****

**Nakala**

**Guide d’utilisation de l’application batch d’envoi de données**

|  |  |  |  |
| --- | --- | --- | --- |
| *Historique des révisions* | | | |
| *Version* | *Date* | *Auteur* | *Modifications* |
| V1 | 07/11/2013 | Thomas Francart | Version initiale |
| V2 | 27/01/2014 | Thomas Francart | Indication des créations de liens inverses |
| V3 | 01/02/2014 | Thomas Francart | Ajout de la procédure de lancement batch |
| V4 | 10/02/2014 | Thomas Francart | Modification pour tenir compte du nouveau schéma XSD |
| V5 | 28/02/2014 | Thomas Francart | Renommage de l’application en Nakala |

Sommaire

[Qu’est-ce que l’application batch d’envoi de données ? 3](#_Toc381369451)

[Comment installer l’application batch d’envoi de données ? 3](#_Toc381369452)

[Télécharger l’application 3](#_Toc381369453)

[Installer l’application 3](#_Toc381369454)

[Prérequis 3](#_Toc381369455)

[Installation 3](#_Toc381369456)

[Comment lancer l’application batch d’envoi de données ? 3](#_Toc381369457)

[Paramètres de lancement 3](#_Toc381369458)

[Lancement avec les scripts start.bat ou start.fr 4](#_Toc381369459)

[Lancement direct en java, avec le password dans un fichier 4](#_Toc381369460)

[Fonctionnement du batch 5](#_Toc381369461)

[Description du format des paquets d’entrée à envoyer 6](#_Toc381369462)

[La validation FACILE 9](#_Toc381369463)

[Création de collections à partir du batch 10](#_Toc381369464)

[Description du format des paquets de sortie 11](#_Toc381369465)

[Description du fichier de rapport de batch 12](#_Toc381369466)

# Qu’est-ce que l’application batch d’envoi de données ?

L’application batch d’envoi de données de Nakala, ou application « console », est une application ligne de commande en Java permettant d’envoyer à Nakala des paquets contenant le fichier de données d’une part et un fichier de métadonnées descriptive d’autre part.

# Comment installer l’application batch d’envoi de données ?

## Télécharger l’application

L’application batch est téléchargeable depuis l’interface web d’administration de Nakala, accessible avec vos identifiants Renater. Une fois dans l’interface web, naviguez dans l’onglet « Batch d’envoi de paquets », et cliquez sur le lien de téléchargement dans cet onglet.

## Installer l’application

### Prérequis

Pour que l’application batch puisse fonctionner :

* Java 5 ou plus récent doit être installé sur votre machine ;
* La commande « java » doit être dans votre PATH de lancement ; pour vérifier cela, ouvrez une fenêtre de commande (Linux ou Windows), et tapez la commande « java -version » ; la version de java installer sur votre machine doit s’afficher ;

### Installation

* Dézippez le zip téléchargé dans le répertoire de votre choix, et c’est tout !
* Une fois dézippé, le répertoire nakala-console doit contenir :
  + Les répertoires lib, cfg et input
  + Les fichiers start.bat, start.sh et nakala-console.jar
* Sous Linux, assurez-vous que « start.sh » est exécutable en exécutant « chmod 755 start.sh » ;
* Sous Linux, assurez-vous que le batch a les droits de lecture sur le répertoire d’entrée, et les droits d’écriture sur les répertoires de sortie et d’erreur ;

