**CUSTOM NODE TYPES IN AEM**

As CQ is based on Sling and uses a JCR repository, node types offered by both of these are available for use:

* [JCR Node Types](http://www.day.com/specs/jcr/2.0/3_Repository_Model.html#NodeTypes)
* [Sling Node Types](https://cwiki.apache.org/confluence/display/SLING/Sling+Node+Types)

In addition to these. CQ provides a range of custom node types.

**Audit**

**cq:AuditEvent**

**Description**

Defines the node type of an audit event node.

* @prop cq:time
* @prop cq:userid
* @prop cq:path
* @prop cq:type
* @prop cq:category
* @prop cq:properties

**Definition**

[cq:AuditEvent]  
  - \* (undefined)  
  - \* (undefined) multiple  
  + \* (nt:base) = nt:base multiple version  
- cq:time (date)  
- cq:userid (string)  
- cq:path (string)  
- cq:type (string)  
- cq:category (string)  
- cq:properties (binary)

**Comment**

**cq:Comment**

**Description**

Defines the nodetype of a comment node.

* @prop userIdentifier  
  The user identifier.

**Definition**

[cq:Comment] > mix:title, mix:created, mix:language, nt:unstructured, cq:Taggable  
- email (string)  
- ip (string)  
- referer (string)  
- url (string)  
- userAgent (string)  
- userIdentifier (string)  
- authorizableId (string)

**cq:CommentAttachment**

**Description**

Defines the node type of a comment attachment node

**Definition**

[cq:CommentAttachment] > nt:file  
  - \* (undefined)  
  - \* (undefined) multiple

**cq:CommentContent**

**Description**

Defines the node type of a comment content node

**Definition**

[cq:Comment] > mix:title, mix:created, mix:language, nt:unstructured, cq:Taggable  
- email (string)  
- ip (string)  
- referer (string)  
- url (string)  
- userAgent (string)  
- userIdentifier (string)  
- authorizableId (string)

**cq:GeoLocation**

**Description**

A mixin that defines a geographic location in decimal degrees (DD)

* @prop latitude  
  latitude encoded as double using decimal degrees
* @prop longitude  
  longitude encoded as double using decimal degrees

**Definition**

[cq:GeoLocation]  
mixin  
- latitude (double)  
- longitude (double)

**cq:Trackback**

**Description**

Defines the node type of a trackback node.

**Definition**

[cq:Trackback] > mix:title, mix:created, mix:language, nt:unstructured

**Core**

**cq:Page**

**Description**

Defines the default CQ page.

* @node jcr:content  
  Primary content of the page.

**Definition**

[cq:Page] > nt:hierarchyNode  
    orderable  
    + jcr:content (nt:base) = nt:unstructured copy primary  
    + \* (nt:base) = nt:base version

**cq:PseudoPage**

**Description**

Defines a mixin type that marks nodes as pseudo pages. This means they can be adapted for Page and WCM editing support.

**Definition**

[cq:PseudoPage] mixin

**cq:PageContent**

**Description**

Defines the default node for page content, with the minimum properties as used by WCM.

* @prop jcr:title  
  Title for the page.
* @prop jcr:description  
  Description of this page.
* @prop cq:template  
  Path to the template used to create the page.
* @prop cq:allowedTemplates  
  List of regular expressions used to determine the path(s) to allowed template.
* @prop pageTitle  
  Title usually displayed in the <title> tag.
* @prop navTitle  
  Title usually used in navigation.
* @prop hideInNav  
  Specifies whether the page should be hidden in the navigation.
* @prop onTime  
  Time when this page becomes valid.
* @prop offTime  
  Time when this page becomes invalid.
* @prop cq:lastModified  
  Date the page (or its paragraphs) was last modified.
* @prop cq:lastModifiedBy  
  Last user to change the page (or its paragraphs).
* @prop jcr:language  
  The language of page content.

**Note:**

It is not compulsory for page content to use this type.

**Definition**

[cq:PageContent] > nt:unstructured, mix:title, mix:created, cq:OwnerTaggable, sling:VanityPath, cq:ReplicationStatus, sling:Resource  
    orderable  
    - cq:template (string)  
    - cq:allowedTemplates (string) multiple  
    - pageTitle (string)  
    - navTitle (string)  
    - hideInNav (boolean)  
    - onTime (date)  
    - offTime (date)  
    - cq:lastModified (date)  
    - cq:lastModifiedBy (string)  
    - cq:designPath (string)  
    - jcr:language (string)

**cq:Template**

**Description**

Defines a CQ template.

