. This is true, even when the existing resource is inactive. Thus when you are having trouble assigning a unique name to a resource this way, chances are that the unique name has been used on another resource. To find out whether that is the case, you should obtain a full export using XMLAccess and then search the unique name to see if it is already in use.

On the other hand, if you try to assign a unique name that has already been used to a new resource in Manage Custom Unique Names portlet, an error message like the following would be returned:



Or if you tried to do similar assignment using XMLAccess, the following will be shown in the result XML file,

```
<status element="[content-node Z6_4GFA8VO20OJ9F0IE5UOP6K3083 uniquename=sample.test.page.1]" result="failed">
        <message id="EJPXA0187E">com.ibm.wps.command.xml.XmlCommandException:
                EJPXA0187E: The resource content-node already exists.
                [content-node Z6_4GFA8VO20OJ9F0IE5UOP6K3083 uniquename=sample.test.page.1]
        </message>
</status>
```

You cannot delete a unique name if you have not deleted the corresponding resource, but you can change the unique name of the resource, and then use the original unique name for something else. Be careful when you change unique name of a resource if you have used it as a way of updating resources between two different systems. If you change the unique name of, say, a page, you basically cut off the association between the two systems.

As a best practice, we recommend a convention of creating unique names to be adopted in your own company or organization, such that they are easily identifiable for different types of resources. Please do not use names that could be confused with the out-of-the-box DX Portal resources.

Unique names are very useful for release updates, especially for systems that do not share the same objectIDs, and can be the reference of changes when Release Builder is used.

WHEN YOU OMIT USER AND PASSWORD ON COMMAND LINE, WILL THE COMMAND FAIL?

No, In this case, you would be prompted for user ID and password. As a matter of fact, this is preferred when you do not want someone who is watching to read the password when you type the password in the command or terminal window. For example, if you are sharing a web conference session with other people to show them how to run the command. Omitting the username and password from the command line will also prevent the password from showing in cleartext in the operating system's history of commands

DOES OMITTING -URL PARAMETER ON COMMAND LINE CAUSE XMLACCESS COMMAND TO FAIL?

Yes in most cases, but not necessarily. If the -url parameter is omitted, the URL will be default to http://localhost/wps/config. This is not a correct URL in most cases, because it assumes localhost is the local HTTP server's host name, or DX Portal is configured to use port 80. Most HTTP servers or DX Portal are not set this way. Thus most likely this URL is not the one you should run XMLAccess with. Therefore, this URL would generate errors like the following, if your DX Portal is not configured with port 80 or not configured to the local HTTP server using "localhost" as the server name,

EJPXB0006I: Connecting to URL http://localhost/wps/config
EJPXB0009E: Could not connect to portal.
EJPXB0016E: An error occurred on the client: Connection refused: connect

Because of this reason, the -url parameter is pretty much required. If omitted, the XMLAccess command has a higher chance of failing with the error above.

CAN XMLACCESS BE USED IN A SHELL SCRIPT OR BATCH FILE?

Yes. XMLAccess can be called in the same way any external program would.

There are a couple of precautions:

- When multiple XML files are pipelined to run in a single shell script file, they should not be dependent on the result of others. Otherwise, there could be irreversible effects. The reason for this is that XMLAccess is not an API, and there is not a programming way to check the status/return code. Sometimes, even the result is "OK", DX Portal may need to be restarted.
- Though you can run multiple shell scripts simultaneously, it's not supported to run multiple XMLAccess commands at the same time.
- Before running scripts, it's always recommended to back up DX Portal file system, and the databases. Especially if the XML files update DX Portal databases and file systems. The scripts should be fully tested before being made into a workflow.

At this time, HCL Support does not have any sample scripts to provide.

CAN I RUN XMLACCESS THROUGH WEBSEAL JUNCTION?

The answer is no. The reason is that XMLAccess is a command line tool. It can't fulfill WebSEAL's basic authentication challenge, thus the admin user cannot be authenticated by WebSEAL. Since XMLAccess is an administrative function in nature, it should not be an exposure. If there is a concern about sending the password through the open wire, it is recommended to use SSL over HTTP when running XMLAccess.