# Comment lancer l’application batch d’envoi de données ?

## Paramètres de lancement

La console prend en paramètres les informations suivantes

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom du paramètre** | **Nom du paramètre dans la commande java** | **Description** | **Valeur par défaut** |
| Aide | -h | Affiche un message d’aide |  |
| email | -email | l’email d’un compte de déposant dans Nakala associé à un password, avec lequel les données seront envoyées |  |
| Répertoire d’entrée | -inputFolder | Répertoire contenant les paquets à envoyer | input |
| Répertoire de sortie | -outputFolder | Répertoire dans lequel seront enregistrés les paquets de résultat, pour les paquets envoyés avec succès. | output |
| Répertoire d’erreur | -errorFolder | Répertoire dans lequel seront copiés les paquets du répertoire d’entrée pour lesquels une erreur est intervenue. Après le lancement du batch, si ce répertoire est vide, tous les paquets ont été correctement envoyés. | error |
| Validation avec FACILE | -facileValidation | Indique si on veut que les paquets envoyés avec la console fassent l’objet d’une validation par le service FACILE (voir ci-dessous). | false |
| Chemin vers le fichier contenant le password crypté | -passwordFile | Donne le chemin vers un fichier qui contient le mot de passe à utiliser. Si cette option n’est pas présente, le password est demandé interactivement au lancement. Le contenu du fichier de password doit être le mot de passe encodé avec sha1sum (voir plus bas) |  |

## Lancement avec les scripts start.bat ou start.fr

Les scripts start.bat ou start.sh demandent les paramètres de lancement à l’utilisateur de façon interactive :

* Répertoire d’entrée ; (appuyer directement sur Entrée pour laisser la valeur par défaut « input »)
* Répertoire de sortie ; (appuyer directement sur Entrée pour laisser la valeur par défaut « output »)
* Répertoire d’erreur ; (appuyer directement sur Entrée pour laisser la valeur par défaut « error »)
* Validation avec FACILE ; (appuyer directement sur Entrée pour laisser la valeur par défaut « NO »)
* Email ;
* Password ;

## Lancement direct en java, avec le password dans un fichier

Il est également possible de lancer directement la commande en java, en utilisant les paramètres indiqués dans le tableau ci-dessus. De plus, il est possible de passer en paramètre le password crypté dans un fichier.

Le synopsys de la commande est :

java -jar nakala-console.jar -email $EMAIL -inputFolder $INPUT\_FOLDER -outputFolder $OUTPUT\_FOLDER -errorFolder $ERROR\_FOLDER $FACILE –passwordFile $PASSWORD\_FILE

Le fichier de password doit contenir le password encodé en SHA1. Sous Linux, la commande suivante permet de générer ce fichier :

echo -n password\_ici | sha1sum | awk '{printf $1}' > password\_file.sha

Explications :

* echo –n password\_ici : remplacez “password\_ici” par votre mot de passe en clair (sans guillemets)
* sha1sum : commande d’encryptage en sha1
* awk '{printf $1}' : prend la première partie de ce que génère sha1sum, en enlevant le tiret final
* > password\_file.sha : enregistre le résultat dans le fichier “password\_file.sha”; c’est ce fichier qui doit être indiqué en parameter de la commande de lancement du batch;

# Fonctionnement du batch

Lorsqu’on le lance, le batch effectue les opérations suivantes :

* Il lit les paquets « zip » dans le répertoire d’entrée ; la lecture s’effectue par ordre alphabétique des noms de fichier ;
* Ces paquets sont envoyés au serveur Nakala;
* Le serveur renvoie une réponse « DONE » si les conditions suivantes sont remplies :
  + L’email et le password indiqués sont valides ;
  + La validation FACILE n’a pas été demandée, ou bien elle a été demandée et a renvoyé un code de retour valide ;
  + Les métadonnées du paquet sont valides (voir ci-dessous) ;
  + Le fichier et les métadonnées sont correctement enregistrés dans Nakala ;
* Le serveur renvoi une réponse « ERROR » si l’une des conditions ci-dessus n’est pas remplie ;
* Pour chaque paquet d’entrée, un fichier de réponse en XML est créé dans le répertoire de sortie ;
* A la fin du traitement, un rapport d’exécution en XML est créé dans le répertoire de sortie ;

Les fichiers dans le répertoire d’entrée ne sont ni modifiés, ni supprimés par l’application.

# Description du format des paquets d’entrée à envoyer

Les fichiers dans le répertoire d’entrée doivent être des fichiers ZIP (extension \*.zip).

Chaque zip doit contenir :

* Un fichier de données (image, son, vidéo, texte, texte, etc.). L’extension du fichier n’importe pas si la validation FACILE n’est pas requise ;
* Un fichier XML décrivant les métadonnées du fichier ;

Les noms des fichiers de données et de métadonnées n’ont pas d’importance. Ces deux fichiers doivent se trouver à la racine du zip, ou bien dans un sous-répertoire à l’intérieur du zip. Si vous n’êtes pas sûrs de la structure du zip, utilisez un utilitaire zip tel que 7Zip ou le gestionnaire d’archives pour visualiser son contenu.

