Search

Save as

project in (NXP, NXCM) AND resolution = Fixed AND fixVersion = "7.2" AND ("Impact type" = "API change" OR "Upgrade notes" is not EMPTY) ORDER BY component DESC, key DESC

Basic

1–31 of **31**

Columns

Key	Summary	Upgrade notes
NXP-16623	Allow to set a context variable, workflow variable or a workflow node variable to null	SetWorkflowVar, Context.SetVar and Workflow.SetNodeVariable automation operations accept null as value.
NXP-16605	Chained vocabulary: Incomplete selections are not allowed	Sample N-level select widget documentation has been updated accpordingly, custom templates should be reviewed: http://doc.nuxeo.com/x/2ABc, see diff at http://doc.nuxeo.com/pages/diffpagesbyversion.action?pageId=6029528&selectedPageVersions=20&selectedPageVersions=19
NXP-16551	Make JSF file upload pluggable	To plug in a new JSF uploader, use something like: <extension target="org.nuxeo.ecm.platform.ui.web.component.file.JSFBlobUploaderService" point="uploader"> <uploader id="myupload" order="20" class="my.BlobUploader" /> </extension> The class must implement org.nuxeo.ecm.platform.ui.web.component.file.JSFBlobUploader The default implementation (HTMLBlobUploader) is registered by default.
NXP-16339	REST Api: Refactor Marshaller (static methods -> components)	Broken compatibilities (details is below): With a way to get the previous behaviour: - DocumentModel to Json: versionLabel/lock removed - Json to DocumentModel: "" strings is recognized as empty and not null as previous behaviour - Document json loaded schema: header 'X-NXDocumentProperties' is replaced by header 'X-NXproperties' header or 'properties' http GET parameter but still supported - Document json Enrichers: header 'X-NXContext-Category' is replaced by header 'X-NXenrichers.document' header or 'enrichers.document' http GET parameter but still supported Without any way to get the previous behaviour: - REST API: Move org.nuxeo.ecm.restapi.jaxrs.io.directory.DirectoryEntry to nuxeo-platform-directory-api: org.nuxeo.ecm.directory.api.DirectoryEntry - Automation: Move org.nuxeo.ecm.automation.jaxrs.io.audit.LogEntryList to nuxeo-platform-audit-api: org.nuxeo.ecm.platform.audit.api.LogEntryList - Automation: ContentEnricherService removed and replaced by Marshaller implementing AbstractJsonEnricher<EntityType> : the behaviour is the same (existing enrichers still work) but customer's extension must be migrated. Plus enricher parameters are not yet supported. Was used for: - VocabularyEnricher: specific enricher wrote for nuxeo-platform-spreadsheet project - UserPermissionsContentEnricher Read|Write|Everything hard coded - cutom fields / ObjectResolver : Add a new method to resolver which returns the classes managed by the fetching: org.nuxeo.ecm.core.schema.types.resolver.ObjectResolver.getManagedClasses() - Directory custom field: DirectoryEntryResolver fetch org.nuxeo.ecm.directory.api.DirectoryEntry instead of DocumentModel (explicit typing) --- Existing JSON Marshalling: Old marshalling format is maintained except: # BROKEN COMPATIBILITY # Changes for DocumentModel -> JSON: - Document Version and Lock info are not anymore fetched by default

> force fetching using system properties nuxeo.document.json.fetch.heavy=true
> force fetching using HEADER X-NXfetch.document=versionLabel or X-NXfetch.document=lock
> force fetching using GET parameter fetch.document=versionLabel or fetch.document=lock
> force fetching using

Changes for JSON -> DocumentModel:
- inputs must be well typed: string for boolean or int are not accepted anymore
- empty string properties "" are recognized as empty "" insted of null like previous behaviour
- Old reader is usable by setting system property nuxeo.document.json.legacy=true
or by adding a http header to the request: X-NXDocumentJsonLegacy=true

---

New JSON Marshalling features:

- Contribute to marshalling
Write an class extending org.nuxeo.ecm.core.io.registry.Writer<EntityType> or org.nuxeo.ecm.core.io.registry.Reader<EntityType>
Use @Setup(mode, priority) to specifiy the instantiation mode of your marshaller (singleton, per thread, each use instantiation)
Use @Supports(MediaType) annotation to specify the format (MediaType.APPLICATION_JSON for Json)
!!! Unlike JAX-RS the full generic type is used to manage the activation of the marshaller: if you create a Writer<Map<String, DocumentModel>> marshaller, it will be enable only for JAX-RS web method returning objects implementing Map<String, DocumentModel> (JAX-RS just supports Map matching in this case). There's no need to override the accept method in this case.
Please prefer use abstract classes (this will facilitates the writing of your marshaller):
- org.nuxeo.ecm.core.io.marshallers.json.AbstractJsonWriter<EntityType> : optimized base for Java to JSON
- org.nuxeo.ecm.core.io.marshallers.json.ExtensibleEntityJsonWriter<EntityType> : extensible named entity Java to JSON
- org.nuxeo.ecm.core.io.marshallers.json.DefaultListJsonWriter<EntityType> : list of 'Json-Writable' objects
- org.nuxeo.ecm.core.io.marshallers.json.AbstractJsonReader<EntityType> : optimized base for JSON to Java
- org.nuxeo.ecm.core.io.marshallers.json.EntityJsonReader<EntityType> : named entity JSON reader
- org.nuxeo.ecm.core.io.marshallers.json.DefaultListJsonReader<EntityType> : list of 'Json-Readable' objects
Extension point example:

```
<component name="org.nuxeo.ecm.platform.usermanager.marshallers" version="1.0.0">
<documentation>
Core IO registered marshallers set.
</documentation>
<extension target="org.nuxeo.ecm.core.io.MarshallerRegistry" point="marshallers">
<register class="org.nuxeo.ecm.platform.usermanager.io.NuxeoPrincipalJsonWriter" enable="true" />
<register class="org.nuxeo.ecm.platform.usermanager.io.NuxeoPrincipalJsonReader" enable="true" />
<register class="org.nuxeo.ecm.platform.usermanager.io.NuxeoPrincipalListJsonWriter" enable="true" />
<register class="org.nuxeo.ecm.platform.usermanager.io.NuxeoPrincipalListJsonReader" enable="true" />
<register class="org.nuxeo.ecm.platform.usermanager.io.NuxeoGroupJsonWriter" enable="true" />
<register class="org.nuxeo.ecm.platform.usermanager.io.NuxeoGroupJsonReader" enable="true" />
<register class="org.nuxeo.ecm.platform.usermanager.io.NuxeoGroupListJsonWriter" enable="true" />
<register class="org.nuxeo.ecm.platform.usermanager.io.NuxeoGroupListJsonReader" enable="true" />
</extension>
</component>
```

- All Core IO marshallers are usable in WebEngine projects
You just have to add this MessabeBodyWriter/MessageBodyReader to your application: org.nuxeo.ecm.webengine.jaxrs.coreiodelegate.FullCoreIODelegate
You can customize the delegation using org.nuxeo.ecm.webengine.jaxrs.coreiodelegate.PartialCoreIODelegate
Or write your own delegation

Be careful: JAX-RS Marshallers have a greater priority than core io marshallers (if a DocumentModel MessageBodyWriter is present in your project, it will be used in priority)

- Marshaller overriding:
You can define your own core-io marshallers and override existing marshalling
To achieve that, create a core-io Reader/Writer with a higher priority (Nuxeo base marshallers priority is 2000 -> Priorities.REFERENCE):

```
@Setup(mode = Instantiations.SINGLETON, priority =
Priorities.OVERRIDE_REFERENCE)
public class MyCustomNuxeoPrincipalJsonWriter implements Writer<NuxeoPrincipal> {
...
}
```

Most of existing marshallers could be extended using enrichers (see below) or by extending the existing writer and implementing method 'extend'

```
@Setup(mode = Instantiations.SINGLETON, priority =
Priorities.OVERRIDE_REFERENCE)
public class MyCustomNuxeoPrincipalJsonWriter extends NuxeoPrincipalJsonWriter {
protected void extend(EntityType entity, JsonWriter jg) throws IOException {
...
}
}
```

=> this will add extra JSON info in exiting JSON and will preserve compatibility with existing API

- Properties loading:
Old header X-NXDocumentProperties is still supported
You may use alternative parameter properties=* or properties=dublincore,common or ? properties=dublincore&properties=common or header X-NXproperties=dublincore,common