CAN XMLACCESS BE USED TO RENAME A SERVLET?

The answer is no. Renaming servlets is not supported by XMLAccess. When updating a servlet, the correlation between the existing servlet and the servlet in the update XML is done by name.
In the case of an existing servlet, if the name is changed, and XMLAccess does not find a servlet under the new name, it assumes that this is a new servlet and creates it.

CAN I RUN A CONFIGENGINE OR WPSCONFIG TASK TO EXECUTE XMLACCESS COMMANDS?

The answer is yes.
Using the following XML script file, you should be able to run XMLAccess to export releases:

```
<?xml version="1.0" encoding="UTF-8"?>
	<target name="full-export-release">
		<xmlaccess user="${PortalAdminIdShort}" password="${PortalAdminPwd}"
		srcfile="${WpsInstallLocation}/doc/xml-samples/ExportRelease.xml"
		destfile="${WpsInstallLocation}/bin/fullreleasexport.xml"/>
	</target>
```

where we assume “PortalAdminIdShort” and “PortalAdminPwd” are configured in wkplc.properties
Alternatively, they can be provided on command line as well.
Save these into an XML file and move the file into directory <wp_profile>/ConfigEngine/config/includes
Then you should be able to run the following,

./ConfigEngine.sh full-export-release

and the output file "fullreleaseexport.xml" can be found in <PortalServer/bin/.

Similarly, you can have other XMLAccess commands in the same XML file with different input and output files.

CAN I USE WILDCARD IN UNIQUENAME SEARCH?

When exporting all pages, you may wonder whether you can use something like

<content-node action="export" uniquename="*"/>

The answer is no. For the normal export of Portal resources, the only attribute that takes wild card is "objectid". Thus, to export all content-nodes, using the same example above, the tag should be like

<content-node action="export" objectid="*"/>{

In some special cases, like exporting all policies, you can use the following

<policy-node action="export" path="*"/>

to export all different types of policies.

CAN YOU ASSIGN A UNIQUENAME TO ANY RESOURCE?

Most of DX Portal resources can be assigned a unique name, such that you can reference them by their unique names, when you use administration tools to manage them. The resources that can be assigned a unique name by using DX Portal Administration portlet, Custom Unique Names portlet, are Pages, Portlet Applications, Portlets, URL Mapping Contexts, User Groups, Web Modules, and WSRP Producers.

Many other resources can be assigned unique names through XMLAccess, such as themes and skins, components, credential vault segments and slots, cross-page wires, portlet instances, event handlers, clients, etc.

Some attributes do not allow a unique name by nature, such as policy node, access control items, application roles, markups, locale data, etc.

WHEN THE UNIQUE NAME “WPS.CONTENT.ROOT” IS CHANGED, WOULD IT AFFECT DX?

When you change the unique name "wps.content.root" of Content Root, your DX Portal may not be accessible any more. This is because that the DX Portal rendering content model depends on the correct reference of this uniquename. So it is not supported to change the unique name of the Content Root. If you or someone accidentally changed it, however, you can use XMLAccess to change it back, because XMLAccess does not depend on the content model.

WHEN I DEPLOY APPLICATIONS IN A CLUSTER, DO I NEED TO RUN THE TASK ON EVERY NODE?

The answer is No. You only need to run the XMLAccess command on one node, preferably the primary node, followed by a full synchronization, and then run "activate-portlets" to activate the portlets deployed.

CAN I USE "DELETE" IN THE ACTION ON "PORTAL"?

Is it safe to use XMLAccess with DX Portal? Is it possible for one to accidentally delete the entire DX Portal using "delete" on "portal"?

For example, can I run the following script to delete "portal"?