* @node jcr:content  
  Default content for new pages.
* @node icon.png  
  A file that holds a characteristic icon.
* @node thumbnail.png  
  A file that holds a characteristic thumbnail image.
* @node workflows  
  Auto assign workflow configuration. The configuration will follow the structure below:  
      + workflows  
        + name1  
          - cq:path  
          - cq:workflowName
* @prop allowedParents  
  Regular expression patterns to determine the path(s) to templates allowed as parent templates.
* @prop allowedChildren  
  Regular expression patterns to determine the path(s) to templates allowed as child templates.
* @prop ranking  
  Position within the list of templates in the create page dialog.

**Definition**

[cq:Template] > nt:hierarchyNode, mix:title  
    - \* (undefined)  
    - \* (undefined) multiple  
    + \* (nt:base) = nt:base multiple version  
    + jcr:content (nt:base) copy  
    + icon.png (nt:file) copy  
    + thumbnail.png (nt:file) copy  
    + workflows (nt:base) copy  
    - allowedParents (string) multiple  
    - allowedChildren (string) multiple  
    - ranking (long)

**cq:Component**

**Description**

Defines a CQ component.

* @prop jcr:title   
  Title for the component.
* @prop jcr:description   
  Description of the component.
* @node dialog   
  Primary dialog.
* @prop dialogPath  
  Primary dialog path (alternative to dialog).
* @node design\_dialog   
  Design dialog.
* @prop cq:cellName   
  Name of the design cell.
* @prop cq:isContainer   
  Indicates whether this is a container component. This forces the cell names of child components to be used instead of path names. For example, the **parsys** is a container component. If this value is not defined, the check is made based on the existence of a **cq:childEditConfig**.
* @prop cq:noDecoration   
  If true, no decoration **div** tags are drawn when including this component.
* @node cq:editConfig   
  The configuration that defines the parameters for the edit bar.
* @node cq:childEditConfig  
  The edit configuration that is inherited by child components.
* @node cq:htmlTag   
  Defines additional tag attributes that are added to the "surrounding" **div** tag when the component is included.
* @node icon.png   
  A file that holds a characteristic icon.
* @node thumbnail.png   
  A file that holds a characteristic thumbnail image.
* @prop allowedParents   
  Regular expression patterns to determine the path(s) of components that are allowed as parent components.
* @prop allowedChildren  
  Regular expression patterns to determine the path(s) of components that are allowed as child components.
* @node virtual   
  Contains subnodes that reflect virtual components used for the component drag and drop.
* @prop componentGroup  
  Name of the component group, used for the component drag and drop.
* @node cq:infoProviders   
  Contains subnodes, each of which has a property **className** that refers to a **PageInfoProvider**.

**Definition**

[cq:Component] > nt:folder, mix:title, sling:ResourceSuperType  
    - \* (undefined)  
    - \* (undefined) multiple  
    + \* (nt:base) = nt:base multiple version  
    + dialog (nt:base) = nt:unstructured copy  
    - dialogPath (string)  
    + design\_dialog (nt:base) = nt:unstructured copy  
    - cq:cellName (string)  
    - cq:isContainer (boolean)  
    - cq:noDecoration (boolean)  
    + cq:editConfig (cq:EditConfig) = cq:EditConfig copy  
    + cq:childEditConfig (cq:EditConfig) = cq:EditConfig copy  
    + cq:htmlTag (nt:base) = nt:unstructured copy  
    + icon.png (nt:file) copy  
    + thumbnail.png (nt:file) copy  
    - allowedParents (string) multiple  
    - allowedChildren (string) multiple  
    + virtual (nt:base) = sling:Folder copy  
    - componentGroup (string)  
    + cq:infoProviders (nt:base) = nt:unstructured copy

**cq:ComponentMixin**

**Description**

Defines a CQ Component as mixin type.

**Definition**

[cq:ComponentMixin] > cq:Component mixin

**cq:EditConfig**

**Description**

Defines the configuration for the "editbar".

