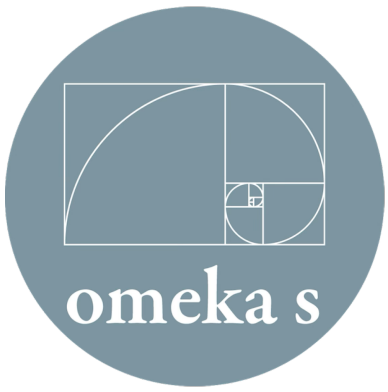
VADÉMÉCUM OMEKA S & IIIF

Association des usagers francophone de Omeka

Version du 12 novembre 2025



AUFO

PRÉSENTATION

Ce document est rédigé collectivement par des membres du groupe de travail « développement » de l'association des usagers francophones de Omeka (AUFO).

- La version en ligne est disponible à cette adresse https://vade-mecum-iiif.omeka.fr/
- Le dépot Gitlab est accessible sur https://gitlab.com/OmekaFR/vade-mecum-iiif

Un Omeka S branché IIIF ?

Vous souhaitez utiliser le protocole IIIF pour intégrer, exposer, agréger, diffuser des images ? Peut-être même offrir de nouveaux services à vos utilisateurs comme la manipulation, la comparaison, la recherche plein texte, les transcriptions ou les annotations ?

Des premiers pas jusqu'aux usages les plus avancés, suivez le guide !

Note d'édition

Ce « vadémécum » se veut à la fois une introduction générale au maniement de IIIF dans le cadre du logiciel Omeka et un guide pratique qui traite au fur et à mesure d'utilisations de plus en plus avancées. Dans la seconde partie, intitulée « Cas pratiques / Retours d'expérience », vous pourrez retrouver des cas d'usages plus détaillés, des recettes et des propositions de configuration pour certains modules ou outils complexes.

Le vadémécum présente l'état actuel des connaissances et observations sur les principales fonctionnalités et modules liés à IIIF sans préjuger de leurs évolutions ou adaptations particulières. Le groupe de travail se donne pour mission de réviser régulièrement ce guide en fonction des nouveaux développements.

La présente version, propose donc un premier panorama que nous espérons pouvoir être utile en l'état. Néanmoins, Omeka évolue et certains modules pourraient être remaniés profondément à plus ou moins brève échéance. Nous nous efforcerons d'actualiser régulièrement ce vadémécum.

Corrections et ajouts

Les sources de ce documents sont accessibles sur un dépôt Gitlab public. Vous pouvez y retrouver les différentes versions de son contenu et contribuer à son amélioration en ouvrant un ticket pour apporter vos remarques, corrections, propositions, etc.

INTRODUCTION

Logiciel libre à la croisée du système de gestion de contenus, de la gestion de collections patrimoniales ou de corpus scientifiques, et de l'édition d'archives numériques. Omeka permet de gérer, de collecter, de publier et de partager des contenus et des données de façon simple, flexible et interopérable. Son architecture de base épurée et modulaire offre la possibilité de s'adapter aux besoins de chaque projet grâce à l'ajout de fonctionnalités et au choix de thèmes personnalisables.

Son développement est assuré par la Corporation for Digital Scholarchip, qui développe également les logiciels Zotero et Tropy.

Omeka existe en deux versions :

- **Omeka Classic**, la version initiale, propose un schéma de bibliothèque numérique complet : gestion des documents et des collections, suivant des descriptions en Dublin Core, création de pages et de parcours virtuels publiques sur un site dédié. Sa prise en main rapide permet de monter facilement des projets individuels.
- **Omeka S**, version développée plus récemment, reprend les mêmes fonctionnalités tout en suivant les standards du web sémantique. Elle a notamment la particularité de permettre une diffusion multi-sites des ressources.

Dans une perspective de partage de ressources, Omeka s'associe souvent au standard ouvert IIIF toujours plus utilisé pour exposer des images, audios et vidéos sur le Web. Omeka Classic et Omeka S supportent IIIF. Le présent vadémécum s'intéresse essentiellement à l'utilisation de IIIF avec Omeka S. Cependant, une annexe présente l'utilisation de IIIF avec Omeka Classic.

> 👉 Omeka https://omeka.org
> 👉 IIIF https://iiif.io

QU'EST-CE QUE IIIF ?

IIIF, le protocole international d'interopérabilité des images, s'est imposé en quelques années comme un standard de fait et comme une brique technologique essentielle pour décloisonner les collections numérisées des institutions patrimoniales et scientifiques à l'échelle mondiale. Pour tout comprendre de cet écosystème, du fonctionnement de IIIF autant de ses principes de base que de son implémentation technique, nous ne pouvons que renvoyer à la riche documentation proposée par le site Biblissima.

Pour faire simple :

- IIIF désigne à la fois un modèle et un ensemble de spécifications techniques permettant de diffuser, de présenter et d'annoter des ressources numériques : images, audio/vidéo et prochainement 3D.
- IIIF représente aussi une communauté, qui développe des API ouvertes (des façons communes de faire dialoguer des logiciels entre eux), les implémente dans des logiciels de différentes natures. Cette communauté expose des contenus interopérables sur le Web, du point de vue des bibliothèques, des archives, des musées, des labos, etc.

Pour les institutions et les administrateurs de sites sous Omeka S, l'utilisation du protocole IIIF peut présenter beaucoup d'avantages : faciliter la diffusion et le partage de ses ressources, utiliser des ressources externes sans les dupliquer, proposer une meilleure expérience utilisateur, développer de nouveaux services, etc. La liste est longue !

> IIIF vise à favoriser la libre circulation des images, leur éventuelle réutilisation et manipulation par des utilisateurs distants. Cette approche peut ne pas convenir à des ressources couvertes par des droits restrictifs ou à des environnements de diffusion fermés.
>
> Il n'est pas non plus question d'utiliser toute la puissance de IIIF sans respecter un certain nombre de bonnes pratiques qui accompagnent son usage, notamment dans le respect des sources, bien entendu.

Donner accès à des ressources via IIIF implique que celles-ci soient accessibles et utilisables dans un temps long. Cela vous engage aussi à mettre en place des outils permettant de stabiliser ces accès et d'en assurer la continuité, par exemple, en maintenant des URL pérennes autant que possible.

Vous trouverez ci-dessous une description succincte des deux API IIIF principales utilisées avec Omeka-S. Les informations sont extraites de la documentation de Biblissima.

API Image IIIF

L'API Image (Interface de Programmation d'Application) propose une méthode qui permet de récupérer les pixels d'une image et de les manipuler à distance. Pour ce faire, on utilise une syntaxe d'URL standardisée, c'est-à-dire une façon spécifique de formuler les adresses URL pour effectuer ces opérations.

Avec cette syntaxe, on peut également récupérer des informations liées à l'image, stockées dans un fichier texte au format JSON :

- modèle d'URL: `{scheme}://{server}{/prefix}/{identifier}/info.json`
- exemple : https://api.nakala.fr/iiif/10.34847/nkl.8e31p9d2/014f975b5550100f7a5b977ae409d4c51f3ae263/info.json

Ce fichier `info.json` stocke des informations techniques telles que les dimensions (largeur et hauteur), les formats pris en charge, les échelles de découpage (*tiles* ou « tuiles »), et les fonctionnalités disponibles pour cette image. C'est grâce à ce fichier que des applications clientes comme les visionneuses peuvent comprendre comment interagir avec l'image, par exemple en déterminant les recoupages possibles ou les formats d'image disponibles. En revanche, ce fichier n'est pas destiné à stocker des métadonnées documentaires de l'image qui seront plutôt prises en charge par l'API Présentation.

Cette syntaxe d'URL peut s'appliquer à l'image directement afin d'opérer certains traitements comme le zoom, le redécoupage, l'inversion, etc. :

- modèle d'URL: `{scheme}://{server{/prefix}/{identifier}/{region}/{size}/{rotation}/{quality}.{format}`
- exemple : https://api.nakala.fr/iiif/10.34847/nkl.8e31p9d2/014f975b5550100f7a5b977ae409d4c51f3ae263/full/full/0/default.jpg

> Le service IIIF Image Manipulation Tool permet d'extraire une zone et d'effectuer ces traitements à partir d'une URL `info.json`.

API Présentation IIIF

L'API Présentation précise les métadonnées (descriptives, structurelles, techniques) nécessaires à la présentation d'un objet numérique dans une interface (par exemple une visionneuse d'images, ou tout autre environnement manipulant des images et autres médias supportés par IIIF).

Toutes ces informations sont contenues dans un fichier appelé « manifeste », une sorte d'enveloppe virtuelle formant l'unité de distribution élémentaire dans l'univers IIIF. Tout comme un manifeste qui énumère les passagers ou recense la cargaison d'un avion ou d'un bateau, un Manifeste IIIF embarque les informations nécessaires à la représentation d'un document : métadonnées, structure, références, agencement des médias, annotations, droits d'utilisation ou d'accès, etc.

C'est en général ce fichier que vont manipuler les logiciels pour interagir avec une ressource, la visualiser, ou la transférer vers un autre outil.

L'API Présentation constitue à la fois :

- un format d'échange de données, structurées et encodées selon le format JSON-LD (JavaScript Object Notation for Linked Data)
- un modèle décrivant la représentation numérique d'un

  objet :
  la séquence ordonnée des médias qui le compose (images, audios, vidéos), sa structure interne, ses métadonnées, ses liens avec d'autres ressources, ses annotations.
- Exemple d'un manifeste IIIF: https://nakala.fr/data/10.34847/nkl.a3d67u78

Trouver des manifestes IIIF

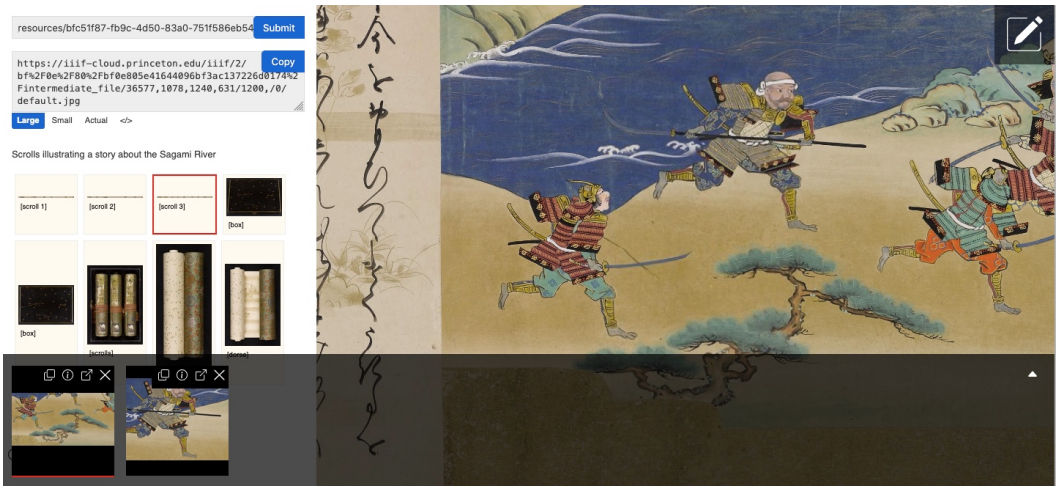
- Les URL des manifestes sont souvent fournies dans les notices des ressources exposées ou dans les informations fournies par les visionneuses. Cependant, elles ne sont pas toujours faciles à trouver. Elles sont normalement indiquées par le logo IIIF :

  .

- Le site officiel IIIF entretient un guide pour trouver des ressources IIIF. Il existe aussi des extensions aux navigateurs qui indiquent la présence de ressources IIIF dans les pages visitées : detektIIIF, IIIF Download, Open in IIIF Viewer,

Liens entre les API Images et Presentation

- ce tutoriel propose un formulaire permettant d'extraire chaque image d'un manifeste et d'en récupérer l'URL `info.json`,
- le service IIIF Cropping Tool permet de visualiser les images d'un manifeste et d'extraire des zones de ces images.



Exemple d'utilisation de IIIF Cropping Tool à partir du manifeste IIIF du manuscript japonais *Sagami River* (Princeton University Library)

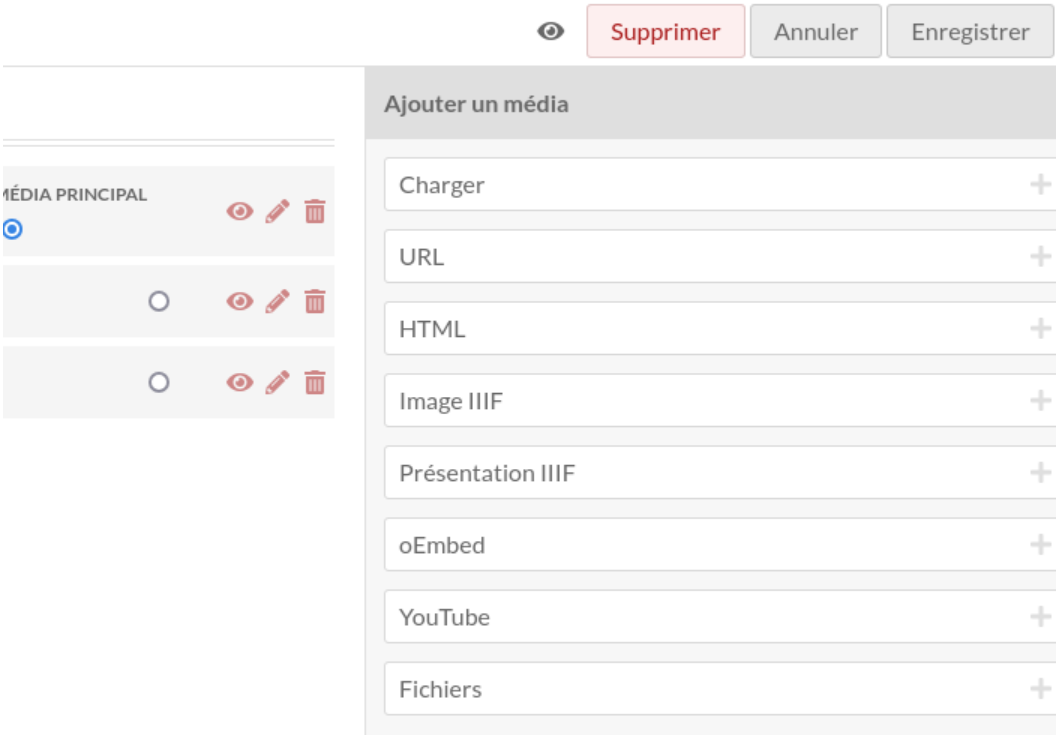
OMEKA S ET IIIF

Que propose Omeka S par défaut ?

Omeka S ne propose pas à proprement parler de support des API IIIF : en effet la version de base du logiciel ne permet pas de produire et d'exposer via IIIF des ressources internes présentes dans une instance Omeka S.

Il offre toutefois de façon native **deux fonctionnalités d'intégration de ressources IIIF externes** (délivrées par des services tiers). Celles-ci sont accessibles via l'onglet `Médias` dans l'interface d'édition d'un contenu Omeka, sous la forme de deux widgets d'ajout de média :

- Média `Image IIIF` (URL d'une image IIIF unique)
- Média `Présentation IIIF` (URL d'un Manifeste IIIF)



Appel du bloc IIIF Image lors de la création d'un media

Média `Image IIIF`

Ce widget permet d'associer à un contenu Omeka une image IIIF via son URL.

L'URL `info.json` fournie par un service tiers (par exemple Nakala ou Cantaloupe) est à entrer dans le formulaire. Il est possible d'associer manuellement plusieurs images IIIF à un même contenu. Pour le faire par lots, il faut passer par un module de type CSVImport.

Par défaut les images sont affichées de façon isolée sur la page, chacune dans une visionneuse séparée, ce qui ne permet pas leur feuilletage.

