Nuxeo Issue Tracker

JQL Query: project = NXP AND status = Resolved AND fixVersion = "7.10" AND ("Impact type" is not EMPTY OR "Upgrade notes" is not EMPTY) ORDER BY component DESC, key DESC
Sorted by: Component/s descending, then Key descending
1–9 of **9** as at: **18/11/15 02:21**

Summary	Description
Refactor and finalize TransientStore	Main changes: ▪ We are no more manipulating a `StorageEntry` object to represent what is stored but simply a list of blobs and a map of parameters. Thus the new API: `void putParameters(String key, Map<String, Serializable> parameters);` `Map<String, Serializable> getParameters(String key);` `void putBlobs(String key, List<Blob> blobs);` `List<Blob> getBlobs(String key);` The purpose of these changes is to get closer to the Redis atomic commands to better manage concurrency and atomicity in the Redis implementation. Note that a `StorageEntry` class is still used as a POJO in the `SimpleTransientStore` in-memory implementation. ▪ The `CacheService` is now only used by the in-memory implementation. ▪ The `AbstractTransientStore` is now only in charge of storing / loading / garbaging the blobs on / from the file system. Everything that is related to entry parameters and blob metadata is handled by the implementers. This refactoring allows the Redis implementation to be cluster aware, see ~~NXP-18051~~ for implementation details.
Add an nxqlEscape automation function	In the platform's Web UI: ▪ I create a File manually with the title test"something ▪ After creation, I export the document as XML and see that the path contains the quote This is troublesome for NXQL queries, for instance in Studio, using the `SELECT * FROM Document WHERE ecm:path STARTSWITH '@{Document.path}'` query in the Fetch > Query operation => if the path is not escaped, this can lead to errors => if the path is escaped, the query does not provide any result With the new `Fn.nxqlEscape`, the solution, is to use `SELECT * FROM Document WHERE ecm:path STARTSWITH '@{Fn.nxqlEscape(Document.path)}'` – Fn.nxqlEscape can now be used in Automation MVEL expressions. Ex: `SELECT * FROM Document WHERE ecm:path STARTSWITH '@{Fn.nxqlEscape(Document.path)}'`
Upgrade Quartz to 2.x	Upgrade Quartz to 2.1.3 in order to try to improve performances.
Make DnD collector use a DocumentModel instead of a Map	The current implementation has some impediments: ▪ it's not possible to use complex fields ▪ it's not possible to use the default values defined in the schemas Using a document model to store the collected data will address these 2 problems. – Drag'n Drop Collector screens now make use of a document model instead of a map. This makes it possible to handle complex properties as well as default values in the UI that pops out when doing a drag'n drop.
Don't allow batch upload with a client-side generated id	The new upload API implemented by `BatchUploadObject` doesn't allow such thing. Whatever request is done using the `/api/v1/upload` endpoint the `batchId` is part of the resource itself, ex: `@POST` `@Path("{batchId}/{fileIdx}")` `public Response upload(@Context HttpServletRequest request, @PathParam(REQUEST_BATCH_ID) String batchId, @PathParam(REQUEST_FILE_IDX) String fileIdx) throws IOException` A 404 status code is returned if the batch matching the given id doesn't exist. Yet the old API, deprecated but maintained for backward compatibility, does allow such a client-side generated id passed as a request header, see `BatchResource`:

Summary	Description
	`` @Deprecated @POST @Path("/upload") public Object doPost(@Context HttpServletRequest request) throws IOException { ... String batchId = request.getHeader("X-Batch-Id"); ... } ``` This is how the old API now behaves: 1. If no batch id is provided, initialize a batch with a server-side generated id by calling `BatchManager#init()`. 2. If a batch id is provided: - If an existing batch matches this id use it. This is possible by making a first call to the new API: `/api/v1/upload` but in this case why not use the new API through the whole upload process? - If no batch matches this id: - If the `allowClientGeneratedBatchId` configuration property is false, return an HTTP 500 error with the following message "Cannot upload a file with a client-side generated batch id, please use new upload API or set configuration property allowClientGeneratedBatchId to true (not recommended)". - If the `allowClientGeneratedBatchId` configuration property is set to true, a batch will be initialized internally with this id by the `BatchManager`. With a warning in the logs: "Allowing to initialize upload batch with a client-side generated id since configuration property allowClientGeneratedBatchId is set to true but this is not recommended, please use new upload API instead". The `allowClientGeneratedBatchId` configuration property is not set by default (thus false) for the LTS 2015 to enforce security. This has consequences on the Nuxeo client code using the old batch upload API with a client-side generated id: - Drag and Drop, see ~~NXP-18031~~ - JS client and nuxeo-elements, see ~~NXJS-29~~ - CAP and Drive Scala bench + Drive Funkload bench, see ~~NXP-18033~~ - Nuxeo Drive, see ~~NXDRIVE-433~~ **Important note about Nuxeo Drive: starting from 7.10, the minimum compatible version of Nuxeo Drive will be the next released version, meaning the one following 2.0.911.** This version will include ~~NXDRIVE-433~~.
STARTSWITH operator in Elasticsearch should work as in VCS	At the moment the startswith operator return the root document that match the path while VCS only returns the children. This should be fixed. Also for other fields than ecm:path the STARTSWITH operator works like a java string startswith. So it does not work properly on hierarchycal vocabulary. Even if it requires an explicit mapping it should behave the same way as in VCS.
Use nuxeo.path.segment.maxsize property to generate document name	There are several places when we use some hardcoded values to limit the length of document name whereas we should use the adequate property
Handle custom messages on subfields	use a validation based on xpath instead of a validation based on field => in nuxeo, a field doesn't know "where it's defined", so giving a subfield, it's not possible to determine the field's full path and this way get the custom message defined
Make Elasticsearch audit and UID sequence indexes configurable	The Elasticsearch index names used for the UID sequencer and for audit logs are now configurable through the following properties in `nuxeo.conf`: ``` ## Name of the Elasticsearch index for audit logs audit.elasticsearch.indexName=${elasticsearch.indexName}-audit ## Name of the Elasticsearch index for the uid sequencer seqgen.elasticsearch.indexName=${elasticsearch.indexName}-uidgen ``` As a reminder, the audit Elasticsearch index is mostly used by `ESAuditBackend` and the UID sequencer index by `ESUIDSequencer`. **This allows to have several Nuxeo instances sharing the same Elasticsearch cluster.** We use the `elasticsearch.indexName` property (default value : `nuxeo`) as a prefix to ease configuration of multiple Nuxeo insatnces sharing an Elasticsearch cluster: changing the single `elasticsearch.indexName` property is enough to give a unique name to all the Elasticsearch indexes. Since the default value for `elasticsearch.indexName` is `nuxeo`, the resulting default values for the new properties are: ``` audit.elasticsearch.indexName=nuxeo-audit seqgen.elasticsearch.indexName=nuxeo-uidgen ```