****

**Nakala**

**Documentation de l’API REST**

|  |  |  |  |
| --- | --- | --- | --- |
| *Historique des révisions* | | | |
| *Version* | *Date* | *Auteur* | *Modifications* |
| V1 | 14/02/2017 | Thomas Francart | Version initiale |
|  |  |  |  |
|  |  |  |  |

Sommaire

[Qu’est-ce que l’API REST Nakala ? 4](#_Toc476825310)

[Qu’est-ce que l’API REST permet de faire ? 4](#_Toc476825311)

[Manipulation des données : service …/api/v1/data 4](#_Toc476825312)

[Manipulation des collections : service …/api/v1/collection 4](#_Toc476825313)

[Gestion de la clé d’API et des filtres IP 4](#_Toc476825314)

[Comment (re-)générer la clé d’API d’un utilisateur ? 4](#_Toc476825315)

[Gérer les filtres IP 5](#_Toc476825316)

[Les formats des paquets utilisés par l’API REST 5](#_Toc476825317)

[Comment tester l’API REST ? 6](#_Toc476825318)

[Avec la commande curl 6](#_Toc476825319)

[Avec l’utilitaire Postman 6](#_Toc476825320)

[Appels HTTPS ou HTTP 6](#_Toc476825321)

[Format JSON de retour de l’API 6](#_Toc476825322)

[Paramètres d’authentification communs à tous les appels 7](#_Toc476825323)

[Description de l’API « data » 7](#_Toc476825324)

[POST /api/v1/data : créer une donnée 7](#_Toc476825325)

[Description 7](#_Toc476825326)

[Paramètres 7](#_Toc476825327)

[Exemple 8](#_Toc476825328)

[Retour 8](#_Toc476825329)

[PUT /api/v1/data/<handle> : modifier une donnée 8](#_Toc476825330)

[Description 8](#_Toc476825331)

[Paramètres 8](#_Toc476825332)

[Exemple 9](#_Toc476825333)

[Retour 9](#_Toc476825334)

[DELETE /api/v1/data/<handle> : supprimer une donnée 10](#_Toc476825335)

[Description 10](#_Toc476825336)

[Paramètres 10](#_Toc476825337)

[Exemple 10](#_Toc476825338)

[Retour 10](#_Toc476825339)

[GET /api/v1/data/<handle> : accéder à une donnée protégée 10](#_Toc476825340)

[Description 10](#_Toc476825341)

[Exemple 11](#_Toc476825342)

[Retour 11](#_Toc476825343)

[Description de l’API « collection » 11](#_Toc476825344)

[POST /api/v1/collection : créer une collection 11](#_Toc476825345)

[Description 11](#_Toc476825346)

[Paramètres 11](#_Toc476825347)

[Exemple d’appel 12](#_Toc476825348)

[Retour 12](#_Toc476825349)

[PUT /api/v1/collection/<handle> : modifier une collection 12](#_Toc476825350)

[Description 12](#_Toc476825351)

[Paramètres 12](#_Toc476825352)

[Exemple d’appel 12](#_Toc476825353)

[Retour 13](#_Toc476825354)

[DELETE /api/v1/collection/<handle> : supprimer une collection 13](#_Toc476825355)

[Description 13](#_Toc476825356)

[Paramètres 13](#_Toc476825357)

[Exemple d’appel 13](#_Toc476825358)

[Retour 13](#_Toc476825359)

# Qu’est-ce que l’API REST Nakala ?

L’API REST Nakala est un ensemble de fonctions qu’une application peut utiliser pour interagir avec la plateforme Nakala. Ces fonctions sont exposées sous la forme d’appels d’URLs en HTTP, qui prennent en paramètre une clé d’API permettant à l’application de s’authentifier avec un compte Nakala. Ces appels HTTP renvoient des réponses formattées en JSON.

Le batch d’envoi de données Nakala utilise l’API REST Nakala pour envoyer ses données.