Le fichier de métadonnées doit être un fichier XML, obligatoirement avec l’extension \*.xml et doit être structuré de cette façon :

* Il doit commencer par une racine « nkl:Data » et doit déclarer l’espace de noms « nkl » comme [http://nakala.fr/schema#](http://nakala.fr/schema). Voici une entête typique d’un fichier :

<nkl:Data xmlns:nkl="http://nakala.fr/schema#"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:dcterms="http://purl.org/dc/terms/"

xsi:schemaLocation="http://purl.org/dc/terms/ http://dublincore.org/schemas/xmls/qdc/2008/02/11/dcterms.xsd">

< !—snip -->

</nkl:Data>

* Il doit ensuite contenir 4 métadonnées Dublin Core obligatoires : title, creator, type, created, dans cet ordre :

<dcterms:title>Venezuela : Estado Portuguesa : Guanare : Esquina Calle 13 et Carrera 5 : Vue d'une rue bordée d'habitations collectives et d'une friche urbaine</dcterms:title>

<dcterms:creator>Pouyllau, Michel</dcterms:creator>

<dcterms:type>Image</dcterms:type>

<dcterms:created>1983</dcterms:created>

* Il peut ensuite contenir n’importe quelles métadonnées génériques exprimées à l’aide du vocabulaire Dublin Core Terms (<http://dublincore.org/documents/dcmi-terms/>). On se reportera à la documentation du vocabulaire Dublin Core Terms pour une liste et une description de ces métadonnées. Toutes ces balises doivent contenir une valeur textuelle de métadonnée à l’intérieur de la balise, par exemple :

<dcterms:subject>habitat traditionnel</dcterms:subject>

<dcterms:subject>friche urbaine</dcterms:subject>

<dcterms:coverage>Venezuela, Bolivarian Republic Of, Guanare</dcterms:coverage>

<dcterms:spatial xsi:type="dcterms:Point">east=-69.748947; north=9.04549;name=Guanare</dcterms:spatial>

<dcterms:created>1983</dcterms:created>

<dcterms:type>Image</dcterms:type>

<dcterms:rights>Michel Pouyllau</dcterms:rights>

<dcterms:source>Centre de Documentation REGARDS</dcterms:source>

* *Note avancée* : Certaines de ces métadonnées Dublin Core Terms peuvent être des références à des URIs, comme par exemple :

<dcterms:licence xsi:type="dcterms:URI">http://creativecommons.org/licenses/by-nc-sa/2.0/fr/</dcterms:licence>

Ces valeurs seront alors interprétées comme des « object properties » dans le RDF de Nakala, et non des litéraux.

* Ensuite, il doit contenir dans cet ordre des métadonnées spécifiques à Nakala, qui appartiennent à l’espace de nom [http://nakala.fr/schema#](http://nakala.fr/schema). Ces métadonnées spécifiques sont décrites dans le tableau ci-dessous. Toutes ces métadonnées sont facultatives, sauf « nkl:dataFormat » seulement si on demande la validation FACILE.