> **À noter :**
> l'intégration se fait non pas par l'URL de l'image en tant que telle mais grâce au fichier d'informations sur l'image exposé par l'API Image de IIIF (URL avec le suffixe `/info.json`);
>
> l'image IIIF pleine taille n'est pas copiée dans Omeka S, elle est simplement référencée via l'API Image de IIIF. Seules des métadonnées techniques propres à l'image sont importées. Comme pour les autres images, Omeka S génère plusieurs versions basse résolution de l'image (*large*, *medium* et *square*) utilisées dans l'interface d'administration ou sur les sites d'Omeka S.

Une visionneuse intégrée basée sur le composant libre OpenSeadragon permet de visualiser les images IIIF dans l'interface d'administration comme dans les pages publiques des sites.

Média `Présentation IIIF`

Ce widget permet d'associer à un contenu Omeka un Manifeste IIIF via son URL. Par défaut l'objet numérique en question est affiché sur la page du contenu dans la visionneuse Mirador, embarquée par défaut dans Omeka S.

> **À noter :**
> les images ainsi que les métadonnées référencées dans le Manifeste ne sont pas importées dans Omeka S ;
>
> l'URL du Manifeste et son contenu au format JSON-LD sont néanmoins stockés dans la base de données Omeka S en tant que média associé à un contenu.

Une visionneuse intégrée basée sur Mirador permet de visualiser individuellement les documents présentés sous forme de Manifestes IIIF dans l'interface d'administration comme dans les pages publiques des sites.

> **À noter :**
> Cette version de Mirador est allégée et n'intègre pas toutes les fonctionnalités habituelles de l'outil.

Étendre les fonctionnalités IIIF de Omeka S avec des modules

Les possibilités liées à IIIF offertes par la version « de base » du logiciel sont donc minimales, bien qu'elles présentent déjà beaucoup d'intérêt pour un certain nombre d'usages (voir par exemple le cas d'usage Omeka S comme agrégateur).

Plusieurs modules ont été développés afin d'améliorer l'intégration du protocole IIIF à Omeka S : d'une part pour utiliser au mieux les diverses potentialités de IIIF et, d'autre part, pour transformer sa propre instance Omeka S en véritable service IIIF pour les clients externes.

Tableau récapitulatif

	Correspondance avec les API IIIF	Fonctionnalités	Complexité d'utilisation
Par défaut (à partir de la version 4)		Intégration et affichage d'images IIIF dans Omeka S	☆
Module IIIF Presentation	*Presentation API*	Génération de Manifestes IIIF	☆☆
Module IIIF Server	*Presentation API* (et +?)	Génération de Manifestes IIIF	☆☆☆
Module Image Server	*Image API*	Serveur d'images IIIF	☆☆☆☆
Module IIIF Search	*Content Search API*	Recherche plein texte dans un document transcrit dans un format ALTO/XML (ou TSV)	☆☆

Transformer son Omeka S en service IIIF ?

Pour exposer des ressources internes via le protocole IIIF, il est recommandé de satisfaire aux préconisations des deux principales API IIIF : l'API Image et l'API Présentation.

> **À noter :**
> Ces deux API peuvent être implémentées de façon indépendante : ainsi il est possible de proposer l'API Image sans l'API Présentation, et vice versa. Néanmoins les deux sont le plus souvent utilisées de façon conjointe pour tirer le meilleur parti de IIIF. Pour une bibliothèque numérique, le fait de ne proposer que l'API Présentation, sans l'API Image, peut se justifier si les images diffusées sont toutes en faible résolution et ne nécessitent pas de zoom profond. De plus, dans le cas de contenus audio ou vidéo, seule l'API Présentation sera mobilisée.

Exposer ses médias avec l'API Image

Il s'agit de pouvoir délivrer des images manipulables par des clients externes. Pour cela, elles doivent être préparées en amont (génération statique) ou servies directement (génération dynamique ou « à la volée ») par un dispositif spécifique qui générera également le fichier d'information de l'image (`info.json`) et l'URL correspondante. Dans ce cas, trois solutions s'offrent à vous :

- Utilisation d'un service externe qui s'occupe d'exposer les images en IIIF. C'est le cas par exemple de l'entrepôt de données Nakala (voir le cas d'usage Nakala). Il existe aussi des services commerciaux : Micrio, IIIFHosting, Teklia, etc.
- Ajout d'un module spécifique sur l'instance Omeka S qui va faire office de serveur d'images. C'est ce que propose le module ***Image Server*** (Voir plus bas).
- Installation autonome d'un logiciel dédié, dit « serveur d'images », que l'on va coupler à l'instance Omeka S. Dans ce cas, il faut avoir certaines compétences et ressources informatiques comme l'accès à une machine virtuelle (qui peut être la même que celle de l'instance Omeka S, ou non).

Le choix entre ces différentes options dépend totalement de votre infrastructure, de la typologie de vos documents et médias, des ressources et compétences dont vous disposez. Pour une bibliothèque virtuelle basée sur Omeka S, l'utilisation d'un serveur d'images IIIF externe plutôt qu'un serveur interne reposant sur des modules Omeka S présente plusieurs avantages :

- Un serveur d'images externe est spécialement conçu pour le traitement et la diffusion d'images en respectant au mieux l'API Image de IIIF. Il offre de bonnes performances et fournit des fonctionnalités avancées.
- Il permet d'optimiser les ressources allouées par rapport aux besoins : il peut être dimensionné indépendamment de l'installation Omeka S et s'adapter aux évolutions des volumes et des demandes.

Le recours au module ***Image Server*** d'Omeka S est plus simple à configurer et est suffisant pour des collections modestes. Cependant, pour des projets plus importants ou nécessitant des performances optimales, un serveur d'images IIIF dédié représente une solution plus robuste.

Cantaloupe et IIPImage sont les serveurs d'images les plus utilisés ; ce sont des logiciels libres. Les premiers pas avec un serveur d'images externe sont abordés dans la partie Utiliser un serveur d'images IIIF autonome – Cantaloupe.

Module *Image Server*

Le module *Image Server*, développé et maintenu par Daniel Berthereau, simule un serveur d'images interne à l'instance Omeka S quand on ne dispose pas de serveur d'images en propre (comme Cantaloupe). Il n'est donc pas utile de l'implémenter si on utilise déjà un serveur d'images externe.

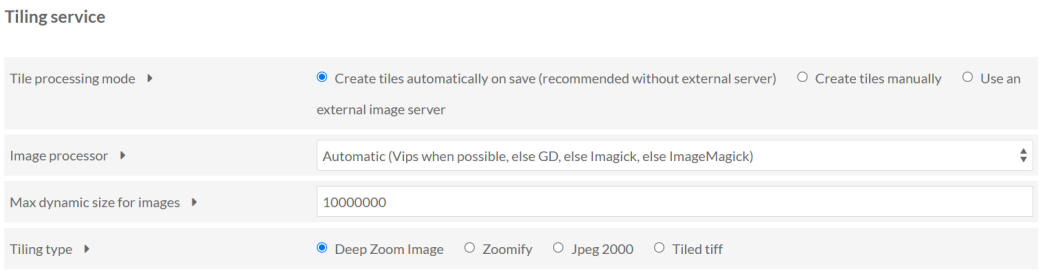
> ⚠ L'installation du module *Image Server* nécessite l'installation préalable de deux autres modules *Common* et *IIIF Server*.

Une fois les modules *Common* et *IIIF Server* installés, vous pouvez passer à l'installation du module *Image Server*.

Image Server dote Omeka d'un serveur d'images IIIF. Les spécifications complètes de l'API Image sont prises en charge (versions 2 et 3). Pour des besoins standards vous pouvez laisser cochées les cases **Image Api 2 niveau 2** et **Image API 3 niveau 2** dans le formulaire de paramétrage du module.

Pour permettre les fonctionnalités de zoom proposées par IIIF, *Image Server* extrait des *tuiles* par niveau de zoom à partir des images originales liées aux ressourcs. Il stocke les tuiles à part dans le répertoire `files/tile` de l'installation d'Omeka S. Ces tuiles seront appelées par la visionneuse pour afficher l'image. C'est ce que l'on nomme un *tuilage statique*. On peut suivre et garder le paramétrage par défaut proposé dans la section *Service de tuilage*.

Au moins un processeur d'image, qui se charge de découper les images en tuiles, doit être installé sur le serveur qui héberge votre Omeka S. Quatre options sont possibles dans les paramétrages du module. Elles sont rangées par ordre décroissant de rapidité pour cette opération : **Vips**, **GD (extension PHP)**, **Imagick (extension PHP)**, **ImageMagick**. Si aucun de ses logiciels n'a été installé lors de l'installation de votre Omeka S par votre équipe informatique, il vous sera nécessaire de revenir vers eux pour en faire la demande. En l'absence de renseignement clair sur la question, la case par défaut *Automatic : Vips when possible, else GD, else Imagick, else ImageMagick* convient pour le fonctionnement du module.



> **À noter :**
> Le processeur d'image **ImageMagick** est nécessaire pour importer des médias image dans Omeka S. On peut raisonnablement considérer que c'est le premier processeur installé sur un serveur Omeka S dans une configuration basique.

Le format de tuilage indique le format sélectionné pour générer les tuiles par le processeur d'image sélectionné en amont (**ImageMagick**, **Vips**, etc.).

DeepZoom est sélectionné par défaut et est recommandé pour la plupart des usages. Les tests effectués montrent qu'il est pour le moment le plus performant des formats de tuiles proposés par le module *Image Server*.

⚠ Pour vérifier que *Image Server* fonctionne bien, vous pouvez tester l'URL IIIF pour afficher un media donné.

▼ Détail de la procédure de test

Comme vu précédemment pour l'API Image, la syntaxe de l'URL d'accès aux informations liées à l'image est de la

forme :

```
{scheme}://{server_url}/iiif/{api_image_version}/{media_id}/info.json
```

=> exemple: https://bina.bulac.fr/iiif/2/408053/info.json

Pour la version de l'API Image vous avez le choix entre les valeurs *2* et *3*.

Si la réponse JSON renvoyée est similaire à la capture ci-dessous, c'est que votre module *Image Server* est correctement configuré :


```
{
  "@context": "http://iiif.io/api/image/2/context.json",
  "@id": "https://bina.bulac.fr/iiif/2/408053",
  "protocol": "http://iiif.io/api/image",
  "width": 949,
  "height": 1417,
  "sizes": [
    {
      "width": 134,
      "height": 200
    },
    {
      "width": 536,
      "height": 800
    },
    {
      "width": 949,
      "height": 1417
    }
  ],
  "tiles": [
    {
      "width": 256,
      "scaleFactors": [1, 2, 4, 8]
    }
  ],
  "profile": [
    "http://iiif.io/api/image/2/level2.json"
  ]
}
```


Vous pouvez également tester en affichant directement l'image servie par *Image Server* à l'aide de la syntaxe suivante :

```
{scheme}://{server_url}/iiif/{api_image_version}/{media_id}/{region}/{size}/{rotation}/{qualit
```

=> exemple: https://bina.bulac.fr/iiif/2/408055/full/673,800/0/default.jpg

Pour aller plus loin, vous pouvez consulter l'annexe qui est dédiée à la configuration d'*Image Server* et aux processus de tuilage.

Exposer ses ressources avec l'API Présentation

Deux modules ont pour objectif de générer des Manifestes IIIF à partir des métadonnées descriptives et des médias de chaque Contenu :

- ***IIIF Presentation***
- ***IIIF Server***

Les deux modules suivent la syntaxe définie par l'API IIIF Présentation, qui a actuellement **2 versions en activité**. Ils entendent prendre en charge les deux versions. Dans les faits, ils créent donc 2 Manifestes pour chaque Contenu : un pour l'API Présentation 2, et un pour l'API Présentation 3.

Les deux modules forgent des Manifestes présentant les informations essentielles à la contextualisation d'un Contenu : titre et autres métadonnées descriptives, attribution, licence d'utilisation, mention de l'API Omeka S qui a permis leur récupération, et - plus accessoirement - sens de lecture des médias. Ils signalent aussi les collections Omeka ainsi que les Collections IIIF auxquelles le Manifeste peut appartenir.

Les deux modules affichent de manière complète la liste de tous les Canevas (c'est-à-dire les "vues" d'un document numérisé) compris dans le document, avec, pour chacun, l'URI de chaque image fourni par l'API Image. Les Manifestes intègrent également les potentielles Annotations qui ont pu y être jointes. Chaque Manifeste dispose aussi de sa miniature image (*thumbnail*).

Les versions API Présentation 2 et API Présentation 3 n'utilisent pas exactement la même syntaxe, mais dans tous les cas ces informations figurent.

Module *IIIF Presentation*

Ce module est produit par Omeka Team, concepteurs d'Omeka S, qui en assure également la maintenance. Il permet d'obtenir des Manifestes IIIF contenant les informations détaillées ci-dessus. Il n'est pas paramétrable depuis le tableau de bord de votre installation. Il diffuse par défaut des Manifestes utilisant la version 3 de l'API Présentation.

Module *IIIF Server*

Ce module est développé par Daniel Berthereau. Il était initialement fusionné avec *Image Server*, ce qui se traduit aujourd'hui par leur dépendance mutuelle. Il offre de plus larges possibilités de personnalisation que *IIIF Presentation*, pourvu que l'on s'approprie son tableau de configuration pour le moins foisonnant.

La configuration par défaut du module est fonctionnelle. Aucune modification n'est requise pour afficher les images et les métadonnées qui leur sont associées. Il est néanmoins utile de tester le bon fonctionnement de cette configuration par défaut avant de commencer à la personnaliser.

⚠ Avant d'effectuer ce test, si ce n'est pas déjà fait, il vous faut finaliser et tester l'installation du module ***Image Server***.

▼ Détail de la procédure test

Ce test peut s'effectuer dans un premier temps sur **l'URL d'un Manifeste IIIF**, dont la syntaxe proposée par *IIIF Server* est de

type :

`{scheme}://{server_url}/iiif/{api_presentation_version}/{item_id}/manifest`
=> exemple: https://bina.bulac.fr/iiif/2/406292/manifest

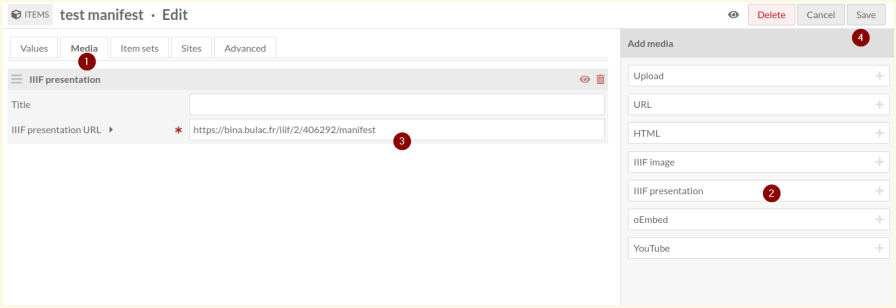
Entré en barre de recherche, cet URL doit charger l'entièreté du Manifeste au format JSON, comme sur cet exemple :


```
{
    "@context": "http://iiif.io/api/presentation/2/context.json",
    "@id": "https://bina.bulac.fr/iiif/2/406292/manifest",
    "@type": "sc:Manifest",
    "label": "رباعي",
    "metadata": [
        {
            "label": "Titre",
            "value": [
                "رباعي",
                "Rubā'ī"
            ]
        },
```

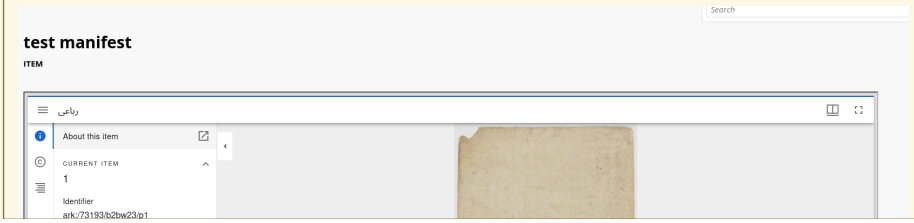

Une fois cette première étape passée, vous pouvez vérifier que le Manifeste est utilisable par Omeka S en l'enregistrant comme média pour un Contenu, afin qu'il puisse être visualisé :

Dans la fenêtre d'édition d'un item dans l'interface admin d'Omeka S :

1. Rendez vous dans l'onglet **Media**
2. Cliquez sur ajouter un media **IIIF Presentation**
3. Renseignez l'URL du Manifeste que vous avez testé au préalable
4. Sauvegardez les modifications



Le Contenu test donnera à voir l'entièreté des images fournies par le Manifeste IIIF, dans une fenêtre Mirador qu'Omeka S propose par défaut depuis ses versions 4.0.0.