- Enrichment:
Available for 'Json-Writable' entities: DocumentModel, ACL, DocumentType, Schema, Facet, Constraint, DocumentValidationReport, NuxeoPrincipal, NuxeoGroup, Directory, LogEntry
The corresponding marshaller supports Enrichers.
You have to write and register in core-io a class extendind AbstractJsonEnricher<EntityType> where EntityType is the original Marshalled type
This will add an aditionnal Json property 'contextParameters' which will contain all related enrichers.
Enrichers is not enable by default, you have to add a paremeter or header in your request:
> Parameter: enrichers.document=children,acls
> Header: X-NXenrichers.document=children,acls
This supports and will aggregate multiple parameters or headers: ? enrichers.document=children&enrichers.document=acls
This supports old header X-NXContext-Category
To enrich a NuxeoPrincipal with your own enricher 'userComputedInfos', you have to put the parameter enrichers.user=userComputedInfos

- Fetching:
When you defined some document's properties as a reference to an object using an ObjectResolver (contrib to component org.nuxeo.ecm.core.schema.ObjectResolverService, extension point 'resolvers'), you can fetch the corresponding object if the Json reader and Json writer are available (this is the case for DocumentModel, Directory, NuxeoPrincipal and NuxeoGroup properties).
To enable the fetching, use the parameter fetch.document=properties => this will fetch all properties
or with a specific property path : fetch.document=mySchema:users/user[0]/nuxeoPrincipalId => this will fetch the property nuxeoPrincipalId or the first user of the user list property users
or with a generic path : fetch.document=mySchema:user:users/user/nuxeoPrincipalId => this will fetch all property nuxeoPrincipalId
or with a parent path : fetch.document=mySchema:users => this will fetch all fetchable properties under users
!!! The property value is accepted in Json by the reader and the corresponding object too (you can use either the reference or the object to update your document)

Distribution Tomcat:

- Add new library commons-lang3 in ${distribution.dir}/lib"

Nuxeo pom.xml:

- Add dependency management for nuxeo-core-io test-jar

- Add dependency management for commons-lang3

Query API:

- Move most of object of the package nuxeo-platform-query-api:
org.nuxeo.ecm.platform.query.api in nuxeo-core-query project
SAME PACKAGE USED
org.nuxeo.ecm.platform.query.api.Aggregate
org.nuxeo.ecm.platform.query.api.AggregateDefinition
org.nuxeo.ecm.platform.query.api.AggregateRangeDateDefinition
org.nuxeo.ecm.platform.query.api.AggregateRangeDefinition
org.nuxeo.ecm.platform.query.api.Bucket
org.nuxeo.ecm.platform.query.api.PageProvider
org.nuxeo.ecm.platform.query.api.PageProviderChangedListener
org.nuxeo.ecm.platform.query.api.PageProviderDefinition
org.nuxeo.ecm.platform.query.api.PageSelection
org.nuxeo.ecm.platform.query.api.PageSelections
org.nuxeo.ecm.platform.query.api.PredicateDefinition
org.nuxeo.ecm.platform.query.api.PredicateFieldDefinition
org.nuxeo.ecm.platform.query.api.WhereClauseDefinition

- Move nuxeo-platform-search-api:
org.nuxeo.ecm.core.search.api.client.querymodel.Escaper in nuxeo-core-query project
SAME PACKAGE USED
org.nuxeo.ecm.core.search.api.client.querymodel.Escaper

Web Engine:

- Add exception handling for DocumentValidationException and mediatype 'application/json'

- Add maven and bundle dependency to nuxeo-core-io

- Add a converter from javax.servlet.http.HttpServletRequest to nuxeo-core-io RenderingContext (request parameters, attributes and headers)

- Add JAX-RS MessageBodyReader/MessageBodyWriter that delegates marshalling to nuxeo-core-io MarshallerRegistry:
org.nuxeo.ecm.webengine.jaxrs.coreiodelegate.FullCoreIODelegate
org.nuxeo.ecm.webengine.jaxrs.coreiodelegate.PartialCoreIODelegate

- Ugly copy of VirtualHostHelper: // TODO: refactor with
org.nuxeo.ecm.platform.web.common.vh.VirtualHostHelper

- Add "org.nuxeo.ecm.core.io" bundle to WebEngineFeature test's feature

---

Rest API v1:

- BROKEN COMPATIBILITY: Move
org.nuxeo.ecm.restapi.jaxrs.io.directory.DirectoryEntry to nuxeo-platform-directory-api:
org.nuxeo.ecm.directory.api.DirectoryEntry

Automation:

- BROKEN COMPATIBILITY: Move org.nuxeo.ecm.automation.jaxrs.io.audit.LogEntryList to nuxeo-platform-audit-api: org.nuxeo.ecm.platform.audit.api.LogEntryList

- Move 4 Pagination related objects to nuxeo-core-io project
SAME PACKAGE USED
org.nuxeo.ecm.automation.core.util.Paginable<T>
org.nuxeo.ecm.automation.core.util.PaginablePageProvider<T>

org.nuxeo.ecm.automation.core.util.PaginableDocumentModelList
org.nuxeo.ecm.automation.jaxrs.io.documents.PaginableDocumentModelListImpl

- contrib marshaller-contrib.xml : Remove
<writer>org.nuxeo.ecm.automation.jaxrs.io.documents.JsonDocumentWriter</writer>
<writer>org.nuxeo.ecm.automation.jaxrs.io.documents.JsonDocumentListWriter</writer>
<writer>org.nuxeo.ecm.automation.jaxrs.io.audit.LogEntryWriter</writer>
<writer>org.nuxeo.ecm.automation.jaxrs.io.audit.LogEntryListWriter</writer>

- contrib marshaller-contrib.xml : Add
<writer>org.nuxeo.ecm.webengine.jaxrs.coreiodelegate.FullCoreIODelegate</writer>

- BROKEN COMPATIBILITY: ContentEnricherService remove extension point - maintain feature - existing marshallers migrated
Remove the service - replace all enrichers by custom marshallers class extending:
org.nuxeo.ecm.core.io.marshallers.json.enrichers.AbstractJsonEnricher<EntityType>
(class used by marshallers extending org.nuxeo.ecm.core.io.marshallers.json.ExtensibleEntityJsonWriter<EntityType>)
New marshallers:
- nuxeo-core-io: org.nuxeo.ecm.core.io.marshallers.json.enrichers.ACLJsonEnricher
- nuxeo-core-io: org.nuxeo.ecm.core.io.marshallers.json.enrichers.BasePermissionsJsonEnricher
- nuxeo-core-io: org.nuxeo.ecm.core.io.marshallers.json.enrichers.BreadcrumbJsonEnricher
- nuxeo-core-io: org.nuxeo.ecm.core.io.marshallers.json.enrichers.ChildrenJsonEnricher
- nuxeo-core-io: org.nuxeo.ecm.core.io.marshallers.json.enrichers.ContextualParametersJsonEnricher
- nuxeo-platform-url-core: org.nuxeo.ecm.platform.url.io.DocumentUrlJsonEnricher
- nuxeo-platform-preview: org.nuxeo.ecm.platform.preview.io.PreviewJsonEnricher
- nuxeo-platform-ui-web: org.nuxeo.ecm.platform.ui.web.io.ThumbnailJsonEnricher
- nuxeo-platform-spreadsheet: org.nuxeo.ecm.platform.spreadsheet.DCVocabulariesJsonEnricher
New enrichers are enable by their projects using contrib to nuxeo-core-io MarshallerRegistry:

```
<?xml version="1.0"?>
<component name="org.nuxeo.ecm.platform.spreadsheet.marshallers" version="1.0.0">
<documentation>
Core IO registered marshallers set.
</documentation>
<extension target="org.nuxeo.ecm.core.io.MarshallerRegistry" point="marshallers">
<!-- vocabularies dublincore document enricher -->
<register class="org.nuxeo.ecm.platform.spreadsheet.DCVocabulariesJsonEnricher"
enable="true" />
</extension>
</component>
```

Core Extended fields (updates of initial ~~NXP-15975~~) :

- BROKEN COMPATIBILITY: DirectoryEntryResolver fetch org.nuxeo.ecm.directory.api.DirectoryEntry instead of DocumentModel (explicit typing)

