AEM 6.4 Updates

Developer Guide

Jindal, Tanuj Mohan (Cognizant)

Tanuj.jindal@cognizant.com

Contents

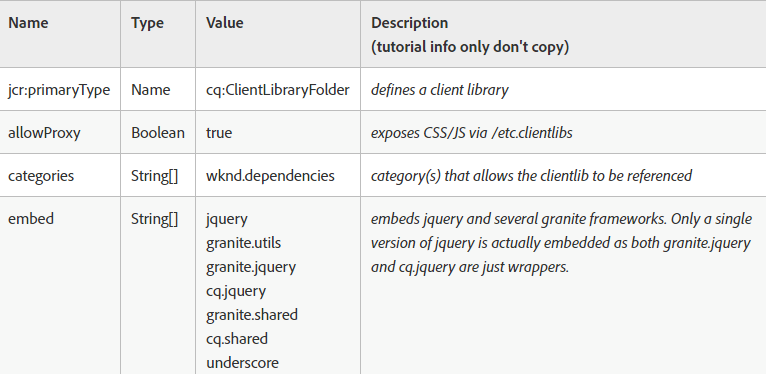
[1) Client Libraries: 2](#_Toc516006556)

[2) Configurations 3](#_Toc516006557)

[3) New Mixins 5](#_Toc516006558)

[4) Reference: 6](#_Toc516006559)

# 1) Client Libraries:

* Due to removal of /etc in AEM 6.4 client libraries needs to be moved away from /etc/designs.
* Adobe recommends locating these client libraries now under /apps and making them available to web using the proxy servlet.
* Recommended path : **/apps/<project folder>/clientlibs**
* New Property allowProxy has been introduced to help client libraries residing in /apps to be exposed via /etc.clientlibs path using proxy servlet.
* It is important to use allowProxy for better managing dispatcher rules wherein we need to publish the client libraries residing inside /apps to the outer web, but still hide rest of the code base. (Components, templates, configurations etc...)
* No change in the way client libraries is included or loaded onto page.

Debugging Client Libraries

Various tools to debug client libraries:

* Dump Libs - Lists all of the client libraries registered in the AEM instance.

*URL: <host>/libs/granite/ui/content/dumplibs.html*

* Test Output- allows a user to see the expected HTML output of clientlib includes based on category.

*URL: <host>/libs/granite/ui/content/dumplibs.test.html*

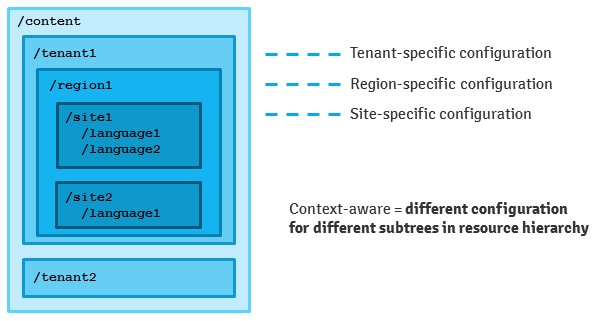
* Libraries Dependencies validation**-**highlights any dependencies or embedded categories that cannot be found.

*URL: <host>/libs/granite/ui/content/dumplibs.validate.html*

* Rebuild Client Libraries - allows a user to force AEM to rebuild all of the client libraries or invalidate the cache of client libraries. This tool is particularly effective when developing with LESS as this can force AEM to re-compile the generated CSS. In general it is more effective to Invalidate Caches and then perform a page refresh versus rebuilding all of the libraries.

*URL: <host>/libs/granite/ui/content/dumplibs.rebuild.html*

# 2) Configurations

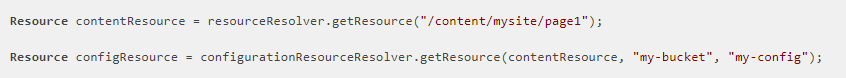
* Again due to removal of /etc all configurations needs to be relocated to /conf.
* This helps enabling use of Apache Sling's *Context Aware Configuration* feature.
* Context Aware Configuration feature allows content in one part of the repository to be overlayed multiple times by content in other parts of the repository. 

There are two different JAVA API’s available to make use of this feature:

* + Context-Aware Resources: 'Low-level' API for accessing configuration resources (which can be anything, e.g. workflow definitions, template definitions or key-value pairs)
  + Context-Aware Configurations: 'High-level' API for accessing configuration data (key/value pairs)

Context – Aware Resource:

Example to get a configuration resource for a content at /content/mysite/page1



Parameter:

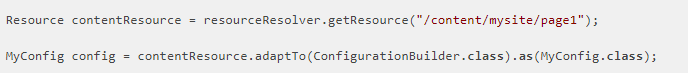
*contentResource*: Resource path of the page.