Voici un exemple d’appel de l’API REST avec la commande « curl », qui permet d’envoyer un paquet de données à Nakala :

curl -X POST --header "Content-Type: application/octet-stream" --data-binary @the\_package.zip "https://www.nakala.fr/nakala/api/v1/data?email=nicolas.larrousse@huma-num.fr&key=28c34986-af44-4e92-8d78-8125267653c7"

# Qu’est-ce que l’API REST permet de faire ?

L’API Nakala permet de manipuler les données et les collections dans Nakala.

## Manipulation des données : service …/api/v1/data

* POST : Créer une nouvelle donnée à partir d’un paquet ;
* PUT : Mettre à jour un fichier de données et/ou les métadonnées descriptives de ce fichier ;
* DELETE : Supprimer une donnée ;
* GET : Accéder à une donnée protégée ;

## Manipulation des collections : service …/api/v1/collection

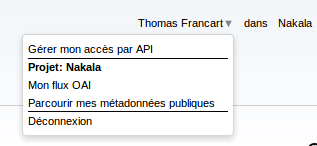
* POST : Créer une nouvelle collection à partir d’un paquet ;
* PUT : Mettre à jour une collection existante ;
* DELETE : Supprimer une collection ;

# Gestion de la clé d’API et des filtres IP

## Comment (re-)générer la clé d’API d’un utilisateur ?

Par défaut, un nouveau compte dans Nakala ne possède pas de clé d’API. L’administrateur Nakala ou vous-même pouvez générer une nouvelle clé d’API.

Pour générer la clé d’API associée à votre compte, connectez-vous dans le backoffice Nakala et cliquez sur votre nom d’utilisateur en haut à droite :



Cliquez sur « Gérer mon accès par API » ; le dialogue de gestion des accès par API s’affiche :



Cliquez sur « Générer une nouvelle clé ».

Si votre compte utilisateur a déjà une clé d’API, pour des questions de sécurité vous pouvez la supprimer ou en regénérer une nouvelle si vous pensez qu’elle a été compromise et est utilisée de façon malveillante.

## Gérer les filtres IP

Il est possible d’associer votre clé d’API à une ou plusieurs adresses IP. Dans ce cas, seuls les appels provenant d’une de ces adresses IP avec votre clé d’API seront autorisés. Les autres seront rejetés. Une clé d’API sans filtre IP peut être utilisée depuis n’importe quelle adresse IP.

# Les formats des paquets utilisés par l’API REST

L’API REST s’appuie sur le même format de paquets ZIP et les mêmes descripteurs XML que ceux utilisés par le batch d’envoi de données Nakala. On se reportera à la documentation du batch pour la description de ce format de paquet et des formats des fichiers descripteur XML.

# Comment tester l’API REST ?

## Avec la commande curl

Le plus simple est d’utiliser l’utilitaire en ligne de commande « curl ».

1. Créez un paquet de données zip suivant le format des paquets Nakala, qui sera envoyé à Nakala par l’API pour créer une nouvelle donnée ; appelez ce paquet « paquetTestApi.zip » ;
2. Si votre compte n’a pas de clé d’API, créez-en une nouvelle dans le back-office ;
3. Entrez la commande suivante :

curl -X POST --header "Content-Type: application/octet-stream" -data-binary @paquetTestApi.zip "https://www.nakala.fr/nakala/api/v1/data

?email=votre.adresse.email@nakala.fr&key=votre.cle.d’API"

A noter : le parameter de Content-Type est obligatoire pour que curl ne positionne pas automatiquement le header “multipart/form-data” ou “application/x-www-form-urlencode”

Vous devriez obtenir un résultat similaire à celui-ci :

{"handleId":"11280/ed1161c8","success":true,"metadata":"...","message":"Donnée créée avec succès : 11280/ed1161c8"}

## Avec l’utilitaire Postman

L’utilitaire Postman permet d’écrire facilement des appels d’URL, de visualiser et tester les résultats, de sauvegarder les requêtes, etc. Il peut être téléchargé à <https://www.getpostman.com/>

## Appels HTTPS ou HTTP

Les exemples donnés ici appellent l’API avec le protocole https. C’est la méthode recommandée pour appeler l’API.