- BROKEN COMPATIBILITY: Add a new method to resolver which returns the classes managed by the fetching:
org.nuxeo.ecm.core.schema.types.resolver.ObjectResolver.getManagedClasses()
+ implementation for UserManagerResolver (NuxeoPrincipal.class and NuxeoGroup.class), DocumentModelResolver (DocumentModel.class), DirectoryEntryResolver (DirectoryEntry.class)

- Handle exceptions on message error generation when there's no i18n properties file

- DocumentValidationReport constructor is now public

NXP-16260	REST API: provide way to get referenced objects	Fetching: When you defined some document's properties as a reference to an object using an ObjectResolver (contrib to component org.nuxeo.ecm.core.schema.ObjectResolverService, extension point 'resolvers'), you can fetch the corresponding object if the Json reader and Json writer are available (this is the case for DocumentModel, Directory, NuxeoPrincipal and NuxeoGroup properties).

		To enable the fetching: - use the parameter fetch.document=properties => this will fetch all properties - or with a specific property path : fetch.document=mySchema:users/user[0]/nuxeoPrincipalId => this will fetch the property nuxeoPrincipalId or the first user of the user list property users - or with a generic path : fetch.document=mySchema:user:users/user/nuxeoPrincipalId => this will fetch all property nuxeoPrincipalId - or with a parent path : fetch.document=mySchema:users => this will fetch all fetchable properties under users !!! The property value is accepted in Json by the reader and the corresponding object too (you can use either the reference or the object to update your document)
NXP-12478	Add workflow and task endpoint to REST API	Workflow can now be managed via a REST API. See new endpoints documentation on the api-playground (http://nuxeo.github.io/api-playground/#/resources): - workflow endpoint - workflowModel endpoint - task endpoint
NXP-16724	Add Drive beta update site URL property	Added org.nuxeo.drive.beta.update.site.url property nuxeo-drive marketplace package, default value=http://community.nuxeo.com/static/drive-tests/
NXP-16394	Drive crashes when updating a Picture	Deprecated: - FileSystemItemManager#getSession(String repositoryName, Principal principal) - FileSystemItemFactory#getDocumentByFileSystemId(String id, Principal principal)
NXP-16473	Remove spurious multiple subwidget ids computation	Ids are generated on the fly only when needed, making ids usually more simple. Computation is now only done once inside nxl:layoutRowWidget and nxl:subWidget tags iterations, unless attribute recomputeIds="true" is added explicitely to the tag.
NXP-3427	Use javascript to handle lists of elements in UI	Old ajaxified list widget can be put back as default by using runtime property nuxeo.jsf.listWidget.compatEnabled=true. This property defaults to false from 7.2 (defaults to true on 6.0 backport)
NXP-16070	Remove the 'Original' view of a Picture	The 'Original' view is not computed anymore. For old Picture documents, the 'Original' view is migrated to 'file:content' (if empty) when the server starts. The PictureViewListener listener has been deprecated, the event 'updatePictureView' is no more triggered.
NXP-16336	Use nuxeo-binary-metadata for handling exif and iptc metadata extraction	- IPTC schema has been removed from document type Picture - Widget "summary_picture_iptc" has been removed from document summary - Mistral engine is removed from metadata extraction of Nuxeo by default. Use Binary Metadata feature to extract metadata with custom engine. - EXIF mapping remains identical - Default contribution for metadata mapping: http://doc.nuxeo.com/display/NXDOC/Binary+Metadata#BinaryMetadata-DefaultContribution
NXP-16505	Reduce number of ACLR by factorizing access to all users - PostgreSQL	To take effect on an existing instance the ACLR table must be rebuilt, this can be done by calling the nx_rebuild_read_acls stored procedure at the database level.
NXP-16437	Fix Redis configuration template	Change default value for nuxeo.cache.type (from "default" to "redis") New property nuxeo.lock.manager set to "redis"
NXP-16586	Allow configuration of MongoDB database name	Property nuxeo.mongodb.dbname can be used to define the database name. The default is "nuxeo".
NXP-16361	Allow configuration of MongoDB replica sets, credentials and options	The <server> part of the MongoDB repository configuration (nuxeo.mongodb.server in nuxeo.conf) can now include a full URI containing credentials, replica sets, and options. See http://docs.mongodb.org/manual/reference/connection-string/ for the syntax.
NXP-16666	Make dc:nature reference to vocabulary "nature"	dc:nature is defined as a "nature" directory valid entry. It's also validated and fetchable on a document's json using fetch.document=dc:nature.
NXP-16616	Add a reference to dc:subjects to the l10nsubjects directory	dc:subjects are defined as a "l10nsubjects" directory valid entries. It's also validated and fetchable on a document's json using fetch.document=dc:subjects.
NXP-16615	Add user reference to dc:contributors field	dc:contributors are defined as valid users. It's also validated and fetchable on a document's json using fetch.document=dc:contributors.
NXP-16614	Add user reference to dc:lastContributor field	dc:lastContributor is defined as a valid user. It's also validated and fetchable on a document's json using fetch.document=dc:lastContributor.
NXP-16518	Add user reference constraint to dc:creator	dc:creator is defined as a valid user. It's also validated and fetchable on a document's json using fetch.document=dc:creator.
NXP-16517	Add Vocabulary constraint on dc:coverage	dc:coverage is defined as a "l10ncoverage" directory valid entry.