Particularités du module *IIIF Server*

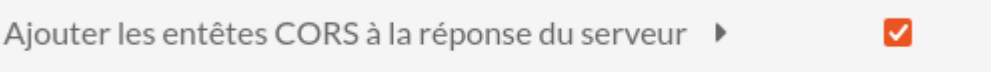
Le module fournit des Manifestes IIIF plus détaillés que ceux du module *IIIF Presentation*, notamment sur les points suivants :

- Mention de tous les médias importés dans le Contenu source, et disponibles au téléchargement, dans la section “rendering” ;
- Intégration d’un formulaire de requête pour l’API IIIF Content Search et lecture de fichiers de transcription texte (voir le point dédié à la recherche en texte intégral) ;
- Utilisation d’un **identifiant pérenne** dans l’URI du Manifeste (ARK, DOI…), intégré dans Omeka S au préalable au moyen du module **Clean Url** (attention, cette fonctionnalité n’est toutefois pas encore complètement étendue aux Manifestes IIIF intégrant de la transcription texte)

Configurer *IIIF Server*

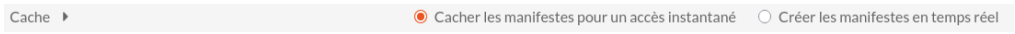
Le module est entièrement configurable, et la pléthore d’options peut être déroutante. Nous listons ici les plus importantes, par ordre d’affichage dans le menu :

- **En-têtes CORS** : elles sont essentielles à la transmission du Manifeste IIIF au navigateur des usagers. La case est cochée par défaut. Néanmoins, si vos gestionnaires informatiques ont *déjà* paramétré l’ajout de ces en-têtes au niveau du serveur de votre installation, cela aura pour effet de les rendre inopérants. Rapprochez-vous donc de votre équipe informatique en cas de message d’erreur concernant un *“strict-origin-when-cross-origin”*

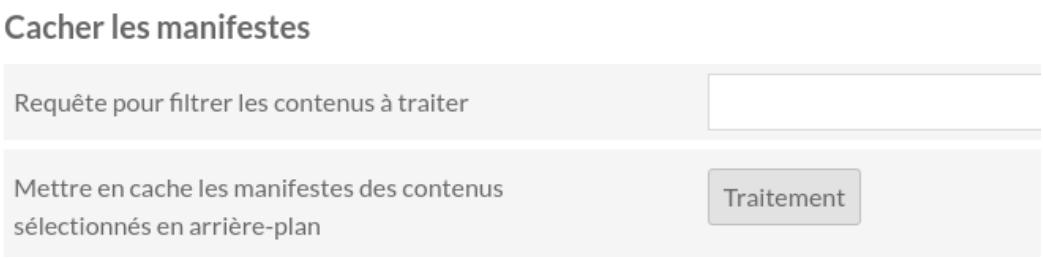


> La présence de ces entêtes CORS est indispensable pour permettre l’appel de vos Manifestes et de vos images via IIIF depuis des clients externes, c’est-à-dire depuis un nom de domaine différent de celui de votre institution. Ces clients sont le plus souvent des visualiseurs IIIF, mais toute application tierce s’exécutant dans un navigateur web est concernée. Sans cela le navigateur de l’internaute bloquera les requêtes pour des questions de sécurité. Ce paramétrage conditionne donc en grande partie l’interopérabilité et la réutilisation effectives des ressources que vous exposez via les API IIIF.
>
> Pour voir comment activer ces entêtes dans différents environnements serveur, vous pouvez vous reporter à cette page : https://enable-cors.org/server.html.
>
> Voir aussi les indications de cette annexe des spécifications IIIF : https://iiif.io/api/annex/notes/apache/#enabling-cors

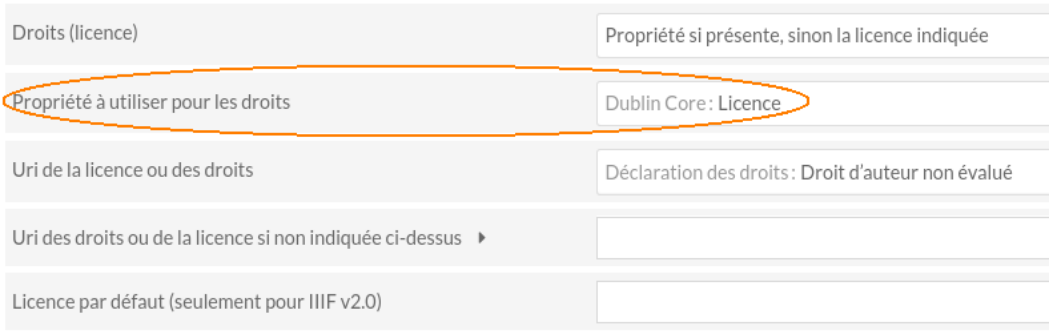
- **Cache** : par défaut, *IIIF Server* créé chaque Manifeste à la volée, en interrogeant l'API IIIF Image et l'API Omeka à chaque requête sur un Contenu. Mais si le Manifeste est particulièrement long, parce que le document initial comporte beaucoup de pages et/ou beaucoup de métadonnées descriptives, il peut être avantageux de cocher la case *"Cacher les manifestes pour un accès instantané"*. Dans ces conditions, *IIIF Server* écrit le Manifeste IIIF lorsque le Contenu est enregistré dans Omeka dans la *mémoire cache* du serveur de l'installation, pour qu'il puisse être remis à disposition très rapidement.



- Ce Manifeste *en cache* se met à jour de lui-même lorsque le Contenu est modifié ; néanmoins, ce n'est pas le cas lorsque les paramétrages du module *IIIF Server* sont modifiés. Il est alors nécessaire de lancer une tâche de mise à jour du cache, à la toute fin du tableau de configuration. Par défaut, le module met à jour l'ensemble des Manifestes IIIF du cache.



- **Licence d'exploitation** : dans le bloc ci-dessous, il convient soit de renseigner la propriété utilisée pour la licence par les Contenus (choix fait dans cet exemple), soit de renseigner l'URI de la licence choisie deux champs plus bas. Les champs non utilisés peuvent rester vides.



Les principales visionneuses IIIF

Par défaut, Omeka S comporte bien une visionneuse dédiée aux médias susceptible de prendre les traits d'une fenêtre Mirador en présence de Manifestes ou d'OpenSeadragon en présence d'images IIIF. Toutefois, ces visionneuses demeurent limitées dans leurs fonctionnalités et leur personnalisation.

Disposer d'une meilleure marge de manœuvre passe par l'installation de modules. Actuellement, **4 modules** proposent la visualisation de ressources IIIF : **Universal Viewer**, **Mirador Viewer**, **Diva Viewer** et **Octopus Viewer**.

Universal Viewer

Le module Universal Viewer adapte pour Omeka le projet du même nom. Comme son nom l'indique,

Universal Viewer se veut une visionneuse universelle capable de prendre en charge une grande variété de formats.

Elle peut prendre notamment en charge les médias hébergés sur un serveur IIIF (y compris ceux au format pdf et ePub) *et* les URL de vidéos YouTube.

Le téléchargement des médias est possible, tant qu'il s'effectue *vue par vue*: tous les médias associés à une vue (un *Canevas*) dans tous leurs formats, sont accessibles depuis une icône dédiée. Mais le téléchargement groupé de l'ensemble des médias d'un Contenu n'est pas envisageable.

Le module installe par défaut une version épurée de la visionneuse (dite *"vanilla"*), que l'on peut souhaiter compléter par les différents plugins créés par les contributeurs du projet. La marche à suivre est détaillée dans la documentation du module ; elle requiert néanmoins d'être familier des opérations d'administration système (navigation dans le système et gestion de fichiers par ligne de commande, installation de librairies). Ces plugins concernent au moins le traitement de l'OCR dans la visionneuse et l'ajout d'une barre de recherche en texte intégral.

Par défaut, le module est limité aux formats mp3 et ogg pour l'audio , et webm et ogv pour la vidéo. Toutefois, la visionneuse a été initialement conçue pour supporter une plus grande variété de formats, notamment mp4, wav et mpeg ; ces options peuvent être activées dans le code du module.

Contrairement aux deux autres visionneuses suivantes, Universal Viewer ne permet pas de charger un Manifeste IIIF externe par son URL d'accès, et n'affiche pas les éventuelles annotations, ou indexation des médias.

Le développement du module est actif. Après quelques temps sans évolutions, le projet Universal Viewer a été repris à la fin de l'année 2024, et la version 4.1.0 est en production ; le module a été mis à jour par Daniel Berthereau pour l'implémenter en mai 2025.

Mirador Viewer

Le module Mirador Viewer fournit une visionneuse Mirador conçue comme un espace de travail à part entière pour l'utilisateur.

Premièrement, elle permet aux utilisateurs d'appeler n'importe quels Manifestes IIIF de leur choix via son bouton **+**, qui a pour effet de scinder la fenêtre en deux, et attribuer à chaque Manifeste un espace de visualisation.

Elle permet d'implémenter cinq plugins développés par les contributeurs du projet Mirador général, moyennant là encore l'exécution d'une série de commandes données en documentation :

- ***Annotations*** : ce plugin permet non seulement l'aperçu des annotations provenant du manifeste IIIf dans le volet latéral dédié aux informations descriptives, mais aussi l'ajout manuel de nouvelles annotations dont le graphisme put être personnalisé (formes, remplissages, contenu texte). À noter que ces nouvelles annotations ne sont pas exportées dans le manifeste IIIF ; elles restent circonscrites à la visionneuse.
- ***Image tools*** : ce plugin permet d'effectuer quelques petites modifications sur les médias de type image, par exemple sur leur couleur, leur luminosité ou leur orientation ;
- ***Download*** : Téléchargement dans les mêmes conditions qu'Universal Viewer : vue par vue
- ***Share*** : copie simplement l'URL du manifeste IIIF ;
- ***Text Overlay*** : plugin complexe proposant de visualiser la couche texte d'une ressource image (OCR/HTR), dans un volet latéral séparé et directement en surbrillance sur l'image. Le texte ainsi affiché peut être sélectionné au curseur et entré en barre de recherche. Pour fonctionner correctement, ce plugin nécessite que les fichiers xml-alto aient été importés avec les autres médias, et que leur contenu textuel ait été intégré au manifeste IIIF en tant qu'annotation (ce que propose le module IIIF Server).

Rappelons qu'en termes de médias, Mirador Viewer n'accepte que ceux qui proviennent d'un serveur IIIF, et ce sans les pdf et les ePub ; si les médias importés sont en pdf, la visionneuse ne les affichera pas, manifeste IIIF ou non. Mais il n'y a pas de limitation sur les formats image ou vidéo hors de celles imposées par Omeka.

Le module Mirador Viewer fournit également d'autres paramètres personnalisables, comme l'import/export d'un thème personnalisé pour sa fenêtre. Cette option n'est pas toujours fonctionnelle sur Omeka.

Il peut résulter de cette offre pléthorique une certaine opacification de la visionneuse : son espace se sature d'icônes cliquables, dont la fonction n'est pas d'emblée évidente. Le module Mirador Viewer est également l'un des plus complexes et longs à installer sur Omeka.

Le développement général du projet Mirador est actif et continu. La dernière version (3.4.3) est sortie en janvier 2025. Le développement du module Mirador Viewer pour Omeka est assuré par Daniel Berthereau, et est mis à jour environ une fois par an.

Octopus Viewer

Octopus Viewer se présente comme une méta - visionneuse spécifiquement développée pour Omeka S et susceptible de s'adapter au type de média en présence.

Par défaut, un volet latéral liste les médias, et un autre les métadonnées qui y sont rattachées (à ces médias seuls, et non l'entièreté du Contenu). Lorsque l'utilisateur sélectionne un média, si celui est de type pdf ou un manifeste IIIF importé, Octopus Viewer appelle une autre visionneuse *à l'intérieur de sa propre fenêtre*. Dans le cas de IIIF, cette autre visionneuse est une version standard de Mirador. La fenêtre complète de Mirador est encapsulée dans celle d'Octopus Viewer.

Néanmoins, le module n'a pas recours aux plugins détaillés plus haut : il en résulte que les Annotations IIIF peuvent être montrées, mais non créées sur place ; qu'il n'y aura pas d'outils de retouche d'image ; qu'il n'y aura pas de visualisation à part ou en surbrillance de l'OCR ; et pas de téléchargement direct, ou de copie de l'URL du manifeste IIIF. L'option de téléchargement est possible depuis la fenêtre d'Octopus Viewer, média par média.

Mais la possibilité de charger un manifeste IIIF extérieur reste la même - si la perspective de fenêtres encapsulées les unes dans les autres à plus de deux niveaux ne cause pas de vertige.

Il est recommandé d'importer le manifeste IIIF en tête des médias de chaque Contenu : Octopus Viewer ne parcourt pas les médias de lui-même, et n'affiche pas l'icône IIIF par défaut. Dans le cas d'une longue liste de

médias, l'utilisateur pourrait ignorer qu'une visualisation par IIIF est possible.

Le module prend au moins les formats jpeg, jp2, png, tiff, webp, pdf et mp4 en charge.

Le développement d'Octopus Viewer est assuré par l'entreprise BibLibre ; créé il y a deux ans seulement, des évolutions importantes de ses fonctionnalités sont sans doute à attendre.

Diva Viewer

Diva Viewer est une adaptation de la visionneuse Diva par Daniel Berthereau.

> ⚠ Sans maintenance active depuis 3 ans, le module présente plusieurs traits d'obsolescence, comme sa **limitation aux médias image** et son **incompatibilité avec les versions d'Omeka S 4.0.0 et plus**.

- Ce module se distingue par son recours à la technologie Ajax, dont le but est d'afficher directement les médias image en résolution maximale. La fenêtre de visualisation est donc prédominante sur la page du Contenu.
- Les métadonnées descriptives et options de téléchargement sont masquées derrière de petites icônes discrètes. Diva propose elle aussi des options de retouche d'image minimale, comme le plugin *Image tools* de Mirador Viewer.
- Initialement dédiée à la visualisation d'images classiques, son extension au protocole IIIF reste limité ; elle ne propose pas de visualisation des annotations, ni de chargement de manifeste IIIF externe.

Tableau de synthèse

Module	Visionneuse polyvalente	Chargement d'un Manifeste IIIF externe	Affichage des Annotations	Gestion de la recherche plein texte	Options de téléchargement	Options de retouche d'image	État de maintenance et de développement du module
Aucun (par défaut dans Omeka)	Non : IIIF seulement	Non	Non	Non	Non	Non	Maintenance et développement actifs
Universal Viewer	Oui: IIIF (dont pdf et ePub) + YouTube	Non	Non	Oui	Oui	Non	Maintenance et développements actifs
Mirador Viewer	Non : IIIF seulement	Oui	Oui	Oui	Oui	Oui	Maintenance et développement actifs
Octopus Viewer	Oui : IIIF + médias bruts (dont pdf)	Oui	Oui (mais sans ajout possible)	Non	Oui	Non	Maintenance et développement actifs
Diva Viewer	Oui : IIIF + médias image bruts	Non	Non	Non	Oui	Oui	Abandonné

Activer une visionneuse :
Une fois l'installation du module effectuée, renseigner la configuration choisie dans les ***Paramètres généraux*** de l'installation Omeka (Il est possible de varier les détails de cette configuration d'un site à l'autre ensuite).