Il est possible d’appeler l’API sur le protocole http, mais cela déclenche des redirections vers l’adresse https correspondante. Avec curl, pour gérer correctement ces redirections, il faut ajouter les paramètres suivants :

* -L pour suivre les redirections ;
* --post302 pour conserver le même verbe http lors de la redirection ;

# Format JSON de retour de l’API

Les propriétés du JSON retourné par l’API sont les suivantes :

|  |  |  |
| --- | --- | --- |
| **Propriété** | **Description** | **Valeurs** |
| success | Indique si la requête a été traitée avec succès | true ou false |
| message | Un message en français qui donne des détails sur le résultat de l’appel, ou l’erreur rencontrée. | Une chaine de caractère explicative. |
| handleId | Le handle de la donnée créée, pour les appels de création. | Un handle |
| metadata | Le descripteur XML ou CSV, enrichi avec les métadonnées de traitement Nakala. | Le descripteur XML ou CSV |

# Paramètres d’authentification communs à tous les appels

Les appels à l’API REST de Nakala doivent tous être authentifiés à l’aide de ces paramètres

|  |  |  |  |
| --- | --- | --- | --- |
| **Paramètre** | **Obligatoire** | **Description** | **Valeurs** |
| email | oui | l’email de l’utilisateur Nakala qui effectue la requête. | Un email |
| key | oui | La clé d’API associée à l’email | La clé, non encodée. |
| project | non | L’identifiant du projet dans lequel on souhaite que les appels soient effectués. Si l’email n’a accès qu’à un seul projet, ce paramètre peut être omis et c’est ce projet qui sera utilisé. Sinon, ce paramètre doit être indiqué. | Le handle d’un projet, par exemple « 11280/77c7777e » |

# Description de l’API « data »

## POST /api/v1/data : créer une donnée

### Description

Créé une nouvelle donnée dans Nakala, et retourne le nouveau handle de la donnée ainsi créé dans le message de retour.

### Paramètres

|  |  |  |  |
| --- | --- | --- | --- |
| **Paramètre** | **Obligatoire** | **Description** | **Valeurs** |
| *Paramètres d’authentification, voir la section «  »* | | | |
| facile | non | Si « true », indique qu’on veut que le paquet soit validé par Facile | true / false |
| Le paquet zip | oui | Le paquet ZIP passé dans le body de la requête ; attention que le « Content-Type » de la requête ne soit pas positionné automatiquement à “multipart/form-data” ou “application/x-www-form-urlencode”; |  |

### Exemple

curl -H "Content-Type: application/octet-stream" -X POST --data-binary @moine-petit.zip "https://www.nakala.fr/nakala/api/v1/data?email=thomas.francart@sparna.fr&key=28c34986-af44-4e92-8d78-81252696b3c7"

### Retour

{

"success":true,

"handleId":[HERE PUT THE NEW HANDLE],

"metadata":"<?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\">...\n\n<nkl:identifier>11280/d2ae4a92</nkl:identifier><nkl:checksum>dbb7ecc01bb94aba881075bd3fb92b3b2f738f0b</nkl:checksum><nkl:fileName>funny-like-hand-Schrodinger.jpg</nkl:fileName></nkl:Data>",

"message":"Donnée créée avec succès : 11280/15c2235e"

}

## PUT /api/v1/data/<handle> : modifier une donnée

### Description

Remplace le fichier de donnée et/ou les métadonnées d’une donnée déjà présente dans Nakala. Si les métadonnées sont remplacées, c’est un remplacement intégral : toutes les métadonnées existantes sont supprimées, et toutes les nouvelles métadonnées envoyée sont insérées.

### Paramètres

|  |  |  |  |
| --- | --- | --- | --- |
| **Paramètre** | **Obligatoire** | **Description** | **Valeurs** |
| *Paramètres d’authentification, voir la section «  »* | | | |
| Handle de la donnée | oui | Le handle de la donnée à modifier doit suivre la racine « …/api/v1/data/… », par exemple « …/api/v1/data/11280/abcdef ». | Le handle de la donnée à modifier. |
| facile | non | Si « true », indique qu’on veut que le paquet soit validé par Facile. Cela n’est possible que si le paquet contient la donnée ET le descripteur de métadonnées. | true / false |
| Le paquet zip | oui | Le paquet ZIP passé dans le body de la requête ; attention que le « Content-Type » de la requête ne soit pas positionné automatiquement à “multipart/form-data” ou “application/x-www-form-urlencode”;  Le paquet peut contenir uniquement le descripteur de métadonnées ou uniquement le fichier de données selon ce que l’on veut remplacer :   * Si le paquet contient les métadonnées et le fichier de donnée, les 2 sont remplacés ; * Si le paquet ne contient que les métadonnées, seules les métadonnées sont remplacées; * Si le paquet ne contient que le fichier de data, seul le fichier de data est remplacé; |  |