	property	It's also validated and fetchable on a document's json using fetch.document=dc:coverage.
NXP-16441	Make dirty field management more accurate	A document's property is now dirty only when the new value is different. The DublinCoreListener update the dc:modified, dc:contributors and dc:lastContributor on beforeDocumentModification only if the document is dirty (i.e. only if some value changed). The modified and contributors are still updated on lifecycle_transition_event even if the document is not dirty. There's a dirty management on array property items: ArrayProperty.isDirty(index). A document's snapshot version is created only if any field changed (doc is dirty). A method is available on property to force the dirty status even if the property value doesn't change on setValue (setForceDirty). It's used by the SimpleDocumentModel class to keep the previous behaviour on it. The DocumentValidationService validates only dirty properties if it's asked as usual - but also validates only dirty sub properties and dirty arrayproperty's items. Many test were updated (when a snapshot creation was expected without any change).
NXP-16409	Document validation > fix sub fields validation	SchemaManager#getField now resolves xpath, and added API SchemaManager#getField(parentField, subname)
NXP-16400	Fix constraints registration on sub sub list item property	Constraints can de added to XSD list item by defining a simpleType and associated restrictions.
NXP-16376	Cleanup Blob class hierarchy	A new Blobs utility class has been introduced, providing factory methods for Blob creation. It should be used in preference over the existing constructors. So instead of: Blob blob = new InputStreamBlob(request.getInputStream()); Blob blob = new FileBlob(file, mimeType); Blob blob = new StringBlob("foo"); Blob blob = new ByteArrayBlob(contentBytes, mimeType, encoding); One should use: Blob blob = Blobs.createBlob(request.getInputStream()); Blob blob = Blobs.createBlob(file, mimeType); Blob blob = Blobs.createBlob("foo"); Blob blob = Blobs.createBlob(contentBytes, mimeType, encoding); Note that Blobs.createBlobWithExtension is also available. StorageBlob has been renamed to BinaryBlob and must not be used outside of the storage implementations.
NXP-15970	Allow ACL-related queries from NXQL	Queries on ACLs can be done using the pseudo-list properties: ecm:acl/*/principal : the user or group ecm:acl/*/permission : the permission ecm:acl/*/grant : true/false (1/0) for grant/deny ecm:acl/*/name : the ACL name ecm:acl/*/pos : the position Like for all lists, the * should be suffixed by a correlation integer if several references to the same access control entry are done.
NXP-15895	Bridge CMISQL to Elasticsearch	To send the CMISQL queries to Elasticsearch, set the following property: org.nuxeo.cmis.elasticsearch=true
NXP-16677	Directory Entry JSON marshalling: allow to fetch parent and translation	The REST-endpoints, which return a directory entry, can handle the following parameters: - fetch.directoryEntry=modelFieldName : it tries to load the entry referenced by the modelFieldName value of this entry - usefull to load xvocabulary path: fetch.directoryEntry=parent - translate.directoryEntry=modelFieldName : it translates "in-place" the value of the specified model field. It's usefull for directories like "nature" which contains i18n keys
NXP-16479	Finalize Automation Scripting integration	Automation scripting module requires jdk8u25 at least.
NXP-16203	Global Automation Operation/Chain	96 Automation operations have been renamed (aliases have been set).

Renaming

Listing: http://doc.nuxeo.com/x/7gIz

1–31 of **31**