Dans la section ***Administration -> Paramètres*** de chaque site, inscrire alors les paramètres retenus pour chacun ; s'ils ne changent pas, mettre ceux des Paramètres généraux une nouvelle fois.

Dans la section ***Thème -> Configurer les pages de ressources*** de chaque site, décider de la présence et l'emplacement de la visionneuse sur les pages des Contenus, Collections et Médias.

CAS PRATIQUES / RETOURS D'EXPÉRIENCE

Ces « cas pratiques / retours d'expériences » vous proposent des mises en application et des recettes de configuration pour les principaux besoins et services liés à IIIF sur une instance Omeka S.

Ils ont été directement rédigés par des administrateurs et gestionnaires en exercice et reflètent leur pratique concrète des outils, sans viser ni à l'exhaustivité ni à l'unicité des procédures.

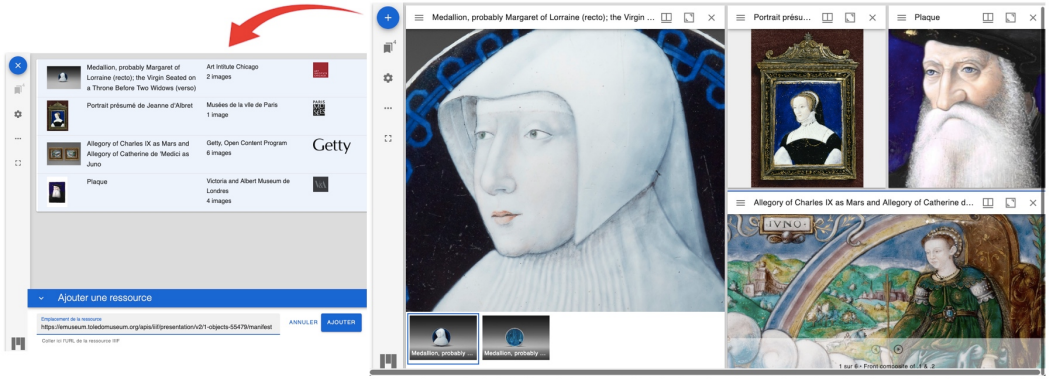
Cas d'usage n° 1 : Omeka S comme agrégateur

Prérequis :

- installation standard de Omeka S

Un agrégateur IIIF est une application dont le rôle est de collecter, d'indexer et de rendre accessibles en un endroit particulier des ressources numériques exposées via IIIF en provenance de multiples sources disséminées.

Cette approche consubstantielle à IIIF est parfaitement mise en œuvre par la visionneuse Mirador qui permet de rapprocher des documents.



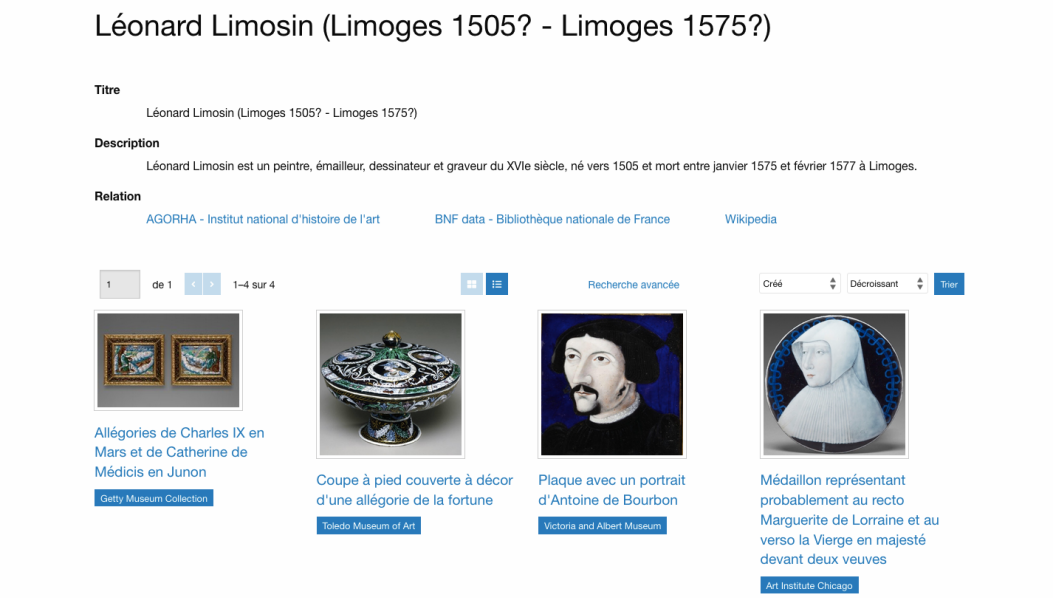
Mirador permet de charger de nouveaux documents via leurs manifestes IIIF et d'agencer leurs fenêtres de visualisation. Par exemple, ici diverses œuvres de ou attribuées à Léonard Limosin, peintre, émailleur, dessinateur et graveur français, né et ayant vécu à Limoges au XVI^{e} siècle.

Dans sa configuration de base, Omeka S peut déjà exploiter ce potentiel de IIIF en permettant de collecter et de stocker manuellement des médias de types « Image IIIF » et « Presentation IIIF » (pour une importation en nombre, se reporter au cas suivant).

Omeka S

- ne dispose pas de moyens pour automatiquement découvrir, indexer et maintenir à jour des listes de ressources IIIF,
- n'héberge pas directement les contenus numérisés.

Une bibliothèque virtuelle basée sur Omeka S référence des ressources IIIF et offre la possibilité d'ajouter ses propres métadonnées à ces ressources. Les métadonnées embarquées dans les Manifestes ne sont pas intégrées dans les champs des items et des collections. Cependant, elles sont affichées par la visualiseuse.



Quelques œuvres de Léonard Limosin de diverses provenances rassemblées dans une collection Omeka.

De nombreux projets de recherche utilisent ainsi Omeka S pour constituer un corpus à partir diverses sources d'archives exposées en IIIF. La gestion des métadonnées dans Omeka S permet alors de décrire ou classifier ces documents de manière interne au projet.

Astuce

L'extension de navigateur detektiiif3 permet d'identifier et récupérer facilement l'URL de Manifestes IIIF au fil de la navigation sur le web.

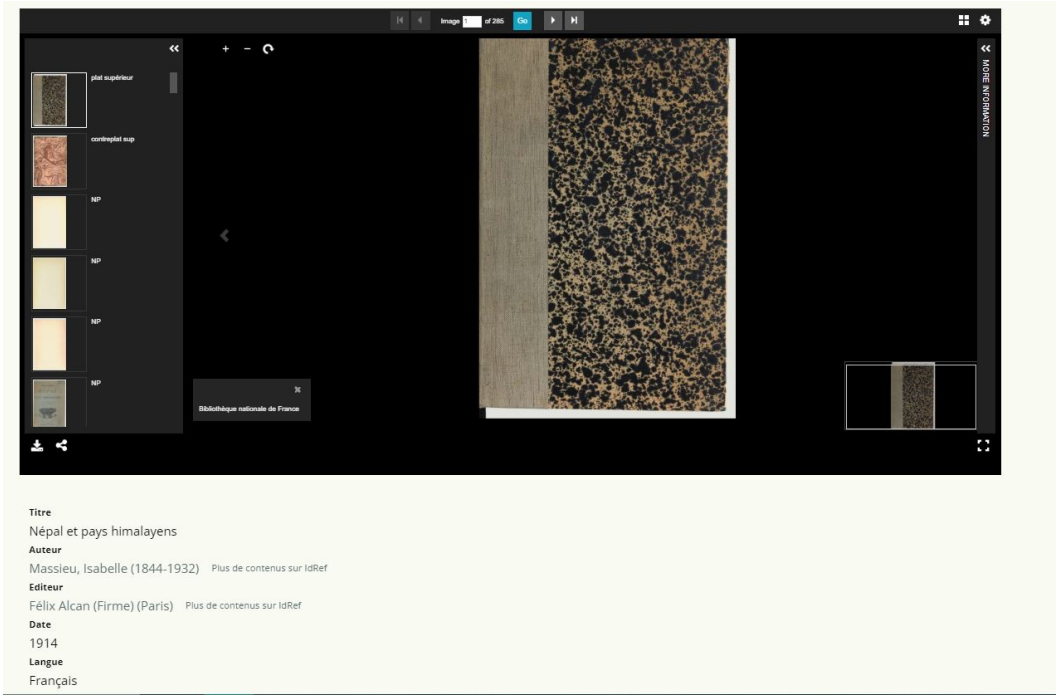
Cas d'usage n° 2 : Intégration, récupération de données

> Prérequis :
>
> - installation Universal Viewer ou Mirador
> - CSV Import
> - IIIF Server

Pour intégrer des Manifestes IIIF externes, plusieurs solutions s'offrent à vous :

I) Intégration du Manifeste IIIF complet depuis une source extérieure par le champ Dublin Core `A un format`

Objectif : obtenir dans votre visionneuse une source extérieure avec vos métadonnées en Dublin Core en dessous :

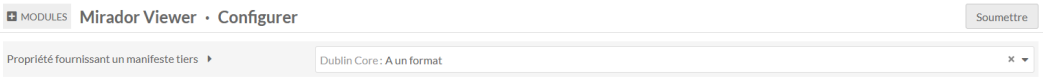


Exemple de visionneuse avec une source extérieure et les métadonnées en Dublin Core en dessous :

1. Paramétrer votre visionneuse en choisissant le champ où vous souhaitez indiquer l'adresse du Manifeste IIIF provenant d'une source exterieure à afficher dans votre visionneuse (dans cet exemple, le champ Dublin Core `A un format` a été choisi pour indiquer le Manifeste IIIF exterieur).

Dans *Universal Viewer* :

Ou dans *Mirador* :



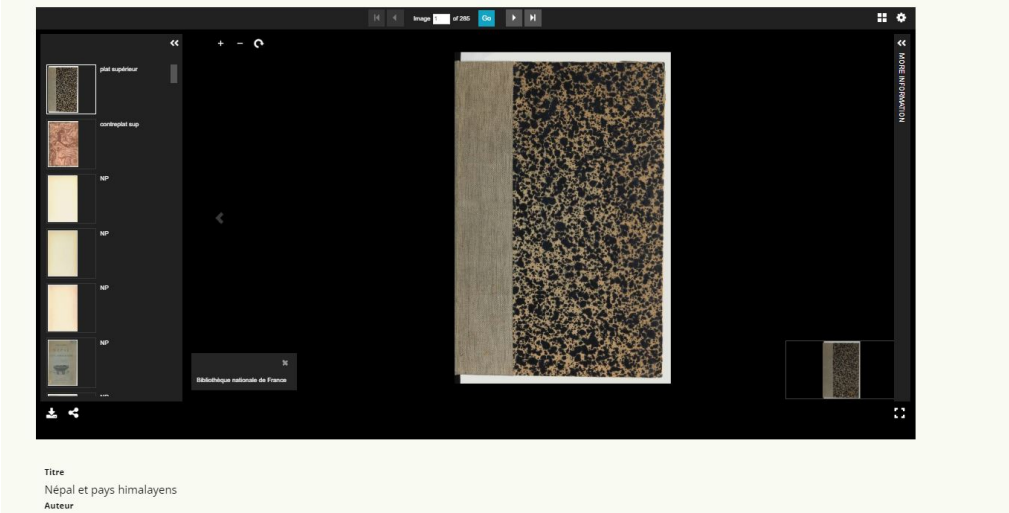
2. Récupérer le Manifeste sur le site source à l'aide du logo

3. Indiquer dans votre item dans la zone choisie pour lire le Manifeste IIIF, l'adresse du Manifeste IIIF de la source tierce à insérer (dans notre exemple dans le champ Dublin Core `A un format`)

4. Votre

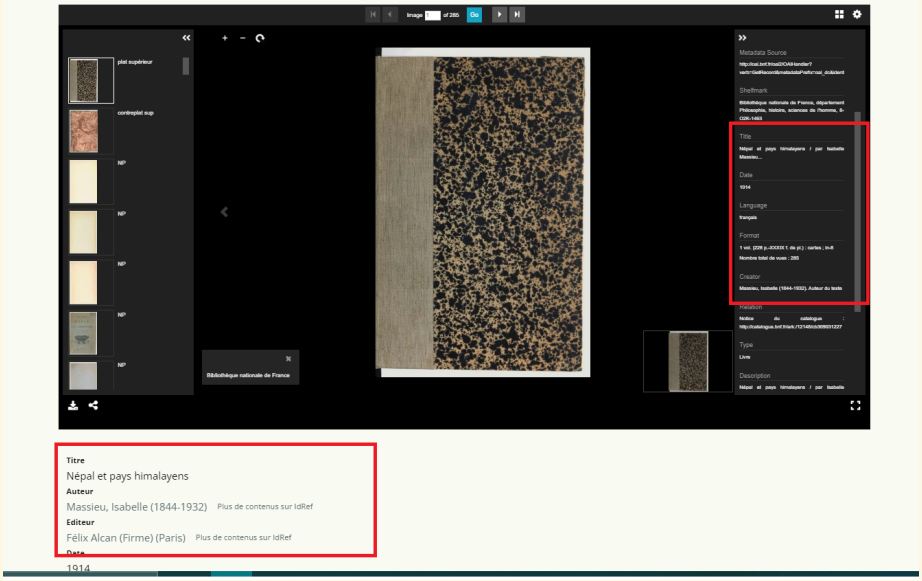
document est visible dans la

visionneuse :



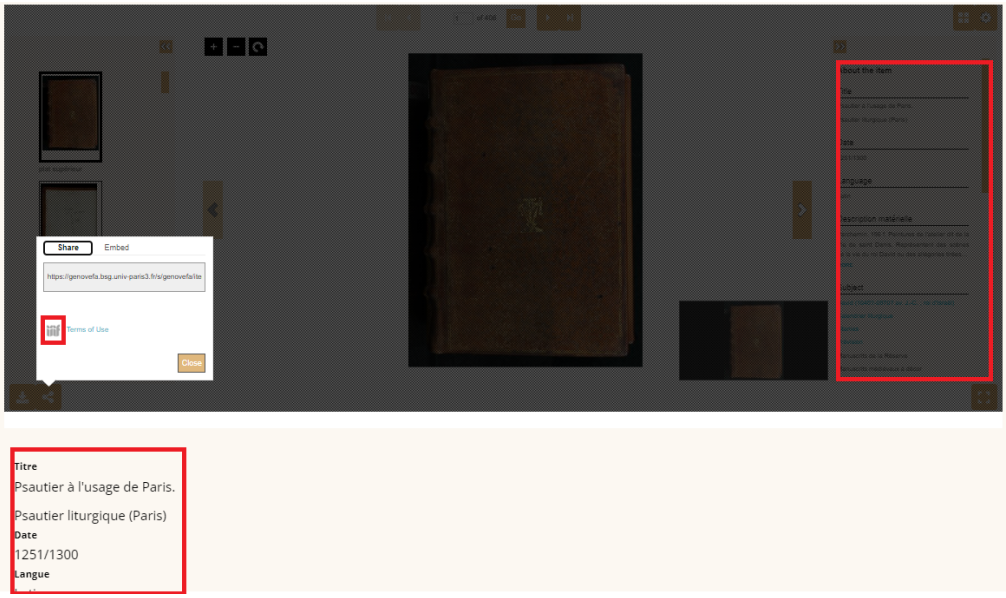
Limites :

- Les métadonnées de votre item (données locales) sont différentes de celles de votre visionneuse et du Manifeste IIIF, ce qui peut créer des problèmes de cohérence d'information sur votre page. En effet, ces deux dernières informations proviennent de la source de votre Manifeste. - Attention, l'ajout d'un Manifeste IIIF via la balise "A un format" ne fonctionne a priori pas pour le moment avec la visionneuse OctopusViewer.