### Exemple

curl -H "Content-Type: application/octet-stream" -X PUT --data-binary @moine-petit.zip "https://www.nakala.fr/nakala/api/v1/data/11280/ d2ae4a92?email=thomas.francart@sparna.fr&key=28c34986-af44-4e92-8d78-81252696b3c7"

### Retour

En cas de mise à jour des données+métadonnées :

{

"success":true,

"handleId":"11280/15c2235e",

"metadata":"<nkl:Data><!-- metadonnees -->\n\n<nkl:identifier>11280/d2ae4a92</nkl:identifier><nkl:checksum>dbb7ecc01bb94aba881075bd3fb92b3b2f738f0b</nkl:checksum><nkl:fileName>nouvea-nom-fichier-du-fichier-de-data</nkl:fileName></nkl:Data>",

"message":"donnée et métadonnées mises à jour avec succès : 11280/15c2235e"

}

En cas de mise à jour de la donnée seulement :

{

"success":true,

"handleId":"11280/15c2235e", "metadata":"<nkl:Data><nkl:identifier>11280/d2ae4a92</nkl:identifier><nkl:checksum>dbb7ecc01bb94aba881075bd3fb92b3b2f738f0b</nkl:checksum><nkl:fileName>nouvea-nom-fichier-du-fichier-de-data</nkl:fileName></nkl:Data>",

"message":"donnée mise à jour avec succès : 11280/15c2235e"

}

Note : dans ce cas, le descripteur de métadonnées en sortie contiendra seulement le nouveau checksum et le nouveau file-name, mais on ne reconstruit pas l'intégralité du descripteur de métadonnées.

## DELETE /api/v1/data/<handle> : supprimer une donnée

### Description

Supprime une donnée et les métadonnées associées dans Nakala.

### Paramètres

|  |  |  |  |
| --- | --- | --- | --- |
| **Paramètre** | **Obligatoire** | **Description** | **Valeurs** |
| *Paramètres d’authentification, voir la section «  »* | | | |
| Handle de la donnée | oui | Le handle de la donnée à supprimer doit suivre la racine « …/api/v1/data/… », par exemple « …/api/v1/data/11280/abcdef ». | Le handle de la donnée à supprimer. |

### Exemple

curl -H "Content-Type: application/octet-stream" -X DELETE "https://www.nakala.fr/nakala/api/v1/data/11280/ d2ae4a92?email=thomas.francart@sparna.fr&key=28c34986-af44-4e92-8d78-81252696b3c7"

### Retour

{

"handleId":"11280/d2ae4a92",

"success":true,

"message":"Data supprimée avec succès : 11280/d2ae4a92"

}

## GET /api/v1/data/<handle> : accéder à une donnée protégée

### Description

Retourne une donnée de Nakala, éventuellement protégée par des droits d’accès.

|  |  |  |  |
| --- | --- | --- | --- |
| **Paramètre** | **Obligatoire** | **Description** | **Valeurs** |
| *Paramètres d’authentification, voir la section «  »* | | | |
| Handle de la donnée | oui | Le handle de la donnée à laquelle on souhaite accéder doit suivre la racine « …/api/v1/data/… », par exemple « …/api/v1/data/11280/abcdef ». | Le handle de la donnée à supprimer. |

### Exemple

curl -H "Content-Type: application/octet-stream" -X GET "https://www.nakala.fr/nakala/api/v1/data/11280/ d2ae4a92?email=thomas.francart@sparna.fr&key=28c34986-af44-4e92-8d78-81252696b3c7"

### Retour

* En cas de succès, cette API renvoie directement le fichier de données et ne renvoie pas de JSON.
* S'il y a une erreur (typiquement le user n'a pas le droit d'accéder à la donnée), l'API renvoie le JSON suivant :

{

"handleId":"11280/15c2235e",

"success":false,

"message":"L'utilisateur nicolas.larrousse@huma-num.fr ne peut pas accéder à la donnée : 11280/15c2235e"

}

# Description de l’API « collection »

## POST /api/v1/collection : créer une collection

### Description

Créer une nouvelle collection dans le projet Nakala.