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="PortalConfig_7.0.0.0.xsd" type="update" domain="rel">
<portal action="delete" />
</request>
```

The answer is No.
You will see similar error messages as follows, when you run XMLAccess with the script above as input:

```
<status element="unknown" result="failed">
<message id="EJPXA0049E">com.ibm.wps.command.xml.XmlFormatException: EJPXA0049E: Input syntax error in line 4 column 28: the XML input does not conform to the XML schema PortalConfig.xsd. unknown</message>
<message>org.xml.sax.SAXParseException: cvc-enumeration-valid: Value 'delete' is not facet-valid with respect to enumeration '[locate, export]'. It must be a value from the enumeration.</message>
</status>
```

Apparently, "portal" should not take action "delete" and this is guaranteed by the XMLAccess schemas.

HOW WOULD I CHANGE THE UNIQUENAME OF A PAGE USING XMLACCESS

Remember, objectIDs represent the uniqueness of an object in the DX Portal database. Therefore, we may reference the objectID of a resource with an "update" action to update the unique name.

For example, using a script like the following:

<content-node objectid="6_CGAH47L0084G00I4BONHCQ10A3" uniquename="com.acme.sample.page"/>

you can set an existing page with that object ID to the new uniquename.

On the other hand, you cannot use uniquename to set objectID to a different value.

WHEN CREATING A CONTENT-NODE, CAN I SPECIFY THE POSITION IN A HIERARCHY?

The answer is yes. Portal pages are arranged as a tree hierarchy with Content Root as the top. Each node, called content node, in the tree can be a page, a label, or a URL (internal or external). The position of a content node under a parent node in the hierarchy can be ranked by its ordinal, an integer. The smaller the ordinal, the higher the rank. The display and rendering of the content tree in DX Portal is according to the page ordinals.

When you add a node, you can specify its ordinal which can take the following values
•**first**: this puts the content-node at the top of the hierarchy;
•**last**: this puts the content-node at the bottom of the hierarchy;
•**an integer**: this puts the content-node by comparing the ordinals of those currently in the system and inserting it in the proper position;
•**a hash mark "#" followed by an integer:** this puts the page to the absolute location by counting the top one as 0, the second 1, and so on. Thus, "#12" will put the new content node to the 13th position in the current hierarchy.

WHAT ACCESS RIGHTS ARE REQUIRED TO RUN XMLACCESS?

We normally run XMLAccess using the DX Portal administrative user or a user in the administrative group. If you want to assign any other users to run XMLAccess, they need Security Administrator@Portal + Editor@XML Access where Portal and XML Access are two virtual resources. This assumes that there are no role blocks set for inheritance in Portal resource hierarchy. If you have configured any inheritance role block, it is advised to give proper access to the user or group who runs XMLAccess.

What problem could it cause if the user's access rights are not correct? When a user does not have the proper rights to run XMLAccess, you would find the following in the output:

<failure>
com.ibm.wps.command.xml.XmlCommandServlet$AuthenticationException:
EJPXA0011E: Authentication for user **wpsadmin** failed.
</failure>

To assign the proper access rights to users or groups, you can Resource Permissions portlet, or User and Group Permissions portlet. Alternatively, you can also use XMLAccess to assign permissions.

WHAT IS "XMLACCESS SCRIPTING USER" AND HOW IS IT USED?

When the DX Portal is installed, an internal user called "xmlaccess scripting user" is added to the DX Portal database. This user is mapped to the user who runs the XMLAccess command after that user is authenticated and access control is checked.

In the trace log, you should see the entries like the following,

authority: Name: xmlaccess scripting user,

OID:[ExtIDImpl '9eAeOBROMGDC3BP6NP14JRO4NK1EKBQSMJ14LRSA6P1' [xmlaccess scripting user / USER, Domain: [Domain: rel]]]

Without such "authority", the XMLAccess would fail.

Due to its importance, please do not remove or modify this user from DX Portal database.

WHEN XMLACCESS RUNS OUT OF MEMORY, WHAT CAN I DO?

Sometimes, when importing an XML file with a lot of updates, you may see OutOfMemoryError. You may see this problem again if you rerun the same command. This error can occur on either the client or server side, but more often on the server.

If the problem occurs on the client side - The first suggestion is to not run XMLAccess on the same DX Portal server. When you run it on a remote client, it would off load the job from the running DX Portal server.

The second suggestion is to increase the client JVM by doing the following,
- open xmlaccess.bat or xmlaccess.sh under PortalServer/bin, depending on the platform;
- search for the JAVA line, you should see something like the following,

%JAVA% -classpath
%WPS_HOME%\base\wp.xml.client\bin\wp.xml.client.jar;%WPS_HOME%\base\wp.base\shared\app
\wp.base.jar;%WAS_HOME%\lib\j2ee.jar;%cpath% com.ibm.wps.xmlaccess.XmlAccess %*

- adding parameters "-Xms128m -Xmx512m" would give the initial size 128MB and maximum 256MB. You can adjust these parameters to suit your client. Make sure your client machine has that much real memory. If not, try to stop other applications.

If the OutOfMemoryError occurs on the server side, you may want to avoid running XMLAccess when the server is busy, such as at high peak hours during the day. If that doesn't resolve the problem, try to increase the JVM heapsize on WebSphere_Portal server. Also try to make multiple small updates in multiple XML files, rather than making a lot of updates in a single big XML file.