II) Intégration d'URL d'images IIIF au format JSON

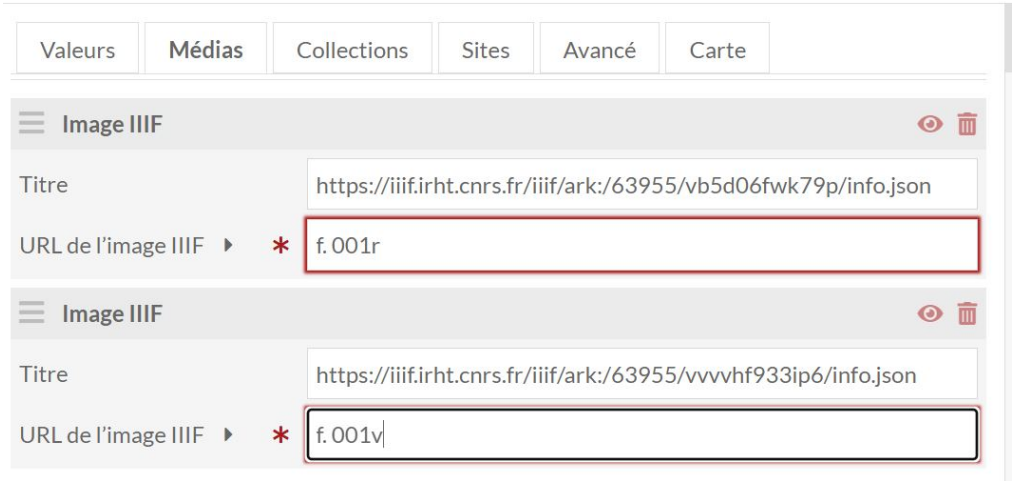
Objectif : obtenir dans votre visionneuse une source extérieure (images sur un serveur tiers IIIF ou sur votre serveur IIIF type Cantaloupe) avec vos métadonnées en Dublin Core dans Omeka, dans la visionneuse et dans le Manifeste IIIF



Méthode 1 : intégration à l'unité

1. Aller dans votre item, onglet media :

2. La page suivante apparaît. Ajouter autant d'image IIIF que vous avez d'images à charger :



Lors de cette étape d'ajout manuel de média, la seule métadonnée qui peut être saisie est un titre. Si vous souhaitez décrire plus en détail chaque fichier, vous pourrez modifier les informations après son téléchargement. Si vous ne fournissez pas de titre, le nom de fichier d'origine apparaîtra comme titre du média.

Vous pouvez télécharger plusieurs fichiers à la fois et ajouter plusieurs types de médias lors de votre enregistrement. Ils apparaîtront dans l'ordre dans lequel ils ont été téléchargés.

Vous pouvez glisser-déposer des fichiers multimédias dans l'ordre de votre choix, lors de leur ajout ou après leur ajout. Sur l'écran d'édition des éléments, cliquez sur l'onglet « Médias » et utilisez les barres horizontales à gauche de chaque fichier pour déplacer les fichiers vers le haut et vers le bas dans la liste.

Méthode 2 : intégration en lot via CSV Import d'images IIIF à rattacher à un item existant

1. Produire un CSV avec les informations suivantes :

	A	B	C	D
1	IIIF	dcterms:title	o:item	Typedia
2	https://iiif.irht.cnrs.fr/iiif/ark:/63955/vmptl7ljguv6/info.json	plat supérieur	BSG_MS62	media
3	https://iiif.irht.cnrs.fr/iiif/ark:/63955/vdphrv70xz90/info.json	contregarde supérieure	BSG_MS62	media
4	https://iiif.irht.cnrs.fr/iiif/ark:/63955/v2uzfg0ecr11/info.json	garde recto	BSG_MS62	media
5	https://iiif.irht.cnrs.fr/iiif/ark:/63955/vln867p02ves/info.json	garde verso	BSG_MS62	media
6	https://iiif.irht.cnrs.fr/iiif/ark:/63955/vd5ol5lkemn8/info.json	f. 001r avec réglet	BSG_MS62	media
7	https://iiif.irht.cnrs.fr/iiif/ark:/63955/vl3pmh1ei3z6/info.json	garde verso	BSG_MS62	media
8	https://iiif.irht.cnrs.fr/iiif/ark:/63955/vb5d06fwk79p/info.json	f. 001r	BSG_MS62	media
9	https://iiif.irht.cnrs.fr/iiif/ark:/63955/vvvvhf933ip6/info.json	f. 001v	BSG_MS62	media
10	https://iiif.irht.cnrs.fr/iiif/ark:/63955/v8boap6dpdpu/info.json	f. 002r	BSG_MS62	media
11	https://iiif.irht.cnrs.fr/iiif/ark:/63955/vqq15mldwb8r/info.json	f. 002v	BSG_MS62	media
12	https://iiif.irht.cnrs.fr/iiif/ark:/63955/v6q7gtbzji7v/info.json	f. 003r	BSG_MS62	media
13	https://iiif.irht.cnrs.fr/iiif/ark:/63955/v6hldvaqk5ga/info.json	f. 003v	BSG_MS62	media
14	https://iiif.irht.cnrs.fr/iiif/ark:/63955/v6mxrzu38m9y/info.json	f. 004r	BSG_MS62	media
15	https://iiif.irht.cnrs.fr/iiif/ark:/63955/vuqs6pbow6ue/info.json	f. 004v	BSG_MS62	media
16	https://iiif.irht.cnrs.fr/iiif/ark:/63955/vr892yu7f5d1/info.json	f. 005r	BSG_MS62	media
17	https://iiif.irht.cnrs.fr/iiif/ark:/63955/v350x88ki7e8/info.json	f. 005v	BSG_MS62	media
18	https://iiif.irht.cnrs.fr/iiif/ark:/63955/vzkva9vh499c/info.json	f. 006r	BSG_MS62	media

a. IIIF : métadonnées des images sur le serveur IIIF externe

b. `Dcterms:title` : titre de l'image. Si vous ne fournissez pas de titre de fichier, le nom de fichier d'origine apparaîtra comme titre du média. Ce nom de fichier peut être modifié à tout moment.

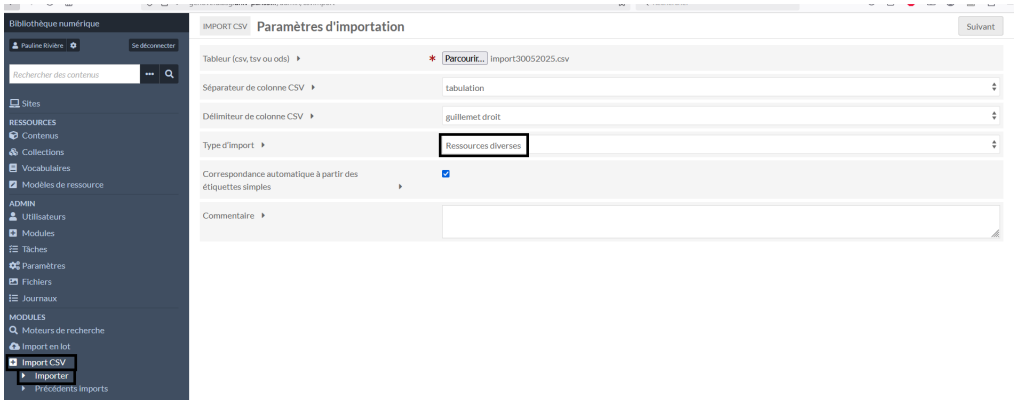
c. o:item : lien vers votre item Omeka

d. Type media : noter media puisque vous importez des medias à rattacher à un item

2. Lancer l'intégration

a. Aller dans CSV Import

b. Charger votre fichier en choisissant des ressources diverses en type d'import



3. Mapper les colonnes

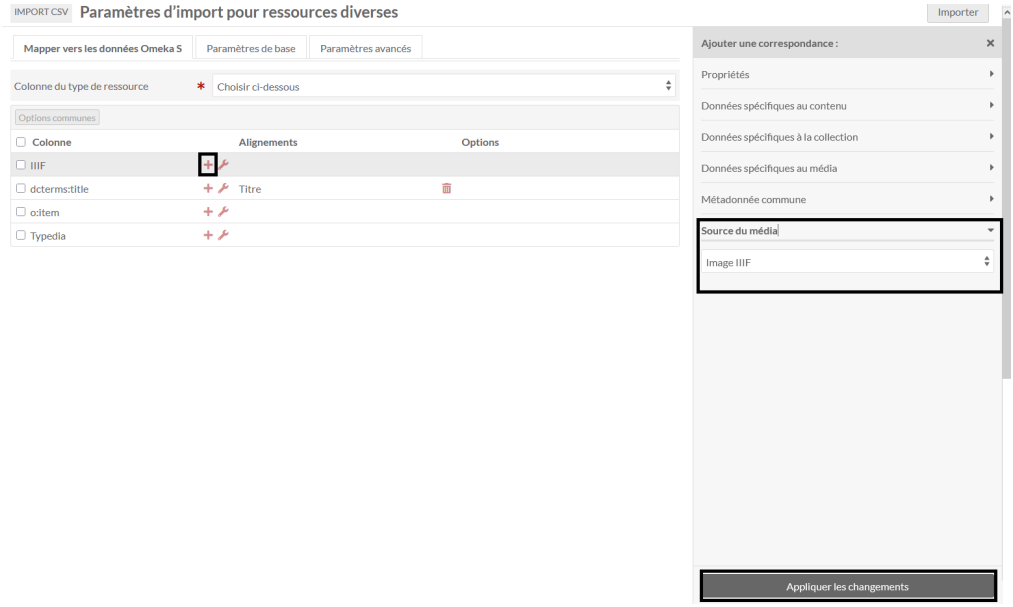
a. IIIF :

- + - puis Source du

média :

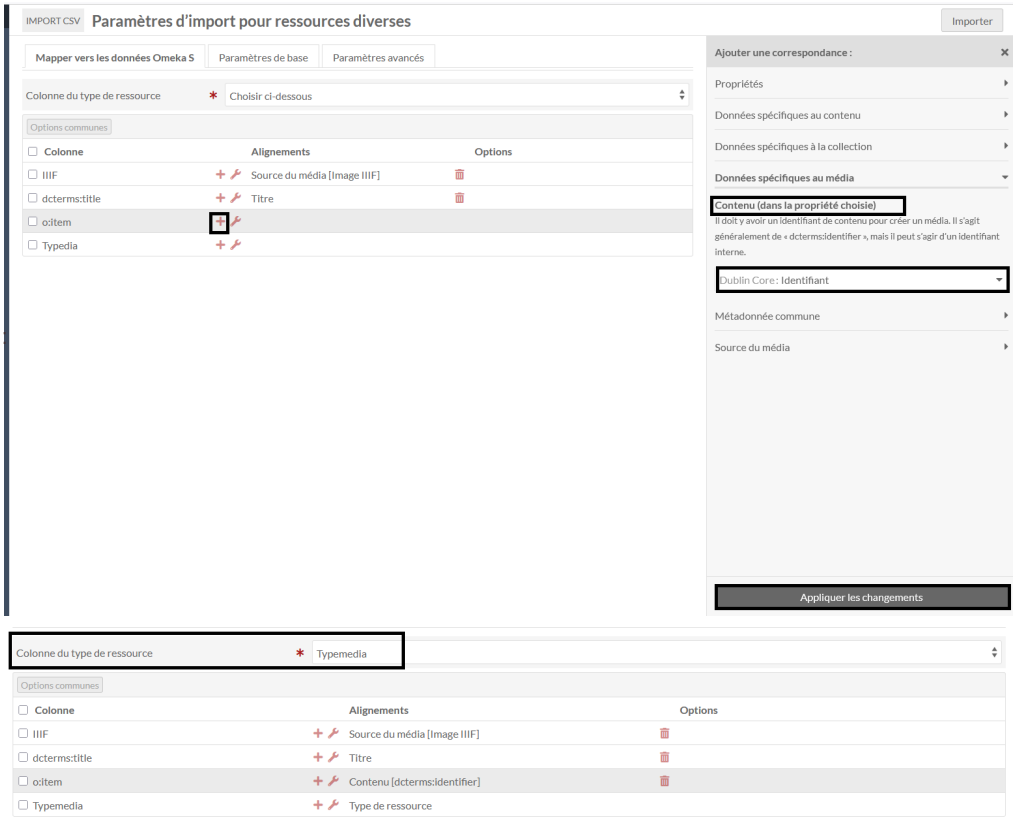
Image IIIF, - puis appliquer chargement!

b. o:item - + - Puis contenu choisir le lien vers le champ pivot avec votre item (dans la capture d'écran dublin core identifiant) - Appliquer les changements

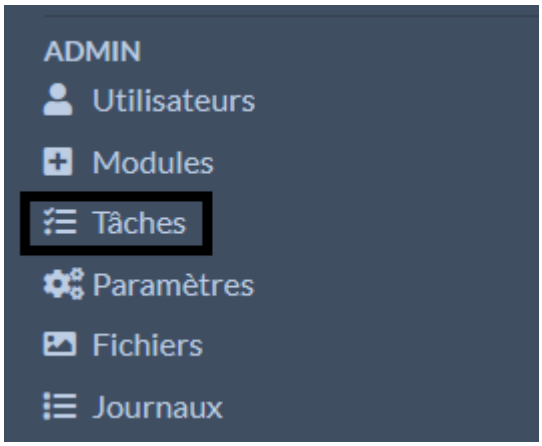


c. Type média :

- colonne du type de ressource : choisir Type Media



4. Lancer l'import. Une tâche s'ouvre.
5. Aller dans le menu tâche. Quand la tâche terminée, cela indique



cela :

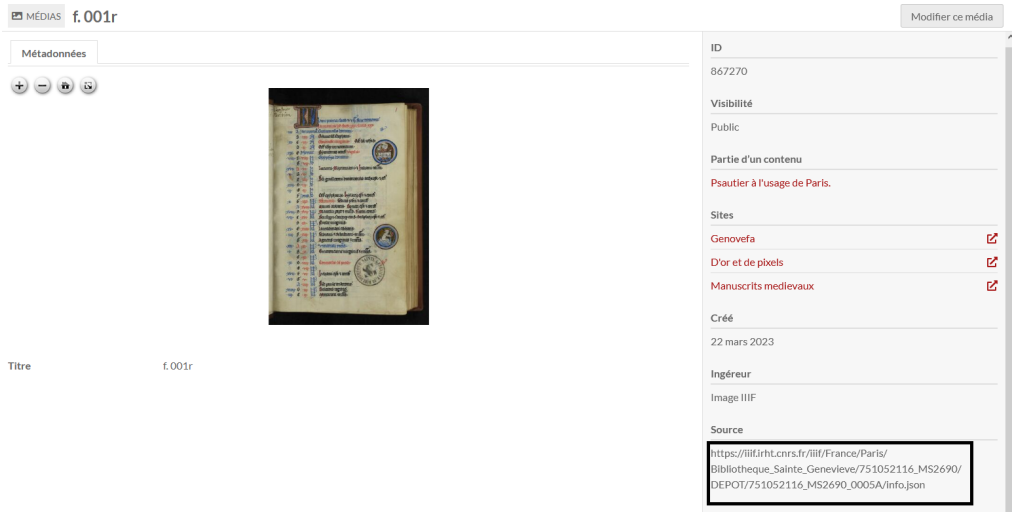
6. Votre item s'affiche avec des médias dans l'ordre où ils apparaissaient dans votre

CSV !