*bucketName:* Bucket name is used to separate different types of configuration resource. Example workflow config resource or template definitions resource or key/value pair resource.

*configName*: The configuration name defines configuration we want to retrieve. The name can be a relative path as well (e.g. "sub1/my-config").

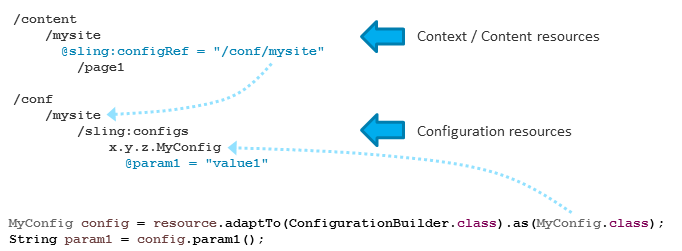
Context-Aware Configurations:

Example to get a context-aware configuration at /content/mysite/page1



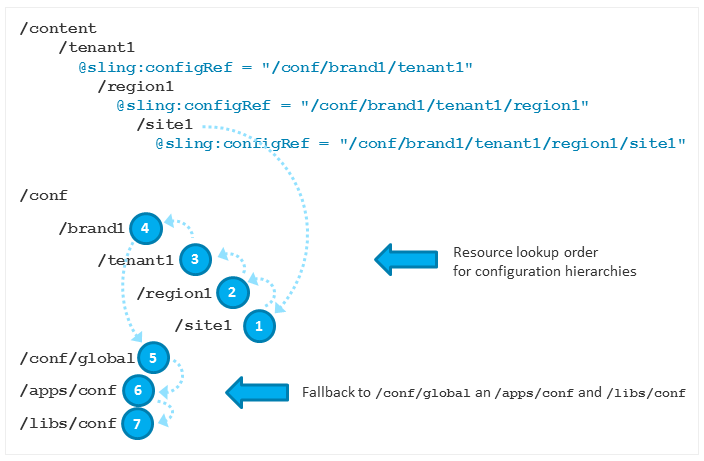
A content is linked to a configuration using sling:configRef property as shown below.

Actual config should be of type sling:configs.



Resource Lookup order:

If we get the context-aware configuration via the API for any resource below /content/tenant1/region1/site1 it is looked up in this path in below order:



Resource Inheritance:

Resource inheritance can be achieved by defining a property sling:configCollectionInherit on the configuration resource. If defined the children of the next resource that is found in the configuration resource lookup order are combined with the children of the current configuration resource and a merged list is returned.

Example: The result of this example is: C, A, B. It would by just C if the sling:configCollectionInherit is not set.



# 3) New Mixins

In order to make it easier for developer to understand which area of /libs are safe to use or extend, AEM has introduced new mixins which indicates the same.

* Public (granite:PublicArea) - Defines a node as public so that it can overlaid, inherited (sling:resourceSuperType) or used directly (sling:resourceType).
* Abstract (granite:AbstractArea) - Defines a node as abstract. Nodes can be overlaid or inherited (sling:resourceSupertype) but must not be used directly (sling:resourceType).
* Final (granite:FinalArea) - Defines a node as final. Nodes classified as final cannot be overlaid or inherited. Final nodes can be used directly via sling:resourceType. Subnodes under final node are considered internal by default
* Internal (granite:InternalArea) - Defines a node as internal. Nodes classified as internal cannot be overlaid, inherited, or used directly. These nodes are meant only for internal functionality of AEM

# 4) Reference:

<https://helpx.adobe.com/experience-manager/6-4/sites/deploying/using/sustainable-upgrades.html>

<https://helpx.adobe.com/experience-manager/kt/sites/using/getting-started-wknd-tutorial-develop/part3.html>

<https://helpx.adobe.com/experience-manager/6-4/sites/deploying/using/repository-restructuring-in-aem64.html>

<https://sling.apache.org/documentation/bundles/context-aware-configuration/context-aware-configuration.html>

<http://sling.apache.org/documentation/bundles/context-aware-configuration/context-aware-configuration-default-implementation.html>

<https://blogs.perficient.com/adobe/2018/03/15/fyi-seismic-changes-coming-in-aem-6-4/>