* @prop cq:dialogMode  
  Mode of the dialog:
  + **floating** - for a normal, floating dialog
  + **inline** - inline editing
  + **auto** - automatic detection (depending on available space)
* @node cq:inplaceEditing   
  Inplace editing configuration for this component.
* @prop cq:layout   
  Layout of the edit bar:
  + **editbar** - edit bar
  + **rollover** - roll over frame
  + **auto** - automatic detection
* @node cq:formParameters   
  Additional parameters to add to the dialog form.
* @prop cq:actions   
  List of actions (edit bar buttons, or menu items).
* @node cq:actionConfigs   
  Widget configurations for edit bar or menu items.
* @prop cq:emptyText   
  Text to be displayed if no visual content is present.
* @node cq:dropTargets   
  Collection of **{@link cq:DropTargetConfig}** nodes.

**Definition**

[cq:EditConfig] > nt:unstructured, nt:hierarchyNode  
    orderable  
    - cq:dialogMode (string) < 'auto', 'floating', 'inline'  
    - cq:layout (string) < 'editbar', 'rollover', 'auto'  
    + cq:formParameters (nt:base) = nt:unstructured  
    - cq:actions (string) multiple  
    + cq:actionConfigs (nt:base) = nt:unstructured  
    - cq:emptyText (string)  
    + cq:dropTargets (nt:base) = nt:unstructured  
    + cq:listeners (nt:base) = cq:EditListenersConfig

**cq:DropTargetConfig**

**Description**

Configures one drop target of a component. The name of the this node will be used as an ID for drag and drop.

* @prop accept   
  List of mime types accepted by this drop target; e.g. **["image/\*"]**
* @prop groups   
  List of drag and drop groups that accept a source.
* @prop propertyName   
  Name of the property used to store the reference.

**Definition**

[cq:DropTargetConfig] > nt:unstructured  
    orderable  
    - accept (string) multiple  
    - groups (string) multiple  
    - propertyName (string)  
    + parameters (nt:base) = nt:unstructured

**cq:VirtualComponent**

**Description**

Defines a virtual CQ component. These are currently used only for the new component drag and drop wizard.

* @prop jcr:title   
  Title of this component.
* @prop jcr:description   
  Description of this component.
* @node cq:editConfig   
  Edit configuration that defines the parameters for the edit bar.
* @node cq:childEditConfig   
  Edit configuration that is inherited by child components.
* @node icon.png  
  A file that holds a characteristic icon.
* @node thumbnail.png   
  A file that holds a characteristic thumbnail image.
* @prop allowedParents   
  Regular expression patterns to determine path(s) of components that are allowed as parent components.
* @prop allowedChildren  
  Regular expression patterns to determine path(s) of components that are allowed as child components.
* @prop componentGroup   
  Name of the component group for the component drag and drop.

**Definition**

[cq:VirtualComponent] > nt:folder, mix:title  
    - \* (undefined)  
    - \* (undefined) multiple  
    + \* (nt:base) = nt:base multiple version  
    + cq:editConfig (cq:EditConfig) = cq:EditConfig copy  
    + icon.png (nt:file) copy  
    + thumbnail.png (nt:file) copy  
    - allowedParents (string) multiple  
    - allowedChildren (string) multiple  
    - componentGroup (string)

**cq:EditListenersConfig**

**Description**

Defines the (client side) listeners to be executed on an edit event. The values must either reference a valid client side listener function or contain a predefined shortcut:

* REFRESH\_PAGE
* REFRESH\_SELF
* REFRESH\_PARENT
* @prop aftercreate   
  Fires after a component has been created.
* @prop afteredit   
  Fires after a component has been edited (modified).
* @prop afterdelete   
  Fires after a component has been deleted.
* @prop afterinsert   
  Fires after a component has been added to this container.
* @prop afterremove   
  Fires after a component has been removed from this container.
* @prop aftermove   
  Fires after components have been moved in this container.

**Definition**

[cq:EditListenersConfig]  
    - \* (undefined)  
    - \* (undefined) multiple  
    + \* (nt:base) = nt:base multiple version  
    - aftercreate (string)  
    - afteredit (string)  
    - afterdelete (string)  
    - afterinsert (string)  
    - afterremove (string)  
    - aftermove (string)

**DAM**

**dam:AssetContent**

**Description**

Content of a DAM asset.