DOES XMLACCESS SUPPORT TRANSACTIONS?

When you use XML scripts to create, update or delete resources, the changes in the DX portal database are grouped into transactions. All changes that are part of one transaction are either executed completely or not at all.

The XML configuration has two different levels of grouping database updates into transactions. The grouping is defined by the transaction-level attribute of the main request element, which can have the following values:

•resource: Every top-level resource in the XML script is processed in one separate transaction. For example, this can be a content node with its complete layout. If an error occurs, all resources up to the one where the error was encountered have been fully processed; the resource where the error was encountered is not created, or, if it already existed, it is left unchanged. This is the default transaction level. The example script below shows how to specify it in request tag:

<request ... type=”update”transaction-level=”resource”>

•request: The entire XML script is executed in one transaction. If an error occurs, all database changes caused by the script are made undone, and the original state is restored. Note that using this level of transactionality might cause large and long-running database transactions if used in large XML scripts. As a result, you might encounter database errors caused by exceeding database limits on transaction duration or transaction log size, depending on the configuration of your database. The sample script below shows how to specify it in request tag:

<request ... type=”update”transaction-level=”request”>

Guidelines:
- It is recommended to have multiple small XML updates when using XMLAccess, especially when the network or database conditions are not very stabilized, or DX Portal and database may be in different firewall zones.
- For large updates, it is recommended to set HTTP connection and transaction timeout values in WebSphere Application Server, especially in a clustered environment. To do so, navigate Servers > Application Servers > WebSphere_Portal > Web container settings > Web container transport chains, on HTTP Inbound channel, and increase read and write timeout values to 180; and under Container Services > Transaction service, increase timeout to 3000.
- In the request tag, you can set "skip-cache-refresh" and "synchronous-cache-update" to adjust whether to refresh some internal caches. The first one determines whether to refreshes the caches at all in the a single run; the second determines whether to refresh the cache after each content node processing is complete, thus gives more granular control. You can use this to control whether or not page updates are to be visible immediately after pages are created, or if they are all visible at once after processing of all pages. slightly improve the processing speed if you want to process several XML requests in sequence. The default value for both is false. The second parameter depends on the first. If the first is set to true, then the second would be ignored.

In some special conditions, you can set a couple of configuration properties in WP ConfigService to avoid a race condition:
content.topology.writelock.timeout=<milliseconds> (default=25000)
- This setting controls the maximum wait time to obtain a writable model before issuing a timeout warning.
content.topology.writelock.dump=true|false (default=false)
- This setting controls if a JAVA core dump should be written in case of a timeout event for debugging.

USE XMLACCESS TO DELETE USER/GROUPS, ARE THEY BE DELETED FROM THE USER REPOSITORY?

The answer is yes. They are not only removed from DX portal databases, but also removed from the corresponding user repository configured for the DX Portal. All data associated with these users and groups stored in DX Portal

database are also removed by database constraints.

WHEN GENERATING EXPORT USING EXPORT.XML, THE RESULT OF "EXPORT-USERS" SET TO TRUE?

In the sample Export.xml, “export-users” is usually set to false. If it is set to true, the export will contains users and groups as <user..> and <group...> tags. The resulted XML file is equivalent to the full export using Export.xml plus the export using ExportAllUsers.xml. If there are invalid users/groups in the DX Portal DB, the export will fail. Thus it is not recommended to enable this parameter, when the focus is not on users and groups, but rather the other DX portal artifacts.

DOES THE ORDER OF TAGS IN THE EXPORTED XML FILE CHANGE?

The general orders of top-level tags in the XML file are the same, under "portal". For example, in an export generated using "Export.xml", they are shown as follows:
{{{
portal
global-settings
services-settings
language
action
virtual-resource
resource-type
markup
client
event-handler
protected-resource
skin
theme
web-app
content- node
credential-sement
url-mapping
wsrp-producer
task
user-resource
policy-node
application-role
wsrp-customized-portletinstance
custom-resource
rating
tag
filter-instance
}}}

However, XMLAccess cannot guarantee that the orders with the same type of artifacts are kept the same when running multiple times. The order of the sub tags can also change.

HOW TO FIND CURRENTLY SCHEDULED DATABASE CLEANUP TASK AND OTHER SCHEDULED TASK?

Use the following XML, you can export the currently scheduled tasks:

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="PortalConfig_7.0.0.xsd" type="export">
<portal action="locate">
<task action="export" name="com.ibm.portal.datastore.task.ResourceCleanup"/>
<task action="export" name="com.ibm.portal.ai.task.AITask_Purger"/>
<task action="export" name="com.ibm.portal.ai.task.AITask_PolicyHandler"/>
</portal>
</request>
```