Exemple d'item côté administration avec des images IIIF intégrées. Vous pouvez glisser-déposer les fichiers pour en modifier l'ordre et en modifier le titre etc. Ces médias ont comme source votre serveur d'images

Id / Paramètres	Date	Classe	Statut / Journal	Propriétaire
882	10 juin 2025 à 11:37	CSVImport\Job\Import	Terminé	pauline.riviere@sorbonne-nouvelle.fr

IIIF :



Cas d'usage n° 3 : Recherche plein texte dans une visionneuse

> Prérequis :
>
> - Installation module Universal Viewer ou module Mirador Viewer
> - Module IIIF Search
> - Fichiers de transcription (alto ou tsv)
> - IIIF Server

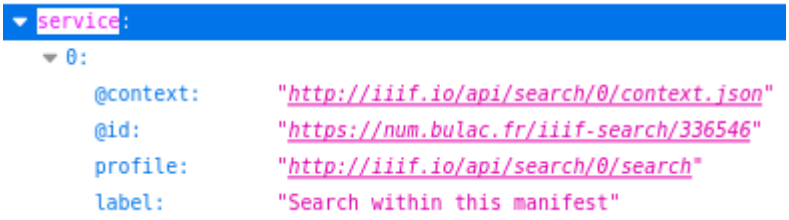
Si vos images IIIF représentent un texte et que vous disposez de ce contenu textuel dans des fichiers de transcription (issus d'un OCR ou d'une transcription automatisée) vous pouvez les associer à vos images et ainsi bénéficier de la recherche plein texte au niveau de la visionneuse publique.

La recherche plein texte est rendue possible par le module **IIIF Search**, développé par Sylvain Machefert et Daniel Berthereau. Ce module interroge les fichiers de transcription texte (*xml-alto* ou *tsv*) importés comme médias.

Son fonctionnement consiste à ouvrir les fichiers, lire le contenu textuel et rassembler toutes les occurrences du terme recherché (avec numéro du canevas et position sur l'image) pour que la visionneuse les affiche dans un volet à part. Pour l'heure, il ne peut traiter qu'une seule expression à la fois ; il n'est pas adapté à la recherche de co-occurrences.

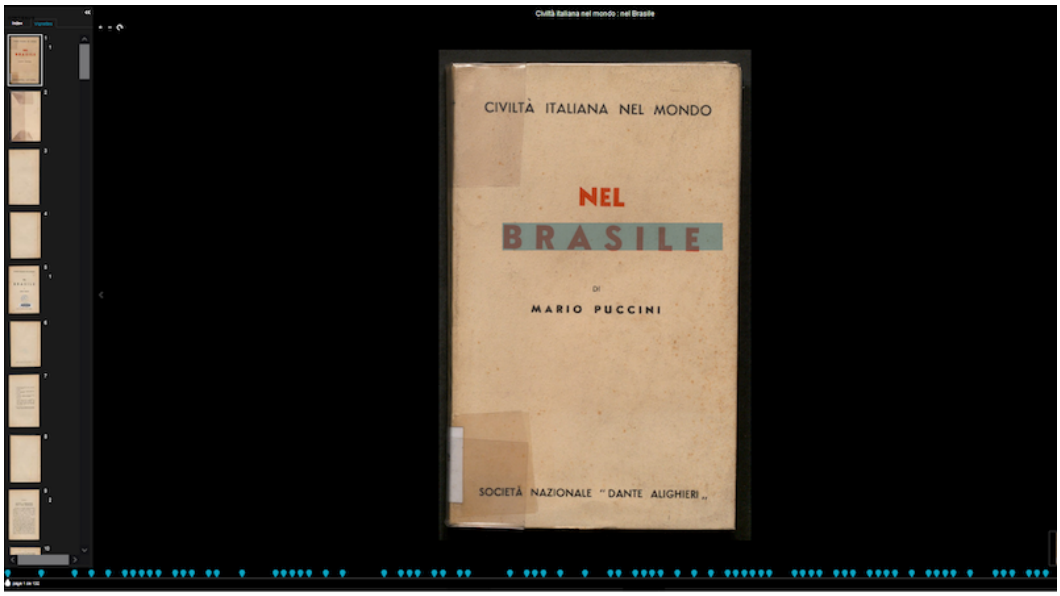
Ce module exploite l'**API IIIF Content Search**, dont la spécification se trouve sur https://iiif.io/api/search/.

Lorsqu'il écrit le manifeste, le module IIIF Server vérifie l'activation de Iiif Search, et, le cas échéant, inscrit une section dédiée à l'API Content-Search dans la section *"service"*:

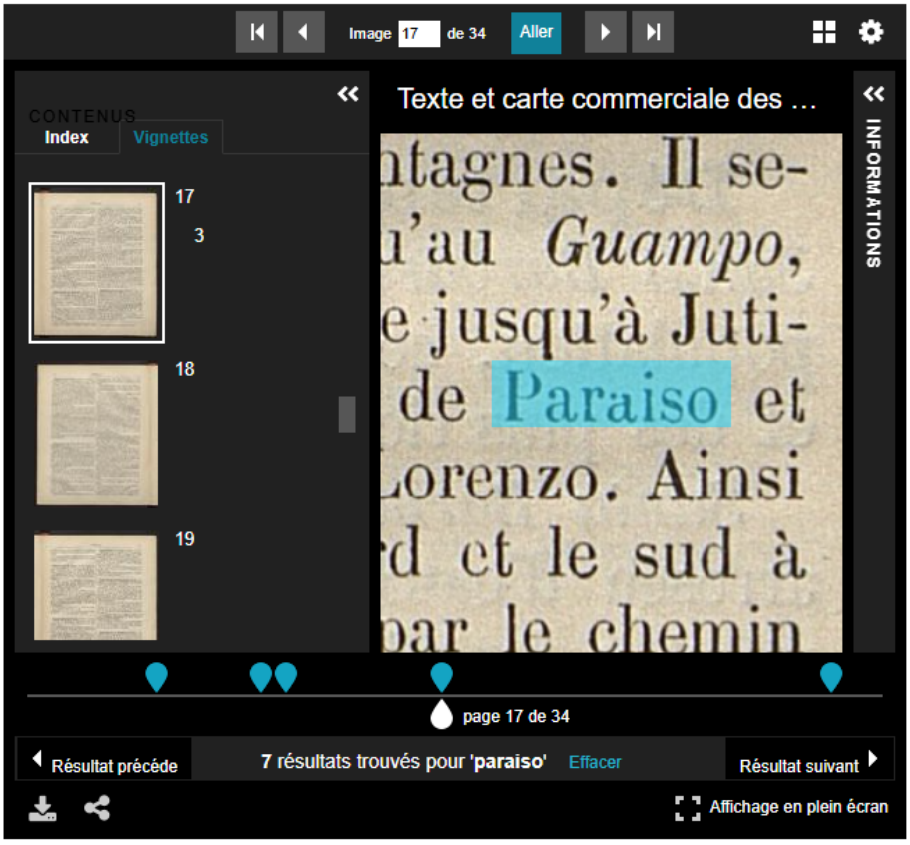


> À
>
> noter :
> Iiif Search ne fonctionne qu'avec des manifestes d'API Presentation 2.

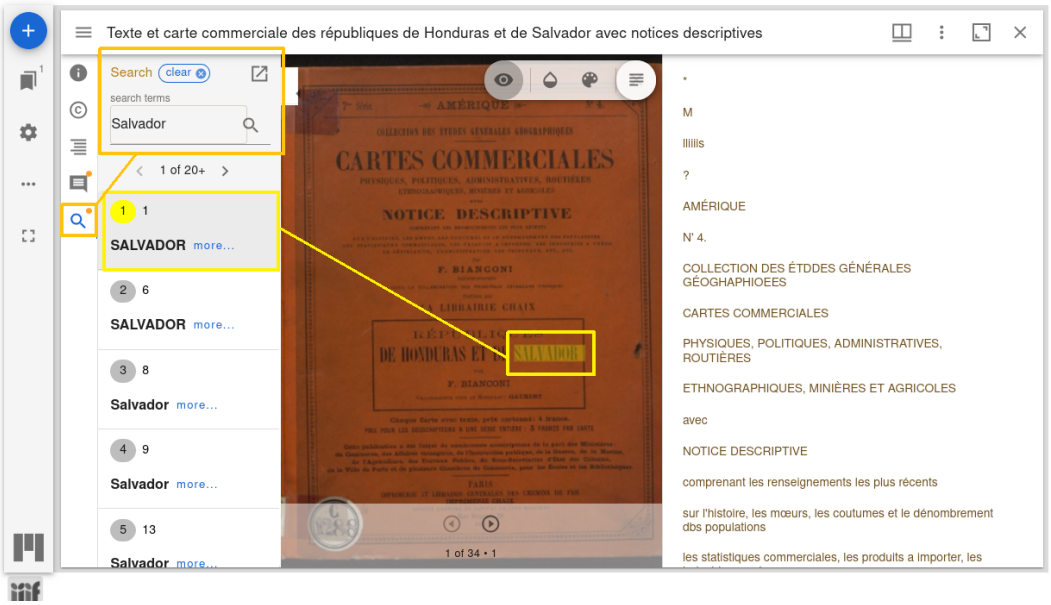
L'option de la recherche plein texte peut se présenter différemment selon les visionneuses :



Iiif Search sur Universal Viewer



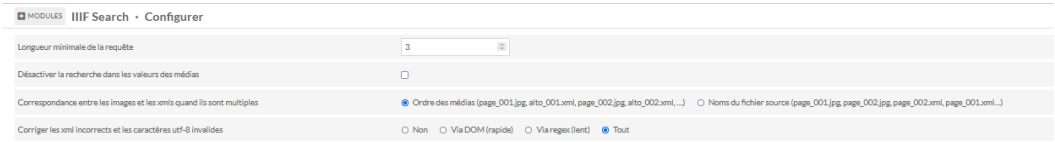
Détail sur Universal Viewer



Iiif Search sur Mirador

Iiif Search accepte trois cas de figure pour les médias : 1. 1 fichier xml-alto par page ; importez-les comme médias avec un nom identique aux fichiers image, extension exceptée. 2. 1 fichier xml-alto pour l'ensemble du document ; 3. 1 fichier tsv pour l'ensemble du document.

Dans tous les cas, la correspondance entre fichier image et fichier texte doit être renseignée dans le paramétrage du module.



Si le manifeste IIIF a été créé ailleurs, puis importé dans votre installation (voir cas d'usage n°2), la recherche plein texte fonctionnera à condition que ce manifeste ait été associé à l'API IIIF Content Search, que ce soit via le module IiifSearch, ou tout autre service. Pour le vérifier rapidement, regarder si cette API est mentionnée au premier niveau du Manifeste.

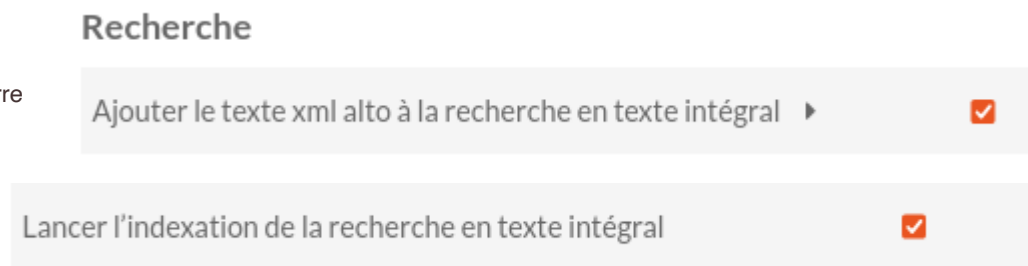
Recherche plein texte hors de la visionneuse

La recherche décrite précédemment s'appuie sur les possibilités de IIIF mais reste cantonnée au document et à sa visionneuse. Si vous souhaitez proposer une recherche plein texte sur vos ressources depuis les moteurs de recherche de votre instance, vous devez paramétrer ces derniers.

Avec le moteur de recherche d'Omeka

Il est possible d'intégrer un fichier alto ou tsv à l'indexation générale depuis les **Paramètres généraux** de l'installation, en cochant les deux cases suivantes :

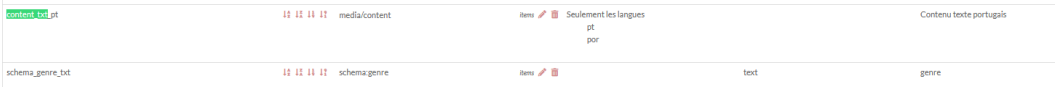
Dès lors, si l'expression entrée en barre de recherche générale est identifée



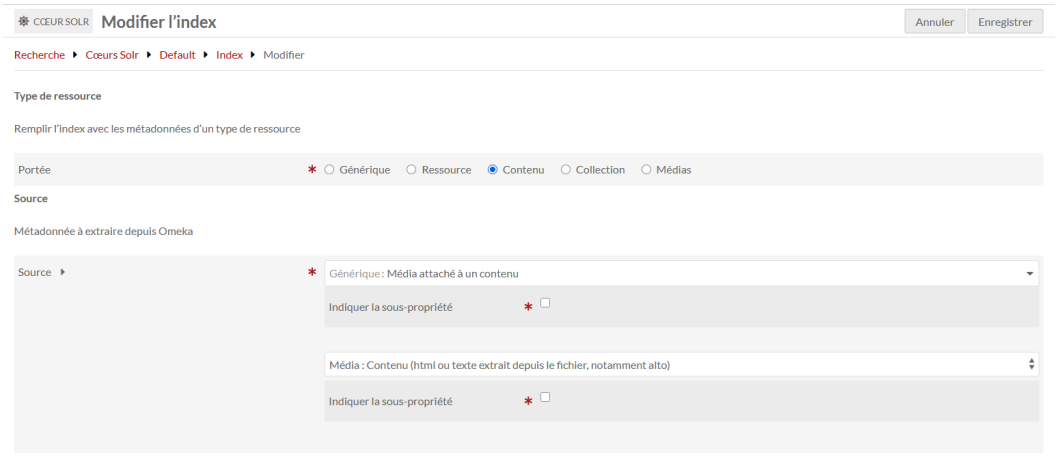
Omeka ne fournit qu'un rebond sur le document concerné dans son intégralité, et non directement sur la page où l'expression apparaît.

Avec Solr

Solr permet de définir un index alimenté par le contenu des fichiers alto stockés dans les médias. Voici le champ défini dans la liste des index, ici "Content":



Et l'interface de paramétrage de l'index (accessible par l'icône "crayon"), ici, le contenu du champ est rempli à partir des fichiers Alto :



Cas d'usage n° 4 : Intégrer un contenu textuel à un Manifeste IIIF

Prérequis :

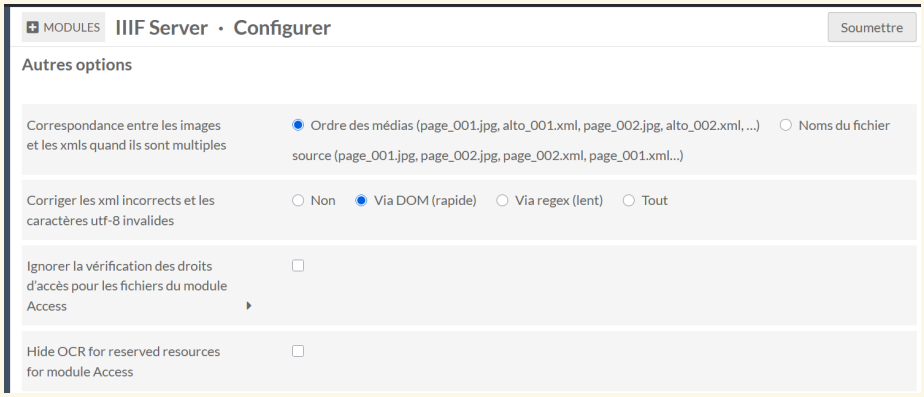
- IIIF Server
- Médias xml-alto (version 4 et plus)

Si vous disposez de fichiers de transcription (issus d'un OCR ou d'une transcription automatisée) que vous avez associés à vos images, leur contenu peut être également embarqué dans le Manifeste IIIF dans une section particulière. Il n'est en effet pas recommandé d'intégrer les contenus de textes transcrits au niveau des métadonnées génériques du document, qui doivent rester purement descriptives.