**Definition**

[dam:AssetContent] > nt:unstructured  
 + metadata (nt:unstructured)  
 + renditions (nt:folder)

**dam:Asset**

**Description**

DAM asset.

**Definition**

[dam:Asset] > nt:hierarchyNode  
 + jcr:content (dam:AssetContent) = dam:AssetContent copy primary  
 + \* (nt:base) = nt:base version

**dam:Thumbnail**

**Description**

Thumbnail to represent a DAM asset.

**Definition**

[dam:Thumbnails]  
  mixin  
  + dam:thumbnails (nt:folder)

**Delivery Container List**

**cq:containerList**

**Description**

Container List.

**Definition**

[cq:containerList]  
    mixin

**Delivery Page**

**cq:Cq4PageAttributes**

**Description**

**cq:attributes** is the node type for the ContentBus version tags. This node only has a series of properties; of which three are predefined "created", "csd", and "timestampe".

* @prop created (long) mandatory copy  
  Timestamp of creation of the version information, generally the time of checkin of the previous version or time of page creation.
* @prop csd (string) mandatory copy  
  csd standard attribute, copy of the cq:csd property of the page node
* @prop timestampe (long) mandatory copy  
  Timestamp of last version modification, generally checkin time.
* @prop \* (string) copy  
  Additional attributes, versioned with the parent node.

**Definition**

[cq:Cq4PageAttributes] > nt:base  
  - created (long) mandatory copy  
  - csd (string) mandatory copy  
  - timestampe (long) mandatory copy  
  - \* (string) copy

**cq:Cq4ContentPage**

**Description**

The node type **cq:contentPage** contains the property and child node definitions for ContentBus content pages. Only when this mixin type is added to a node of type "cq:page", a node becomes a ContentBus content page.

The items in a "cq:Cq4ContentPage" are:

* @prop cq:csd  
  The ContentBus CSD of the page.
* @node cq:content  
  The content of the page. This child node does not exist if the page node is in state "Existing without content" or "Deleted".
* @node cq:attributes  
  The list of page attributes, which were formerly known as version tags. This node is mandatory for the cq:contentPage type. The attrbutes node is versioned, when the page is node is versioned.

**Definition**

[cq:Cq4ContentPage]  
    - cq:csd (string) mandatory copy  
    + cq:attributes (cq:Cq4PageAttributes)

**Importer**

**cq:PollConfig**

**Description**

Poll configuration.

* @prop source (String) mandatory  
  Data source URI, this is required and must not be empty
* @prop target (String)  
  The target location where data retrieved from the data source is stored. This is optional and defaults to the cq:PollConfig node.
* @prop interval (Long)  
  The interval in seconds at which to poll for new or updated data from the data source. This is optional and defaults to 30 Minutes (1800 seconds).
* [Creating Custom Data Importer Services for Adobe Experience Manager](https://helpx.adobe.com/experience-manager/using/polling.html)

**Definition**

[cq:PollConfig]  
    mixin  
  - source (String) mandatory  
  - target (String)  
  - interval (Long)

**cq:PollConfigFolder**

**Description**

Convenience primary node type to easily create poll configuration nodes.

**Definition**

[cq:PollConfigFolder] > sling:Folder, cq:PollConfig

**Location**

**cq:GeoLocation**

**Description**

A mixin that defines a geographic location in decimal degrees (DD).

* @prop latitude    
  Latitude encoded as double using decimal degrees.
* @prop longitude   
  Longitude encoded as double using decimal degrees.

**Definition**

[cq:GeoLocation]  
  mixin  
  - latitude (double)  
  - longitude (double)

**Mailer**

**cq:mailerMessage**

**Description**

MailerService nodetypes. The mailer uses nodes having this mixin as root nodes of message definitions.

**Definition**

[cq:mailerMessage]  
    mixin  
    - messageStatus (string)  
    = 'new'  
    mandatory autocreated

**MSM**

**cq:LiveRelationship**

**Description**

Defines a LiveRelationship mixin. A master node and a slave node can be virtually linked through a LiveRelationship.

**Definition**

[cq:LiveRelationship] mixin  
    - cq:lastRolledout (date)  
    - cq:lastRolledoutBy (string)  
    - cq:sourceUUID (string)

**cq:LiveSync**

**Description**

Defines a LiveSync mixin. If a node is involved in a LiveRelationship with a master node as a slave, it  
is marked a LiveSync.