Notice that you have to specify the name of each task specifically. I.e., you cannot export all the tasks using something like:

{code:} <task action="export" objectid="*"/>{code}
or
{code:} <task action="export" name="*"/>{code}

You can find out why from the result of the export which will be something like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- IBM WebSphere Portal/7.0 build wpnext_528_01 exported on Mon Nov 29 19:23:53 EST 2010 from
machine_name -->
<!-- 1 [task name=com.ibm.portal.datastore.task.ResourceCleanup] -->
<!-- 2 [task name=com.ibm.portal.ai.task.AITask_Purger] -->
<!-- 3 [task name=com.ibm.portal.ai.task.AITask_PolicyHandler] -->
<request build="wpnext_528_01" type="update" version="7.0.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="PortalConfig_7.0.0.xsd">
<portal action="locate">
<task action="update" name="com.ibm.portal.datastore.task.ResourceCleanup">
<dayOfWeek>6</dayOfWeek>
<startTime>20:00</startTime>
</task>
<task action="update" name="com.ibm.portal.ai.task.AITask_Purger">
<dayOfWeek>7</dayOfWeek>
<startTime>00:00</startTime>
</task>
<task action="update" name="com.ibm.portal.ai.task.AITask_PolicyHandler">
<startTime>22:30</startTime>
</task>
</portal>
<status element="all" result="ok"/>
</request>
```

From the export, we can see there are no objectIDs for the task. In addition, wildcard cannot be used on "name" attribute.

CAN I RUN MULTIPLE XMLACCESS SESSIONS SIMULTANEOUSLY?

The answer is no. Sometimes, when you run XMLAccess, you might see the following error,

<failure>
com.ibm.wps.command.CommandFailedException: EJPXA0166E: The XML Configuration Interface is currently busy with another request. Please try again later.
</failure>

This means you have another XMLAccess process running. XMLAccess tool prevents multiple sessions to be run at the same time.

This can also be the case of the following situation: When you have an XMLAccess command hangs or is stuck in a thread, you tried to kill it by interrupting it on the client side, such as Control+C, or "kill <processid>", the JAVA thread on the server side is still stuck and may not be terminated. Issuing another XMLAccess command will generate the same error. This kind of situations sometimes may only be resolved by restarting the server.

CAN I SCHEDULE A CLEANUP TASK TWICE EVERY WEEK?

For performance reasons, it is recommended that the cleanup of HCL Digital Experience database to be delayed to a time when traffic on the web site is light, because the database cleanup is normally CPU extensive and continuously running cleanup can degrade the system performance on the production system. HCL Digital Experience has a delayed database cleanup scheme which takes advantage of WebSphere Scheduler Service

All DX portal installations would have the default configuration for "scheduler.cleanup.enabled" set to **true**, and the time for the delayed clean up is 8 PM Saturday.

Because of this delay, the portal resources deleted are usually not purged from the database immediately. This could cause some problems. For example, after you delete a page, you could not create a new page with the same title; or after you just deleted a virtual portal, you are not able to create a new virtual portal with the same context.

The database cleanup task cannot be run more than once a day as a scheduled task in DX Portal database. In the scheduled task table, there can only exist at most one entry per task.

If you have two entries in the XML file, then the last would be the winner. For example, if you have:

<task action="create" name="com.ibm.portal.datastore.task.ResourceCleanup">
<dayOfWeek>5</dayOfWeek>
<startTime>7:00</startTime>
</task>

<task action="create" name="com.ibm.portal.datastore.task.ResourceCleanup">
<dayOfWeek>4</dayOfWeek>
<startTime>0:00</startTime>
</task>{code}

the database would be cleaned up at 12 AM on Thursday.

To be able to trigger the immediately cleanup, you can run XMLAccess with Task.xml (in directory ../PortalServer/doc/xml-samples). Read the comments in that XML file for different scheduling possibilities. As explained in the file, you can schedule the clean up using the scheduler task once daily, or once weekly, or once