|  |  |
| --- | --- |
| **Métadonnée** | **Description** |
| nkl:inCollection | Indique l’appartenance de la donnée envoyée à une collection dans Nakala. La balise doit contenir l’identifiant handle de la collection. Cet identifiant doit exister comme un identifiant de collection, sinon l’envoi du paquet générera une erreur. Cette balise peut être répétée plusieurs fois pour des données qui doivent appartenir à plusieurs collections.  Exemples :  <nkl:inCollection>11280/a19a5e26</nkl:inCollection> |
| nkl:relation | Indique une relation entre la donnée envoyée et une autre donnée dans Nakala.   * Le type de la relation doit obligatoirement être indiqué dans l’attribut « type ». Seule l’une de ces valeurs est autorisée comme valeur de l’attribut type :   + <http://purl.org/dc/terms/isVersionOf>   + <http://purl.org/dc/terms/hasFormat>   + <http://purl.org/dc/terms/isFormatOf>   + <http://purl.org/dc/terms/hasVersion>   + <http://purl.org/dc/terms/isVersionOf>   + <http://purl.org/dc/terms/replaces>   + <http://purl.org/dc/terms/isReplacedBy>   + <http://purl.org/dc/terms/hasPart>   + <http://purl.org/dc/terms/isPartOf>   + <http://purl.org/dc/terms/isReferencedBy>   + <http://purl.org/dc/terms/references>   + <http://purl.org/dc/terms/isRequiredBy>   + <http://purl.org/dc/terms/requires>   + <http://purl.org/dc/terms/relation>   Si l’attribut “type” contient une autre valeur, l’envoi du paquet générera une erreur. La verification ne tient cependant pas compte des majuscules/minuscules.   * La balise doit contenir l’identifiant handle de l’autre donnée que l’on souhaite référencer dans Nakala. Cet identifiant doit exister dans la base Nakala. Si celui-ci n’existe pas, l’envoi du paquet générera une erreur. * Lorsqu’une relation est indiquée, Nakala créera automatiquement la relation inverse (si « A dcterms:requires B » est créé, Nakala insérera automatiquement « B dcterms:isRequiredBy A »).   Cette balise peut être répétée plusieurs fois pour indiquer plusieurs relations.  Exemples :  <nkl:relation type="http://purl.org/dc/terms/isVersionOf">11280/bf86bec2</nkl:relation>  <nkl:relation type="http://purl.org/dc/terms/isrequiredby">11280/bf86bec2</nkl:relation> |
| nkl:accessEmail | Dans le cas de données protégées en lecture, indique l’email d’un utilisateur RENATER qui sera autorisé à accéder à la donnée. Cette balise peut être répétée plusieurs fois pour indiquer plusieurs adresses emails.  Les données pour lesquelles cette métadonnée est absente sont publiques.  Exemples :  <nkl:accessEmail>charles@cnrs.fr</nkl:accessEmail>  <nkl:accessEmail>mon.email@cnrs.fr</nkl:accessEmail> |
| nkl:dataFormat | Indique le format de la donnée pour la validation avec FACILE. Cette balise doit contenir l’une des valeurs listée à <http://facile.cines.fr/?media=XML>.  Cette balise est obligatoire si la validation FACILE est demandée.  Cette balise ne peut pas être répétée plusieurs fois.  Note : cette balise doit contenir obligatoirement l’une des valeurs listée à <http://facile.cines.fr/?media=XML>. Au moment de l’écriture de cette documentation, cette liste est la suivante, mais il est conseillé de se reporter à la documentation en ligne de FACILE pour la liste la plus à jour :   * AAC/AAC * AIFF/PCM * XDMF * FLAC/FLAC * GeoTIFF * GIF * HDF\5 * HTML * JPEG * JP2 * MKV/AVC/FLAC * MPEG-4/AVC/AAC * MPEG-4/AVC * odt * OGG/Theora/Vorbis * PDF * PNG * SIARD * SVG * TIFF * TXT/ASCII * TXT/UTF-8 * WAV/PCM * XML   Notez que le format de la donnée peut aussi être indiqué de manière libre dans l’attribut dcterms:format. |

L’ordre de ces balises est important et elles doivent apparaitre dans l’ordre donné dans ce tableau.

Voici un exemple complet de fichier de métadonnées d’entrée :

<nkl:Data xmlns:nkl="http://nakala.fr/schema#"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:dcterms="http://purl.org/dc/terms/"

xsi:schemaLocation="http://purl.org/dc/terms/ http://dublincore.org/schemas/xmls/qdc/2008/02/11/dcterms.xsd">

<dcterms:title>Venezuela : Estado Portuguesa : Guanare : Esquina Calle 13 et Carrera 5 : Vue d'une rue bordée d'habitations collectives et d'une friche urbaine</dcterms:title>

<dcterms:creator>Pouyllau, Michel</dcterms:creator>

<dcterms:type>Image</dcterms:type>

<dcterms:created>1983</dcterms:created>

<dcterms:identifier>http://medihal.archives-ouvertes.fr/medihal-00668270</dcterms:identifier>