* @prop cq:master   
  Path of the master node of the LiveRelationship.
* @prop cq:isDeep   
  Defines if the relationship is available for children.
* @prop cq:syncTrigger   
  Defines when is triggered the sync.
* @node \* LiveSyncAction   
  Actions to perform on sync

**Definition**

[cq:LiveSync] > cq:LiveRelationship mixin orderable  
    + \* (cq:LiveSyncAction) = cq:LiveSyncAction  
    + cq:LiveSyncConfig (nt:base) = cq:LiveSyncConfig

**cq:LiveSyncCancelled**

**Description**

Defines a LiveSyncCancelled mixin. Cancel the LiveSync behavior of a slave node which may be involded in a LiveRelationship owing to one of its parents.

* @prop cq:isCancelledForChildren   
  Defines whether a LiveSync is cancelled; also for children.

**Definition**

[cq:LiveSyncCancelled] > cq:LiveRelationship mixin  
    - cq:isCancelledForChildren (boolean)

**cq:LiveSyncAction**

**Description**

Defines a LiveSyncAction attached to a LiveSync.

* @prop name   
  Action name.
* @prop value   
  Action value.

**Definition**

[cq:LiveSyncAction] > nt:unstructured

**cq:LiveSyncConfig**

**Description**

Live Sync configuration.

**Definition**

[cq:LiveSyncConfig]  
    - cq:master (string) mandatory  
    - cq:isDeep (boolean)  
    - cq:trigger (string) /\*\* deprecated \*\*/

For CQ 5.4 add to the end of list:

**- cq:rolloutConfigs (string) multiple  deprecated \*\*/**

**cq:BlueprintAction**

**Description**

Blueprint action.

**Definition**

[cq:BlueprintAction] > nt:unstructured

**Platform**

**cq:Console**

**Description**

Defines the nodetype of a console node.

**Definition**

[cq:Console] > sling:VanityPath, mix:title  
    mixin

**Replication**

**cq:ReplicationStatus**

**Description**

Defines replication status information mixin.

* @prop cq:lastPublished   
  The date the page was last published (not used anymore).
* @prop cq:lastPublishedBy   
  The user who published the page last (not used anymore).
* @prop cq:lastReplicated   
  The date the page was last replicated.
* @prop cq:lastReplicatedBy   
  The user that replicated the page last.
* @prop cq:lastReplicationAction   
  The replication action: activate or deactivate.
* @prop cq:lastReplicationStatus   
  The replication status (not used anymore).

**Definition**

[cq:ReplicationStatus]  
    mixin  
    - cq:lastPublished (date) ignore  
    - cq:lastPublishedBy (string) ignore  
    - cq:lastReplicated (date) ignore  
    - cq:lastReplicatedBy (string) ignore  
    - cq:lastReplicationAction (string) ignore  
    - cq:lastReplicationStatus (string) ignore

**Security**

**cq:ApplicationPrivilege**

**Description**

Defines an application privilege.

**Definition**

[cq:ApplicationPrivilege] mixin

**cq:PrivilegeAcl**

**Description**

Defines an application privilege ACL.

* @prop cq:isPathDependent
* @node \* ACEs

**Definition**

[cq:PrivilegeAcl] > cq:ApplicationPrivilege mixin orderable  
    - cq:isPathDependent (boolean)  
    + \* (cq:PrivilegeAce) = cq:PrivilegeAce

**cq:PrivilegeAce**

**Description**

Defines an application privilege ACE.

* @prop path
* @prop deny

**Definition**

[cq:PrivilegeAce]  
    - path mandatory  
    - deny (boolean)

**cq:ApplicationPrivilege**

**Description**

Defines an application privilege.

**Definition**

[cq:ApplicationPrivilege] mixin

**cq:PrivilegeAcl**

**Description**

Defines an application privilege ACL.

* @prop cq:isPathDependent
* @node \* ACEs

**Definition**

[cq:PrivilegeAcl] > cq:ApplicationPrivilege mixin orderable  
    - cq:isPathDependent (boolean)  
    + \* (cq:PrivilegeAce) = cq:PrivilegeAce

**cq:PrivilegeAce**

**Description**

Defines an application privilege ACE.