monthly.

What if you want to run it twice every week, for example, once during the weekend and another in mid week?

You should be able to run XMLAccess with Task.xml at anytime to trigger a one time cleanup. With this flexibility, you can set up a scheduled Operating System job or cron job to run XMLAccess at the day and time you choose. As a matter of fact, you can schedule this XMLAccess task as many or few as you need.

WHAT ARE THE DIFFERENCES BETWEEN THE FULL EXPORTS GENERATED BY EXPORT.XML AND EXPORTRELEASE.XML?

Export.xml exports data from the release, customization, and community database domains; ExportRelease.xml exports data from the release database domain only
- When moving content of DX Portal database from one system to another, we normally recommend using ExportRelease.xml; it is not recommended to move customization data from one system to another;
- If the Release Builder is used to generate the incremental changes between releases, the full export XML files must be generated using ExportRelease.xml
- It's very useful to use Export.xml when debugging a problem.

If you closely examine the two files, you would see the differences are that in ExportRelease.xml, it specifies domain "rel" (for release). When a domain is specified, "export-users" is automatically set to false.
The export generated by Export.xml is a super set of the one generated by ExportRelease.xml. Data from customization and community domains will be included. Policy nodes will be generated. Because of these additional data, sometimes, the export action using Export.xml fails, but it would succeed if ExportRelease.xml is used. In those occasions, you may want to find out why. Some common reasons are orphaned data. For example, after users are deleted from the LDAP, the customization data remains in Portal Database.

One procedure we recommend is so-called "CleanupUsers'. Running XMLAccess against CleanupUsers.xml (in PortalServer/doc/xml-samples) to generate a list of invalid users and groups. If you run XMLAccess again with this output XML file as input, those invalid user and group entries would be deleted from DX Portal database, taking along with the resources owned by them.

IS THERE A RELATIONSHIP BETWEEN XMLACCESS AND VIRTUAL PORTALS?

When a Virtual Portal is created using the Manage Virtual Portals portlet, the new Virtual Portal's content is created when the file InitVirtualPortal,xml is imported to the Portal by the portlet.

Depending the installation of the DX Portal family products and installation options, there may be several different initialization XML script files under
<wp_profile>/installedApps/<node>/wps.ear/wps.war/VirtualPortal

You can choose which one to be used to populate the virtual portals when you use Manage Virtual Portals portlet. To change the init script, simply configure the portlet through Manage Portlets, and change the name of the file in parameter SCRIPT_INIT_VP.

Note that if any of the out-of-the-box applications are deleted or deactivated, you should not directly use these out-of-the-box virtual portal initialization XML scripts. Doing so, the creation of the virtual portals will likely fail due to the missing applications. You can customize these scripts by deleting or commenting out of those applications.

You can also create your own template initialization XML files, such that virtual portals created with these templates would contain your own custom applications.

If a Virtual Portal is created using the configuration task "create-virtual-portal" the new Virtual Portal's content should be established by importing an export of a Virtual Portal from another DX Portal environment using XMLAccess.

You can replace the default VP initialization XML files with your own. Sometimes, you may want to create your own VP template and use it to create multiple Vps. If you have deleted out of box applications from the base portal, the default initialization XML files should not be used, or they must be customized to delete the referenced applications from the files.