### Paramètres

|  |  |  |  |
| --- | --- | --- | --- |
| **Paramètre** | **Obligatoire** | **Description** | **Valeurs** |
| *Paramètres d’authentification, voir la section «  »* | | | |
| Le paquet zip | oui | Le paquet ZIP passé dans le body de la requête ; attention que le « Content-Type » de la requête ne soit pas positionné automatiquement à “multipart/form-data” ou “application/x-www-form-urlencode”; |  |

### Exemple d’appel

curl -H "Content-Type: application/octet-stream" -X POST --data-binary @collection.zip "https://www.nakala.fr/nakala/api/v1/collection?email=thomas.francart@sparna.fr&key=28c34986-af44-4e92-8d78-81252696b3c7"

### Retour

{

"success":true,

"handleId":"11280/15c2235e ",

"metadata":"<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?><nkl:Collection 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\">...\n\n<nkl:identifier>11280/d2ae4a92</nkl:identifier></nkl:Collection>",

"message":"Collection créée avec succès : 11280/15c2235e"

}

## PUT /api/v1/collection/<handle> : modifier une collection

### Description

Remplace les métadonnées d’une collection existante.

### Paramètres

|  |  |  |  |
| --- | --- | --- | --- |
| **Paramètre** | **Obligatoire** | **Description** | **Valeurs** |
| *Paramètres d’authentification, voir la section «  »* | | | |
| Handle de la collection | oui | Le handle de la collection à modifier doit suivre la racine « …/api/v1/collection/… », par exemple « …/api/v1/collection/11280/abcdef ». | Le handle de la collection à modifier. |
| Le paquet zip | oui | Le paquet ZIP passé dans le body de la requête ; attention que le « Content-Type » de la requête ne soit pas positionné automatiquement à “multipart/form-data” ou “application/x-www-form-urlencode”; |  |

### Exemple d’appel

curl -H "Content-Type: application/octet-stream" -X PUT --data-binary @collection.zip "https://www.nakala.fr/nakala/api/v1/collection/11280/15c2235e?email=thomas.francart@sparna.fr&key=28c34986-af44-4e92-8d78-81252696b3c7"

### Retour

{

"success":true,

"handleId":"11280/15c2235e",

"metadata":"<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?><nkl:Collection 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\">...\n\n<nkl:identifier>11280/d2ae4a92</nkl:identifier></nkl:Collection>",

"message":"collection à jour avec succès : 11280/15c2235e"

}

## DELETE /api/v1/collection/<handle> : supprimer une collection

### Description

Supprime une collection dans Nakala.

### Paramètres

|  |  |  |  |
| --- | --- | --- | --- |
| **Paramètre** | **Obligatoire** | **Description** | **Valeurs** |
| *Paramètres d’authentification, voir la section «  »* | | | |
| Handle de la collection | oui | Le handle de la collection à supprimer doit suivre la racine « …/api/v1/collection/… », par exemple « …/api/v1/collection/11280/abcdef ». | Le handle de la collection à supprimer. |
| deleteData | Non | Indique si l’on souhaite que les données dans la collection soient automatiquement supprimées. Par défaut ce paramètre vaut « false », et les données existantes rangées dans cette collection seront « remontées » sur la collection racine. | true/false. |

### Exemple d’appel

curl -H "Content-Type: application/octet-stream" -X DELETE "https://www.nakala.fr/nakala/api/v1/collection/11280/15c2235e?email=thomas.francart@sparna.fr&key=28c34986-af44-4e92-8d78-81252696b3c7"

### Retour

{

"handleId":"11280/15c2235e",

"success":true,

"message":"Collection supprimée avec succès : 11280/15c2235e"

}