* @prop path
* @prop deny

**Definition**

[cq:PrivilegeAce]  
    - path mandatory  
    - deny (boolean)

**Site Importer**

**cq:ComponentExtractorSource**

**Description**

Defines a mixin type that marks files that can be opened with component extractor.

**Definition**

[cq:ComponentExtractorSource] mixin

**Tagging**

**cq:Tag**

**Description**

Defines a single tag, but can also contain tags, thus creating a taxonomy

**Definition**

[cq:Tag] > nt:base, mix:title  
    - sling:resourceType (String)  
    - \* (undefined) multiple  
    - \* (undefined)  
    + \* (nt:base) = cq:Tag version

**cq:Taggable**

**Description**

Abstract base mixin for taggable content.

* @node cq:tags

**Definition**

[cq:Taggable]  
    - cq:tags (string) multiple

**cq:OwnerTaggable**

**Description**

Only authors/owners are allowed to tag the content (moderated/administered tagging).

**Definition**

[cq:OwnerTaggable] > cq:Taggable

**cq:UserTaggable (ToDo)**

**Description**

Any user/public website can tag the content (Web2.0 style), used inside **cq:userContent**.

**Definition**

[cq:UserTaggable] > cq:Taggable  
    mixin

**cq:AllowsUserContent**

**Description**

Adds a **cq:userContent** subnode that can be modified by users; each user will have their own**cq:userContent/<*userid*>** subnode, that typically has the mixin **cq:UserTaggable**.

**Definition**

[cq:AllowsUserContent]  
    mixin  
    + cq:userContent (nt:unstructured)

TODO: extended variant, more explicitly defining the cq:userContent tree

[cq:AllowsUserContent]  
   mixin  
    + cq:userContent (cq:UserContent)

**cq:UserContent**

**Description**

Can be modified by users.

**Definition**

[cq:UserContent] > nt:unstructured  
    // userids  
    + \* (cq:UserData)  
    // other content  
    + \* (nt:base)

**cq:UserData**

**Description**

User data.

**Definition**

[cq:UserData] > nt:unstructured, cq:UserTaggable

**Widgets**

**cq:ClientLibraryFolder**

**Description**

Client library folder.

**Definition**

[cq:ClientLibraryFolder] > sling:Folder  
  - categories (string) multiple  
  - dependencies (string) multiple

**cq:Widget**

**Description**

Widget.

**Definition**

[cq:Widget] > nt:unstructured orderable  
  - xtype (string)  
  - name (string)  
  - title (string)  
  + items (nt:base) = cq:WidgetCollection copy

**cq:WidgetCollection**

**Description**

Widget collection.

**Definition**

[cq:WidgetCollection] > nt:unstructured  
  orderable  
  + \* (cq:Widget) = cq:Widget copy

**cq:Dialog**

**Description**

Dialog.

**Definition**

[cq:Dialog] > cq:Widget orderable

**cq:Panel**

**Description**

Panel.

**Definition**

[cq:Panel] > cq:Widget orderable

**cq:TabPanel**

**Description**

Tab panel.

**Definition**

[cq:TabPanel] > cq:Panel orderable  
  - activeTab (long)

**cq:Field**

**Description**

Field.

**Definition**

[cq:Field] > cq:Widget orderable  
  - fieldLabel (string)  
  - value (string)  
  - ignoreData (boolean)

**Wiki**

**wiki:Topic**

**Description**

Wiki topic.

**Definition**

[wiki:Topic] > nt:unstructured, nt:hierarchyNode, mix:versionable, mix:lockable  
  + \* (wiki:Topic) version  
  + wiki:attachments (nt:folder) = nt:folder version  
  + wiki:properties (wiki:Properties) = wiki:Properties copy  
  - wiki:text (string) mandatory primary  
  - wiki:lastModified (date) mandatory  
  - wiki:lastModifiedBy (string) mandatory  
  - wiki:topicName  
  - wiki:topicTitle  
  - wiki:lockedBy  
  - wiki:logMessage (string)  
  - wiki:quietSave (boolean)

**wiki:User**

**Description**

Wiki user.

**Definition**

[wiki:User] mixin  
  - wiki:subscriptions (string) multiple

**wiki:Properties**

**Description**

Wiki properties.