<dcterms:identifier>http://medihal.archives-ouvertes.fr/docs/00/66/82/70/IMG/REGARDS\_Venezuela\_DC065.jpg</dcterms:identifier>

<dcterms:subject>[SHS:GEO] Humanities and Social Sciences/Geography</dcterms:subject>

<dcterms:coverage>Venezuela, Bolivarian Republic Of, Guanare</dcterms:coverage>

<dcterms:spatial xsi:type="dcterms:Point">east=-69.748947; north=9.04549;name=Guanare</dcterms:spatial>

<dcterms:rights>Michel Pouyllau</dcterms:rights>

<dcterms:licence xsi:type="dcterms:URI">http://creativecommons.org/licenses/by-nc-sa/2.0/fr/</dcterms:licence>

<dcterms:source>Centre de Documentation REGARDS</dcterms:source>

<nkl:inCollection>11280/a19a5e26</nkl:inCollection>

<nkl:relation type="http://purl.org/dc/terms/isVersionOf">11280/bf86bec2</nkl:relation>

<nkl:relation type="http://purl.org/dc/terms/isrequiredby">11280/bf86bec2</nkl:relation>

</nkl:Data>

# La validation FACILE

Si la validation FACILE a été demandée au moment du lancement du batch, Nakala appellera le service de validation FACILE à l’adresse <http://facile.cines.fr/?media=XML>.

Pour cela, la métadonnée « nkl:dataFormat » doit obligatoirement être présente dans les fichiers de métadonnées des paquets. Si la validation FACILE est demandée mais que cette métadonnée est absente du paquet, l’envoi du paquet générera une erreur.

# Création de collections à partir du batch

Le batch permet d’envoyer des données mais également de créer des collections dans la hiérarchie des collections. Pour cela, il faut :

* Créer un zip ne contenant pas de fichier de données, mais uniquement un fichier de métadonnées XML ;
* Utiliser la balise racine « nkl:Collection » à la place de « nkl:Data » ;

La balise nkl:Collection peut contenir :

* Une balise dcterms:title obligatoire
* Toutes les balises Dublin Core Terms comme pour les données ;
* Uniquement la balise « nkl:inCollection » de l’espace de nom Nakala. Les autres balises « nkl:relation », « nkl:accessEmail », etc. ne sont pas autorisées. Cette balise « nkl:inCollection » sert à indiquer la collection parente de la collection envoyée. Elle doit contenir pour cela l’identifiant handle de la collection parente. Si cette balise n’est pas présente, la collection sera créée directement sous la collection racine du compte email utilisé pour lancer le batch.

Exemple de fichier pour créer une Collection par le batch :

<nkl:Data xmlns:nkl="http://nakala.fr/schema#"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:dcterms="http://purl.org/dc/terms/"

xsi:schemaLocation="http://purl.org/dc/terms/ http://dublincore.org/schemas/xmls/qdc/2008/02/11/dcterms.xsd">

<dcterms:title>Le titre de ma collection</dcterms:title>

<nkl:inCollection>11280/a19a5e26</nkl:inCollection>

</nkl:Data>

*Note avancée* : pour les collections, le contenu de la balise dcterms :title sera copiée automatiquement par Nakala dans un attribut « skos :prefLabel ».

Lorsque la collection est créée par le batch, elle est immédiatement visible dans l’interface web de gestion des collections de Nakala.

# Description du format des paquets de sortie

Chaque fichier présent dans le répertoire d’entrée donne lieu à un fichier de résultat dans le répertoire de sortie, au format XML. Ce fichier de résultat est une copie du fichier de métadonnées du paquet, enrichi avec des métadonnées spécifiques à Nakala, décrites dans le tableau ci-dessous :

|  |  |
| --- | --- |
| **Nom de la métadonnée Nakala** | **Description** |
| nkl:identifier | Contient l’identifiant handle assigné à la donnée stockée dans Nakala. C’est cet identifiant qu’il faut utiliser pour accéder à la donnée en ligne à l’adresse [[URI\_Nakala]/data/<handle](http://silo.huma-num.fr/silo/data/%3chandle)>, et pour créer des relations vers cette donnée dans les balises « nkl:relation » des données envoyées. |
| nkl:checksum | Un checksum du fichier de données calculée par Nakala |
| nkl:fileName | Le nom du fichier de données qui était présent dans le zip |