CAN I EMPTY A VIRTUAL PORTAL AND THEN IMPORT AN XML FILE GENERATED FROM ANOTHER VIRTUAL PORTAL?

Since the resources in a virtual portal are mostly links to those in the base portal, there does not exist a procedure to empty a virtual portal like what the ConfigEngine task "empty-virtual" does to the base portal. As a matter of fact, you should not delete the resources that are not scoped to the virtual portal only, because they may be used and shared by the base and other virtual portals.

The correct procedure should be the following:
- Prepare the XML file for the virtual portal import, i.e. generate full export from the virtual portal;
- Delete the virtual portal you plan to empty;
- Run XMLAccess against Task.xml to clean up the database;
- Set values in wkplc.properties for virtual portal title, context, realm, hostname (if any), and NLS file;
- Run ConfigEngine task "create-virtual-portal" to create a new virtual portal
- Import the XML file generated in the first step into the newly created virtual portal.

WHAT CAN I USE IMPORTXML PORTLET FOR?

The ImportXML portlet is intended for simple import/update jobs. It can only be used in the current DX portal or virtual portal the user is logged into, since you cannot specify a different URL. It cannot be used to export any resources, because one cannot specify the output file. Notice that if the import XML file contains update actions on some resources, the user who is logged in must have proper access on the target resources. The access control rules are applicable to the portlet, just like the command line XMLAccess.

CAN XMLACCESS BE USED TO UPDATE USERDN IN CUSTOMIZATION DATA?

The answer is yes. You can use the following procedures to test this out in a test environment before performing on a real user in production. As with any changes, be sure to take a backup of the Portal databases and the file system before proceeding.

1) Locate the following file:
/opt/IBM/WebSphere/PortalServer/doc/xml-samples/ExportUserResource.xml

2) Make a copy of this file to the /tmp directory. Open this file in a text editor. Modify

from:
<user-resource action="export" owner="uid=wpsadmin,o=defaultwimfilebasedrealm" domain="cust"/>

to:
<user-resource action="export" owner="uid=portaluser,ou=employee,o=ibm,c=us" domain="cust"/>

3) Save the file. Run XMLAccess with this file as an -in parameter, e.g.

./xmlaccess.sh -user wpsadmin -password wpsadmin -in /tmp/ExportUserResource.xml -out /tmp/exported.xml -url http://localhost:10039/wps/config

The above steps provides an XMLAccess script tailored to a specific user, uid=portaluser,ou=employee,o=ibm,c=us that does not interfere with other users data, AND, also will not contain settings that would impact the system as a whole.

The following steps could be performed for the real user in a production environment.

1) Perform the same steps to modify the ExportUserResource.xml file with the real user DN.

2) Run this as an input file to the production system - this can be done at any time without performance impact; however, on general principal it would be safest to perform this action during a maintenance window.

3) Open the output file (results from step #2) in a text editor. You should see the userDN throughout the file.

4) Find and replace the old userDN with the new userDN. Save the file.

5) Import the saved file to the Portal server with the updated userDN.

WHAT ARE THE LIMITATIONS OF XMLACCESS?

XMLAccess is a high level database utility, used to read, create, update, and delete resources in DX Portal database. Through PUMA and VMM, it can also add and delete users/groups in user repositories. Although it is a powerful tool for Portal administration, it also has some limitations.

•XMLAccess cannot be used to create a virtual portal, but it can be used to populate the content of virtual portals;
•it cannot be used to make configuration changes in WebSphere Application Server, for example, security configuration, or data source definition;