**Definition**

[wiki:Properties]  
  - wiki:isGlobal (boolean)  
  - \* (undefined)

**Workflow**

**cq:Workflow**

**Description**

Represents a workflow instance.

**Definition**

[cq:Workflow] > nt:base, mix:referenceable  
 - modelId (String)  
 - modelVersion (String)  
 - startTime (Date)  
 - endTime (Date)  
 - initiator (String)  
 - \* (undefined)  
 - \* (undefined) multiple  
 - sling:resourceType (String)  
   = "cq/workflow/components/instance" mandatory autocreated  
 + workflowStack (nt:unstructured)  
 + wait (nt:unstructured)  
 + orTab (nt:unstructured)  
 + data (cq:WorkflowData)  
 + history (nt:unstructured)  
 + metaData (nt:unstructured)  
 + workItems (nt:unstructured)

**cq:WorkItem**

**Description**

Work item.

**Definition**

[cq:WorkItem]  
 - assignee (String)  
 - workflowId (String)  
 - nodeId (String)  
 - startTime (Date)  
 - endTime (Date)  
 - dueTime (Date)  
 - sling:resourceType (String)  
   = "cq/workflow/components/workitem" mandatory autocreated  
 + metaData (nt:unstructured)

**cq:Payload**

**Description**

Payload.

**Definition**

[cq:Payload]  
 - path (Path)  
 - uuid (String)  
 - jcr:url (String)  
 - binary (Binary)  
 - javaObject (String)  
 - \* (undefined)  
 - \* (undefined) multiple

**cq:WorkflowData**

**Description**

Workflow data.

**Definition**

[cq:WorkflowData]  
 - \* (undefined)  
 - \* (undefined) multiple  
 + payload (cq:Payload)  
 + metaData (nt:unstructured)  
   copy

**cq:WorkflowModel**

**Description**

Auto assign workflow configuration. The configuration will follow this structure below:  
 workflows  
     + name1  
         - cq:path  
         - cq:workflowName  
  + workflows (nt:base)

**Definition**

[cq:WorkflowModel] > nt:base, mix:versionable  
 orderable  
 - title (String)  
 - description (String)  
 - sling:resourceType (String)  
   = "cq/workflow/components/model" mandatory autocreated  
 + nodes (nt:unstructured)  
   copy  
 + transitions (nt:unstructured)  
   copy  
 + metaData (nt:unstructured)  
   copy

**cq:WorkflowNode**

**Description**

Workflow node.

**Definition**

[cq:WorkflowNode]  
 orderable  
 - title (String)  
 - description (String)  
 - maxIdleTime (long)  
 - type (String)  
 - \* (undefined)  
 - \* (undefined) multiple  
 + metaData (nt:unstructured)  
   copy  
 + timeoutConfiguration (nt:unstructured)  
   copy

**cq:WorkflowTransition**

**Description**

Workflow transition.

**Definition**

[cq:WorkflowTransition]  
 orderable  
 - from (String)  
 - to (String)  
 - rule (String)  
 + metaData (nt:unstructured)  
   copy

**cq:OrTab**

**Description**

Or tab.

**Definition**

[cq:OrTab]  
 - workflowId (String) // not compulsory as this node will   
   already be attached to the workflow node  
 - nodeId (String)

**cq:Wait**

**Description**

Wait.

**Definition**

[cq:Wait]  
  - workflowId (String) // not compulsory as this node will be already attached to the workflow node  
  - destNodeId (String)  
  - fromNodeId (String)

**cq:WorkflowStack**

**Description**

Workflow stack.

**Definition**

[cq:WorkflowStack]  
  - containeeInstanceId (String)  
  - parentInstanceId (String)  
  - nodeId (String)

**cq:ProcessStack**

**Description**

Process stack.

**Definition**

[cq:ProcessStack]  
  - workflowId (String) // not compulsory as this node will be already attached to the workflow node  
  - containerWorkflowModelId (String)  
  - containerWorkflowNodeId  
  - containerWorkflowEndNodeId // still needed (if name already defines that id)

**cq:WorkflowLauncher**

**Description**

Workflow launcher.

**Definition**

[cq:WorkflowLauncher]  
  - nodetype (String)  
  - glob (String)  
  - eventType (Long)  
  - description (String)  
  - condition (String)  
  - workflow (String)  
  - \* (undefined)  
  - \* (undefined) multiple