Cette option ne concerne que les médias au format **xml-alto**, et dépend du module **IIIF Server**.

Bon encodage Par ailleurs, nous vous recommandons également de veiller au bon encodage du contenu dans les xml-alto, en UTF-8, afin qu'il soit correctement rendu par différents systèmes. Une coche au niveau du paramétrage d'IIIF Server permet de s'en

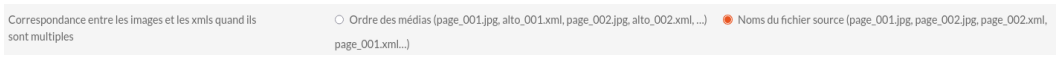
assurer :



Capture d'écran du menu de configuration d'IIIF Server.

Idéalement, à chaque fichier image (contenant du texte) doit correspondre un fichier xml-alto, et leurs noms doivent être identiques, à l'exception de l'extension fichier. Exemple: *cote_document_0002.jpg* et *cote_document_0002.xml* pour la page 2 du document.

La correspondance entre images et fichiers xml-alto doit aussi être précisée dans les paramètres de ce module :



Sur cet
exemple, les xml-alto ont tous été importés après les fichiers image : la correspondance ne peut s'effectuer que sur leur nom.

Le module se charge d'ouvrir et de lire chaque fichier (opération similaire à celle du module Iiif Search). Lors de la génération du manifeste de la ressource, IIIF Server *ajoute* une annotation pour chaque ligne de texte

identifiée dans le fichier xml-alto. Chaque annotation contient le texte sous forme de chaîne de caractères brute, ainsi que sa position sur l'image. ::: info Si l'on utilise également la visionneuse **Mirador** et son plugin *Text Overlay*, ce contenu peut être affiché en surbrillance, ainsi qu'en regard de l'image dans un volet latéral.

:::

Les annotations sont regroupées en une liste, que l'on peut retrouver dans la section "otherContent" de chaque canevas. Contrairement à Iiif Search, IIIF Server permet ainsi d'enregistrer durablement le contenu textuel dans le manifeste.

Exemple de canevas avec transcription intégrée. Le détail des annotations s'obtient en cliquant sur leur URI.

> À
>
> noter :
> IIIF Server ne propose cette option que pour l'API Presentation 2.

> **Obtenir des fichiers xml-alto à partir de pdf** Si le document comprend un seul fichier pdf pour son ensemble, le module **Extract Ocr** est en mesure de fournir un unique fichier alto et/ou tsv. Toutefois, il ne peut pas proposer un fichier xml-alto par page.
>
> Si le document comprend un fichier pdf par page et que le module Extract Ocr ne fonctionne pas, il est possible d'utiliser le programme *linux pdfalto* pour obtenir chaque fichier xml-alto correspondant. La page Github d'installation se trouve ici : https://github.com/kermitt2/pdfalto.

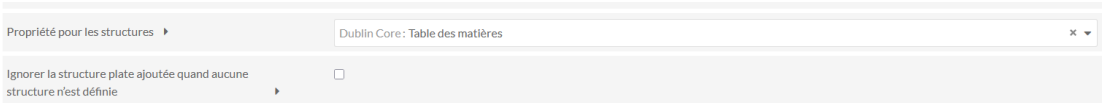
Cas d'usage n° 5 : Ajouter une table des matières

> Prérequis :
>
> - IIIF Server
> - Visionneuse au choix : Universal Viewer, Mirador, Octopus Viewer

Par défaut, IIIF présente les médias (images, pages, etc.) dans l'ordre où ils sont listés. Pour créer une structure plus complexe (chapitres, sections, illustrations, etc.), il faut renseigner une propriété spécifique, soit en JSON, soit sous forme littérale, suivant un format textuel ligne par ligne. Dans un Manifeste IIIF, c'est donc la section *"structure"* qui sert à organiser, décrire et hiérarchiser les contenus (pages, chapitres, illustrations, etc.) d'un document afin de permettre une navigation structurée dans les visionneuses compatibles IIIF.

Principe général

Plus concrètement, dans Omeka S, la table des matières doit être renseignée dans le champ Dublin Core *TableOfContent* suivant un formalisme défini dans la documentation du module IIIF Server et rappelé ici. Dans le paramétrage du modue IIIF Server, on doit ensuite définir l'association de la balise IIIF Structure au champ TableOfContent.



C'est le module IIIFServer qui transforme le formalisme de la table des matières en format JSON et qui alimente ensuite la rubrique "structure" du Manifeste.

Description du formalisme Omeka de la table des matières :

Chaque ligne du format représente une section (ou *range*) de la structure :

```
{id}, {label}, {canvasIndexOuRangeId1}; {canvasIndexOuRangeId2}; …; {canvasIndexOuRangeIdN}
```

Voyons un exemple d'une ligne à quatre champs avec une référence à la hierarchie des titres : le chapitre II de niveau titre 2 (r1-8) contient 8 sous-sections numérotées r1-8-1 à r1-8-8, ce qui s'écrit :

```
r1-8, Chapitre II : La preuve cartographique du Brésil, 65-124, r1-8-1;r1-8-2;r1-8-3;...r1-8

ou plus simplement : r1-8, Chapitre II : La preuve cartographique du Brésil, 65-124, r1-8-1/
```

Détails sur les identifiants et la syntaxe

- Identifiant

  (id) :
  Il doit être unique, alphanumérique, sans espace, accent ou caractère spécial (pour garantir la stabilité des URI). Il ne doit
  Si l'identifiant est omis, la ligne reçoit un identifiant automatique basé sur son numéro d'ordre, mais cela n'est pas recomm
  changerait si l'ordre change.

  ```
  Exemple ici : r1-8
  ```

- Label

  (titre) :
  Il s'agit du titre de la section. S'il est vide, la section sera utilisée dans la structure mais ne s'affichera pas dans la table de

  ```
  Exemple : Un titre affiché dans la table des matières (label) : Chapitre II, Plat supérieur, vu
  ```

- Liste des

  contenus :
  Ce sont les index des pages (*canvases*) ou les identifiants de sous-sections incluses dans la section.

  Attention :
  l'index de la page correspond à la position dans la liste IIIF, qui peut différer de l'ordre des médias dans Omeka ou un autr

  ```
  Exemple : Une liste d'identifiants de canvas de référence : 65-124
  ```

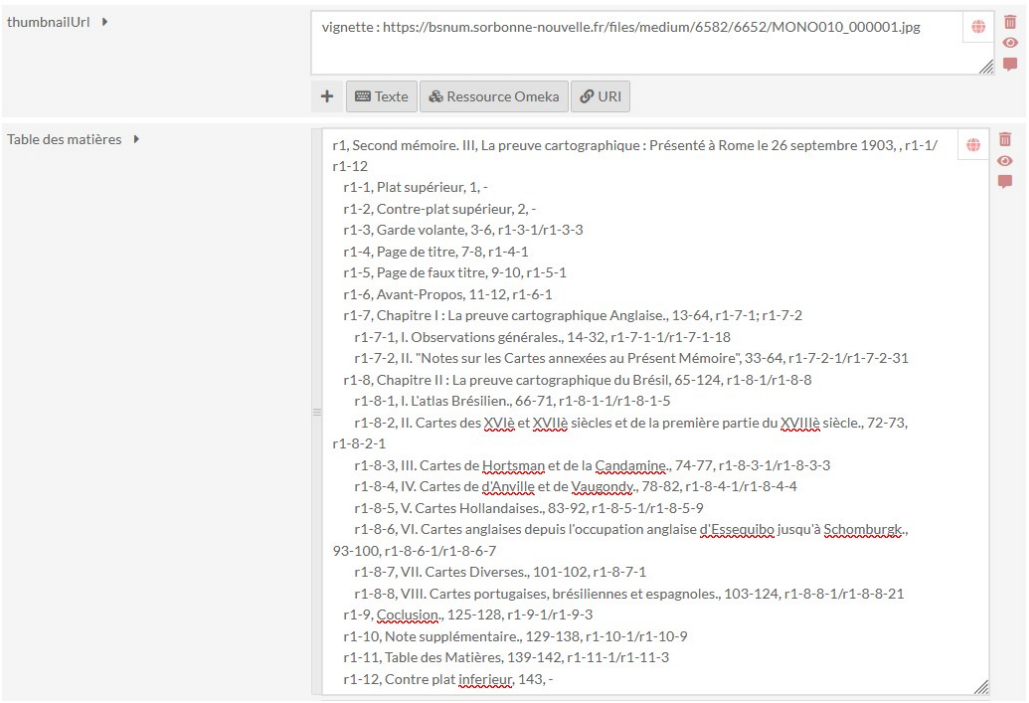
- L'identification de la structure hierarchique des sections et sous-sections, séparées par des ";"

  ```
  Exemple : une liste d'éléments contenus (sous-sections) quand la hierarchie est conservée (ligne
    r1-8-1;r1-8-2;r1-8-3;...r1-8-8
       ou r1-8-1/r1-8-8
  ```

Hiérarchies complexes

Le format de lignes à quatre champs permet donc de décrire des structures très hiérarchisées (chapitres, sections, sous-sectio
imbriquant les identifiants dans la colonne des contenus en utilisant le quatrième champ. Ce bloc de lignes affiche la hiérarchie
possibilité de déplier les sous-sections à l'aide d'un clic de souris sur le symbole + accolé au niveau du titre supérieur :

Exemple de hiérarchie complexe :


thumbnailUrl
vignette : https://bsnum.sorbonne-nouvelle.fr/files/medium/6582/6652/MONO010_000001.jpg
Texte
Ressource Omeka
URI
Table des matières
r1, Second mémoire. III, La preuve cartographique : Présenté à Rome le 26 septembre 1903, , r1-1/r1-12
r1-1, Plat supérieur, 1, -
r1-2, Contre-plat supérieur, 2, -
r1-3, Garde volante, 3-6, r1-3-1/r1-3-3
r1-4, Page de titre, 7-8, r1-4-1
r1-5, Page de faux titre, 9-10, r1-5-1
r1-6, Avant-Propos, 11-12, r1-6-1
r1-7, Chapitre I : La preuve cartographique Anglaise., 13-64, r1-7-1; r1-7-2
r1-7-1, I. Observations générales., 14-32, r1-7-1-1/r1-7-1-18
r1-7-2, II. "Notes sur les Cartes annexées au Présent Mémoire", 33-64, r1-7-2-1/r1-7-2-31
r1-8, Chapitre II : La preuve cartographique du Brésil, 65-124, r1-8-1/r1-8-8
r1-8-1, I. L'atlas Brésilien., 66-71, r1-8-1-1/r1-8-1-5
r1-8-2, II. Cartes des XVIè et XVIIè siècles et de la première partie du XVIIIè siècle., 72-73, r1-8-2-1
r1-8-3, III. Cartes de Hortsman et de la Candamine., 74-77, r1-8-3-1/r1-8-3-3
r1-8-4, IV. Cartes de d'Anville et de Vaugondy., 78-82, r1-8-4-1/r1-8-4-4
r1-8-5, V. Cartes Hollandaises., 83-92, r1-8-5-1/r1-8-5-9
r1-8-6, VI. Cartes anglaises depuis l'occupation anglaise d'Essequibo jusqu'à Schomburgk., 93-100, r1-8-6-1/r1-8-6-7
r1-8-7, VII. Cartes Diverses., 101-102, r1-8-7-1
r1-8-8, VIII. Cartes portugaises, brésiliennes et espagnoles., 103-124, r1-8-8-1/r1-8-8-21
r1-9, Coclusion., 125-128, r1-9-1/r1-9-3
r1-10, Note supplémentaire., 129-138, r1-10-1/r1-10-9
r1-11, Table des Matières, 139-142, r1-11-1/r1-11-3
r1-12, Contre plat inferieur, 143, -

La hiérarchisation permet de dérouler les niveaux de titres, mais seul actuellement le dernier niveau est

cliquable :
les niveaux supérieurs et intermédiaires ne le sont pas, alors que la table des matières devrait permettre une navigation structurée dans l'ensemble des documents quelque soit le niveau de titre. Le rendu dans le champ Index de la visionneuse est le

suivant :
ici seul le niveau 2 est cliquable et permet de visualiser le titre correspondant, ce qui n'est pas possible pour le titre de niveau

1 :

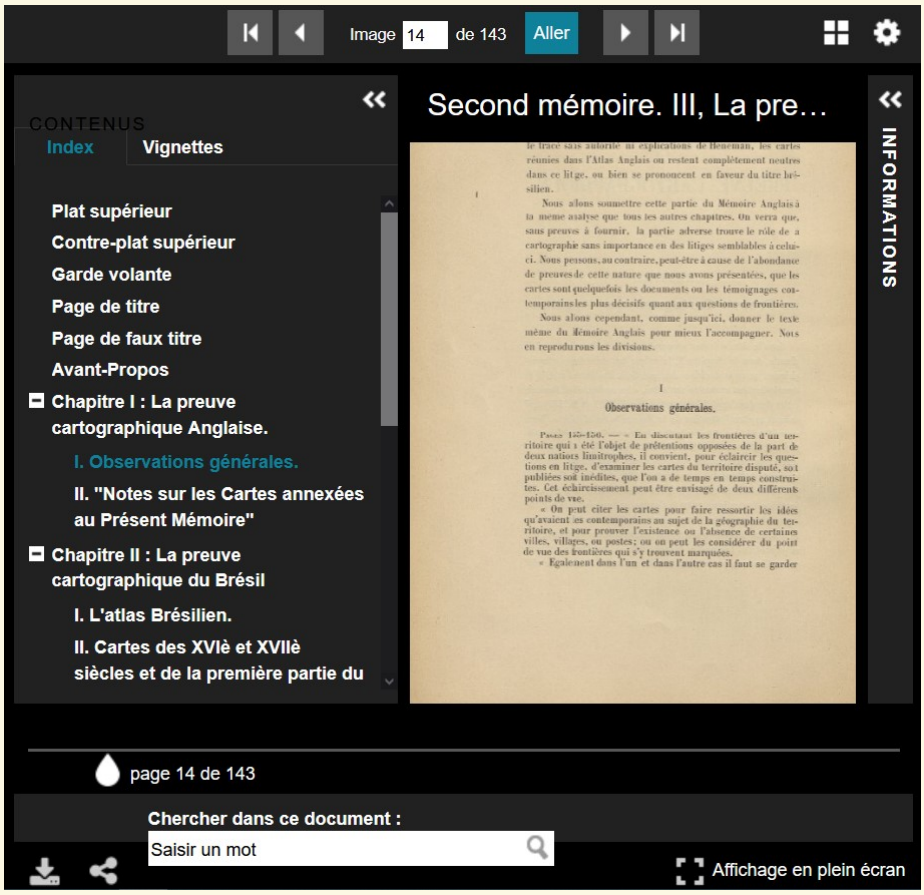
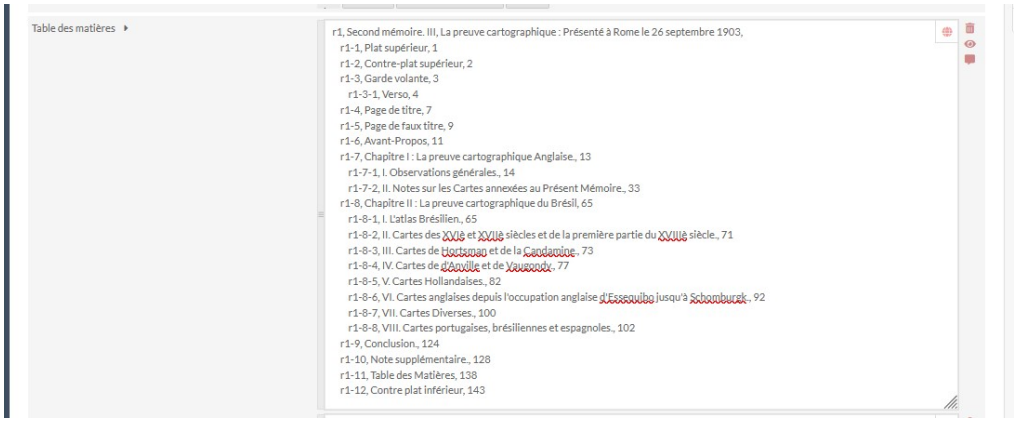


Table des matières sans hiérarchie des titres

- Voici maintenant un exemple d'un bloc de lignes à trois champs. L'ensemble est inséré dans la balise dublin core TableOfContent. La liste des sous-sections n'est pas précisée. La hiérarchie des titres, visible dans les identifiants, ne s'affiche donc pas dans la

  visionneuse :
  tous les niveaux de titres sont alignés les uns sous les autres. Aucun symbôle de dépliement de sous-bloc n'apparait.