Exemple de fichier de sortie :

<?xml version="1.0" encoding="UTF-8" standalone="no"?><nkl:Data xmlns:nkl="http://nakala.fr/schema#" xmlns:dcterms="http://purl.org/dc/terms/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://purl.org/dc/terms/ http://dublincore.org/schemas/xmls/qdc/2008/02/11/dcterms.xsd">

<dcterms:identifier>http://medihal.archives-ouvertes.fr/medihal-00668270</dcterms:identifier>

<dcterms:identifier>http://medihal.archives-ouvertes.fr/docs/00/66/82/70/IMG/REGARDS\_Venezuela\_DC065.jpg</dcterms:identifier>

<dcterms:title>Venezuela : Estado Portuguesa : Guanare : Esquina Calle 13 et

Carrera 5 : Vue d'une rue bordée d'habitations collectives et d'une friche

urbaine</dcterms:title>

<dcterms:creator>Pouyllau, Michel</dcterms:creator>

< !—snip : toutes les autres métadonnées du fichier d’entrée -->

<nkl:identifier>11280/17d7f0f1</nkl:identifier>

<nkl:checksum>7f449e3096b191379211a99e7411577bcc9f8a4b</nkl:checksum>

<nkl:fileName>test.file</nkl:fileName>

</nkl:Data>

## Description du fichier de rapport de batch

A la fin de chaque lancement, l’application batch génère un fichier de rapport en XML dans le repertoire de sortie. Ce fichier est nommé “report\_<timestamp>.xml”

Ce fichier donner les informations suivantes dans un format XML :

* Date de début et de fin du batch
* Nombre de fichiers traités
* Nombre de fichiers valides
* Nombre de fichiers invalids

Et pour chaque fichier :

* Son nom
* Sa date de début de traitement et de fin de traitement
* Son status DONE ou ERROR

Si c’est une erreur, un message d’erreur donnant la cause de l’erreur

Exemple de fichier de rapport :

<report\_data>

<date\_started>2013-11-06T19:06:51.467+01:00</date\_started>

<date\_ended>2013-11-06T19:10:13.051+01:00</date\_ended>

<nb\_files\_total>7</nb\_files\_total>

<nb\_files\_valid>5</nb\_files\_valid>

<nb\_files\_invalid>2</nb\_files\_invalid>

<batch\_status>DONE</batch\_status>

<files\_treated>

<file>

<file\_name>01-dctermsCreation</file\_name>

<file\_status>DONE</file\_status>

<date\_started>2013-11-06T19:06:51.473+01:00</date\_started>

<date\_ended>2013-11-06T19:07:56.989+01:00</date\_ended>

</file>

<file>

<file\_name>02-URI-translation</file\_name>

<file\_status>DONE</file\_status>

<date\_started>2013-11-06T19:07:56.996+01:00</date\_started>

<date\_ended>2013-11-06T19:08:30.559+01:00</date\_ended>

</file>

<file>

<file\_name>03-accessEmails</file\_name>

<file\_status>DONE</file\_status>

<date\_started>2013-11-06T19:08:30.560+01:00</date\_started>

<date\_ended>2013-11-06T19:09:04.090+01:00</date\_ended>

</file>

<file>

<file\_name>04-collection-without-inCollection</file\_name>

<file\_status>DONE</file\_status>

<date\_started>2013-11-06T19:09:04.090+01:00</date\_started>

<date\_ended>2013-11-06T19:09:37.275+01:00</date\_ended>

</file>

<file>

<file\_name>05-error-unknownRootNode</file\_name>

<file\_status>FAILED</file\_status>

<date\_started>2013-11-06T19:09:37.275+01:00</date\_started>

<date\_ended>2013-11-06T19:09:37.520+01:00</date\_ended>

<file\_status\_desc>Invalid xml input :: Metadata file is not well formatted.</file\_status\_desc>

</file>

< !-- etc. -->

</report\_data>