•it cannot copy the portlet WAR files or theme files from one system to another;
•it has limited functionality in JCR and Community domains. It is mainly used in the release and

customization domains.
•it cannot be used to change the custom properties used by Portal runtime services, such as ConfigService, PumaStoreService, or AccessControlService, to name a few.
•it cannot set Portal window or portlet states
•it cannot be used to run direct SQL statements on the Portal database. It is only a scripting tool which provides a concrete visualization and abstracts the need for direct SQL statements

HOW IS RELEASEBUILDER RELATED TO XMLACCESS?

ReleaseBuilder and XMLAccess are directly related. While ReleaseBuilder itself does not depend on XMLAccess to function, an XML file generated by the ReleaseBuilder (a "release") must be imported to the target Portal using XMLAccess.

ReleaseBuilder is designed to take as input 2 XML files exported from the SAME Portal host to determine or calculate the changes made between the times the first and the subsequent export were generated. This process is referred to as "building a release."

If Release_1.xml and Release_2.xml are the XML export files from a source Portal processed by Release Builder such that the output file is Revison_1.xml, then Revision_1.xml is meant to be imported to the target Portal as the XMLAccess input file. I.e.,

releasebuilder.sh -inOld Release_1.xml -inNew Release_2.xml -out Revision_1.xml

and

xmlaccess -in Release_1.xml -out <output_file.xml> -url <URL_of_target_Portal/wps/config>
-user <Portal_admin> -password <Portal_admin_pwd>

This XMLAccess command ensures that the changes in Revision_1.xml are applied to the target Portal server.

Notes:
•For the changes to be successfully applied to the target system, both the source and target system should be built to share the object IDs.
•XMLAccess requires Portal to be running, ReleaseBuilder does not.
•ReleaseBuilder should only take two XML files generated using ExportRelease.xml, not Export.xml. See question “**What are the differences between the exports using Export.xml vs ExportRelease.xml?**”
•XMLAccess will generate entries in Portal SystemOut.log, but ReleaseBuilder does not. ReleaseBuilder has its own SystemOut.log
•ReleaseBuilder has a "-debug" command line option, it is not recommended for normal use.

WHERE CAN I FIND SAMPLE XML FILES?

Most of XMLAccess sample files are located in directory <PortalServer_root>/doc/xml-samples. It is recommended to keep these files as they are. If you need to customize these files to test different scenarios, make copies and rename them, or copy them to a different location.

Sometimes, you may not find the exact sample XML file for what you are trying to accomplish. We recommend you take a full export of the Portal content, using either Export.xml or ExportRelease.xml, depending on the tags you want to review. Then extract the top level component and adjust to your needs. For example, if you want to export all URL mappings, but there is not a sample file, you can easily create one, like the following (save it as ExportAllURLMappingContexts.xml),

```
<?xml version="1.0" encoding="UTF-8"?>

<request
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="PortalConfig_1.4.xsd"
type="export">

<!-- This XML script exports all URL mapping contexts. -->
<portal action="locate">
<url-mapping-context objectid="*" action="export"/>
</portal>
</request>
```

Sometimes, you have to try and test some cases before you can get it right.

Note that in every XML script, your action must be start from a top level element under "portal". For example, you cannot make a change directly to a "portlet", rather the change must be done starting at "web-app".

REFERENCES

HCL Digital Experience Product Documentation 9.5 - Working with XML configuration interface
https://help.hcltechsw.com/digital-experience/9.5/admin-system/adxmltsk.html

HCL Digital Experience Product Documentation 9.5 - ReleaseBuilder
https://help.hcltechsw.com/digital-experience/9.5/deploy/dep_rbabout.html