- L'affichage correspondant de la table des matières dans l'index de la visionneuse est le

  suivant :
  tous les titres sont cliquables et chaque clic permet d'accéder à la vue correspondante.

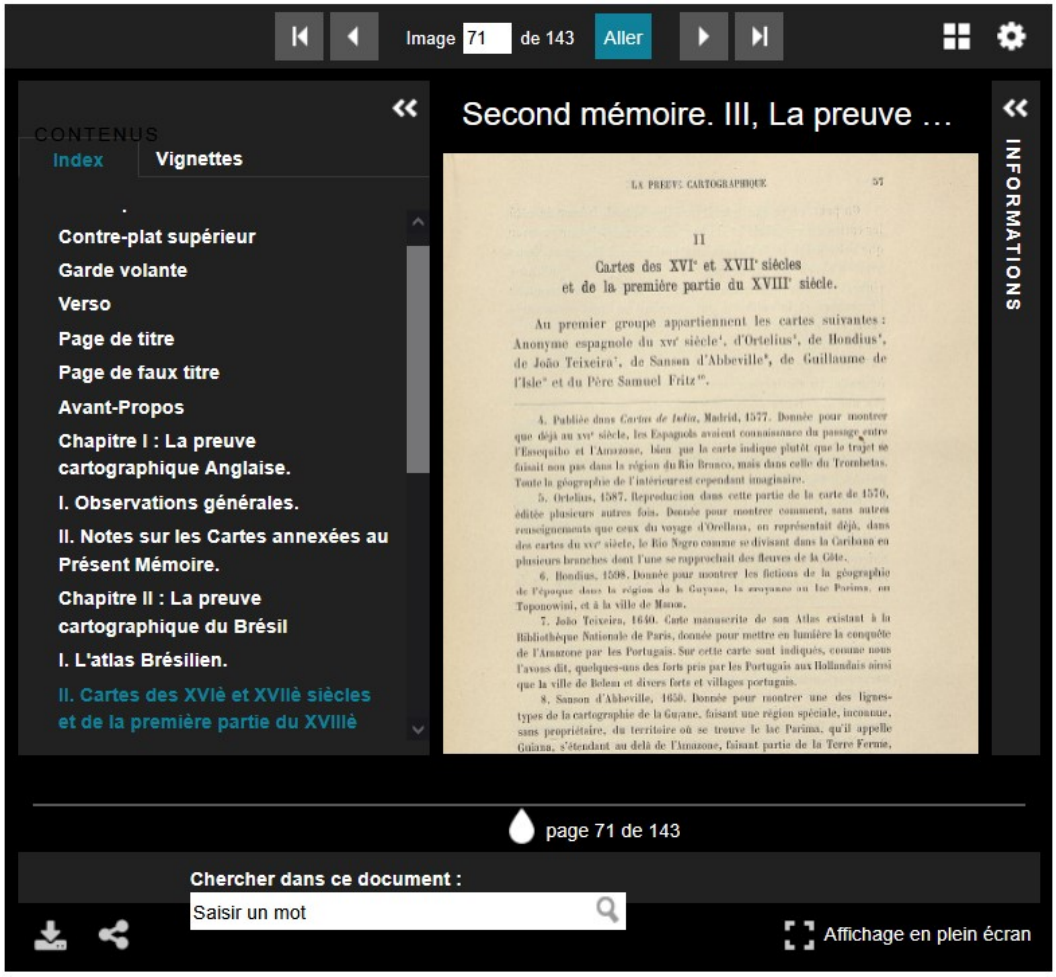


Table des matières non hiérarchisée dans Universal Viewer

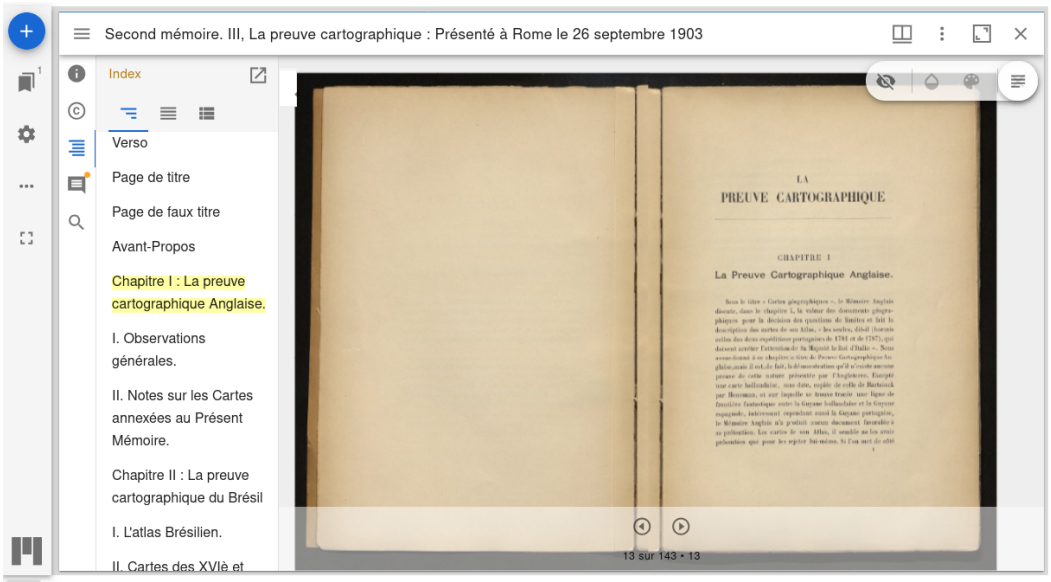
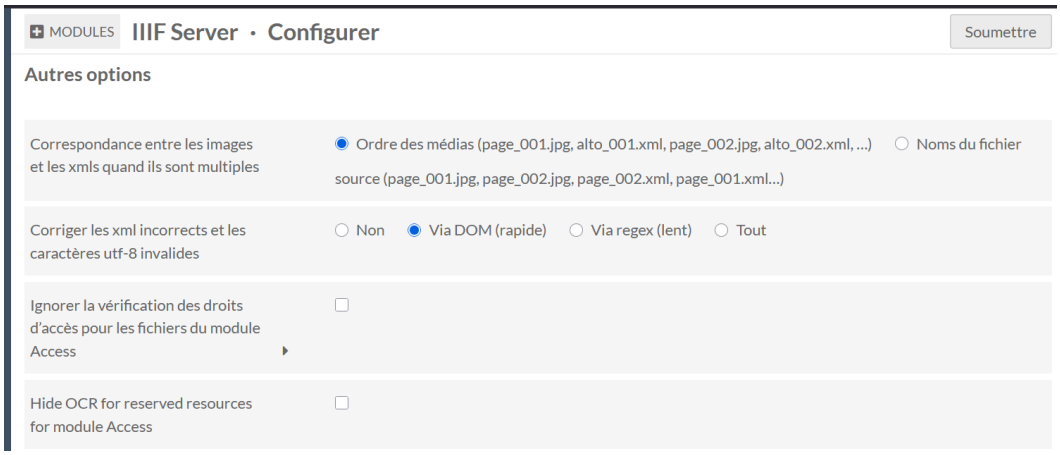


Table des matières non hiérarchisée dans Mirador

Format XML et JSON

La structure est convertie en JSON pour IIIF, chaque section devenant un objet avec ses sous-sections ou pages listées dans des propriétés *ranges* ou *canvases*. Le tout est ajouté dans la rubrique “structure” du Manifeste (ici, un exemple en JSON IIIF v2) :


MODULES
IIIF Server · Configurer
Soumettre
Autres options
Correspondance entre les images et les xmls quand ils sont multiples
Ordre des médias (page_001.jpg, alto_001.xml, page_002.jpg, alto_002.xml, ...)
Noms du fichier source (page_001.jpg, page_002.jpg, page_002.xml, page_001.xml...)
Corriger les xml incorrects et les caractères utf-8 invalides
Non
Via DOM (rapide)
Via regex (lent)
Tout
Ignorer la vérification des droits d'accès pour les fichiers du module Access
Hide OCR for reserved resources for module Access

```
structures": [

    {
          "@id": "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/range/r1-1",
          "@type": "sc:Range",
          "label": "Plat supérieur",
          "canvases": [
                "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/canvas/p1"
          ]
    },
    {
          "@id": "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/range/r1-2",
          "@type": "sc:Range",
          "label": "Contre-plat supérieur",
          "canvases": [
                "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/canvas/p2"
          ]
    },
    {
          "@id": "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/range/r1-3",
          "@type": "sc:Range",
          "label": "Garde volante",
          "canvases": [
                "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/canvas/p3"
          ]
    },
    {
          "@id": "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/range/r1-3-1",
          "@type": "sc:Range",
          "label": "Verso",
          "canvases": [
                "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/canvas/p4"
          ]
    },
    {
          "@id": "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/range/r1-4",
          "@type": "sc:Range",
          "label": "Page de titre",
          "canvases": [
                "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/canvas/p7"
          ]
    },
    {
          "@id": "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/range/r1-5",
          "@type": "sc:Range",
          "label": "Page de faux titre",
          "canvases": [
                "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/canvas/p9"
          ]
    },
    {
          "@id": "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/range/r1-6",
          "@type": "sc:Range",
          "label": "Avant-Propos",
          "canvases": [
                "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/canvas/p11"
          ]
    },
    {
          "@id": "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/range/r1-7",
          "@type": "sc:Range",
          "label": "Chapitre I : La preuve cartographique Anglaise.",
          "canvases": [
                "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/canvas/p13"
```

```
        },
        {
              "@id": "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/range/r1-7-1",
              "@type": "sc:Range",
              "label": "I. Observations générales.",
              "canvases": [
                    "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/canvas/p14"
              ]
        },
        {
              "@id": "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/range/r1-7-2",
              "@type": "sc:Range",
              "label": "II. Notes sur les Cartes annexées au Présent Mémoire.",
              "canvases": [
                    "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/canvas/p33"
              ]
        },
        {
              "@id": "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/range/r1-8",
              "@type": "sc:Range",
              "label": "Chapitre II : La preuve cartographique du Brésil",
              "canvases": [
                    "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/canvas/p65"
              ]
        },
        {
              "@id": "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/range/r1-8-1",
              "@type": "sc:Range",
              "label": "I. L'atlas Brésilien.",
              "canvases": [
                    "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/canvas/p65"
              ]
        },
        {
              "@id": "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/range/r1-8-2",
              "@type": "sc:Range",
              "label": "II. Cartes des XVIè et XVIIè siècles et de la première partie du XVIIIè si
              "canvases": [
                    "https://bsnum.sorbonne-nouvelle.fr/iiif/2/6652/canvas/p71"
              ]
        },
```

Cas d'usage n° 6 : Créer des ressources à partir de Nakala

Prérequis :
- Données Nakala au format Image - Module *CSV Import*

Nakala est un entrepôt de données spécialisé en SHS et géré par l'IR* Huma-Num. Il permet de constituer des *collections* qui regroupent des *données*. Ces données, décrites par des métadonnées, peuvent elles-mêmes être constituées de un ou plusieu *fichiers* (des images aux formats TIFF, JPG, des PDF, des fichiers tabulés ou encodés, etc.). Une fois publiées, ces données se attribuer un identifiant pérenne DOI et ne sont plus supprimables (mais versionnables).

L'entrepôt Nakala intègre un serveur IIIF Cantaloupe qui repose sur l'API Image, comme détaillé dans le Cas d'usage n° 7. Seu API Image est disponible, et il n'exite pas de services complémentaires à l'heure actuelle pour la génération automatique de manifestes. Les fichiers images peuvent donc être appelés un à un, mais ne sont pas accessibles comme une unité documenta réunissant plusieurs fichiers, accompagnés de leurs métadonnées, et interrogeables via un manifeste IIIF Présentation.

Seuls les fichiers dont les types MIME correspondent à des formats d'images seront servis via le protocole

IIIF :
image/jpeg, image/tiff, image/png, image/jp2. Le type application/pdf ne renverra au mieux que la première page du document.

Voir à ce sujet Formats d'image et processeurs d'images.

Récupération des métadonnées et création de l'item Omeka

Si vous souhaitez intégrer à votre bibliothèque numérique Omeka un ensemble de fichiers publiés sur Nakala, il faut préalabler créer un item pour lui associer des médias.

Si besoin, les métadonnées documentaires d'une collection ou d'une donnée Nakala peuvent être consultées et exporter selon méthodes: - en moissonnant l'entrepôt OAI-PMH de Nakala, dans lequel les métadonnées sont structurées en XML. :::info :::sp Exemples de requêtes OAI-PMH Pour une collection: `https://api.nakala.fr/oai2?verb=ListRecords&metadataPrefix=qdc&set={identifiant de la collection}` Pour u donnée: `https://api.nakala.fr/oai2?verb=GetRecord&metadataPrefix=oai_dc&identifier=oai:nakala.fr:doi_{identifia`

- en requetant l'API REST de Nakala, où les métadonnées sont exposées en JSON et en XML. :::info :::spoiler Exemples de requêtes API Pour les métadonnées d'une collection: `https://api.nakala.fr/collections/{identifiant de la collection}/metadatas` Pour les métadonnées d'une `https://api.nakala.fr/datas/{identifiant nakala de la donnée}/metadatas`

Une fois les métadonnées récupérées, elles peuvent être retravaillées, converties aux formats CSV ou TSV, et importées grâce modules *Bulk Import* (Import en lot) ou *CSV Import* pour enrichir l'item. Il est alors possible de lui associer des médias.

Appeler les fichiers Nakala par leurs manifestes Image IIIF

Les fichiers images peuvent être manipulés et appelés individuellement grâce à l'API Image du serveur Cantaloupe de Nakala, qui fera office de serveur IIIF externe à la bibliblioth\u00e8que numérique Omeka, comme expliqué précement (voir API Image IIIF). Ils peuvent donc être intégrés au niveau du média Omeka grâce au widget d'integration natif *Image IIIF* (voir Média Image IIIF) pour être associés à un item.

Créer l'URL d'un manifeste Image IIIF pour un fichier unique

L'URL Nakala d'accès au manifeste Image IIIF d'un fichier est constitué ainsi `https://api.nakala.fr/iiif/{identifiant de la donnée}/{identifiant du fichier}/info.json`. Le préfixe `/info.json` indique bien le fichier JSON du manifeste Image IIIF, sans lui, on peut accéder directement au fichier original.

> La documentation de l'API Nakala peut aider à construire l'URL du manifeste IIIF, et à transformer l'image le cas échant. - Requête GET `/iiif/{identifier}/{fileIdentifier}/info.json` - Requête GET `/iiif/{identifier}/{fileIdentifier}/{region}/{size}/{rotation}/{quality}.{format`

Ces deux identifiants, de la *donnée* d'une part, et du *fichier* d'autre part, sont accessibles directement depuis la page du document: - le DOI de la donnée se trouve sous le titre de la donnée (ou dans son URL); - l'identifiant du fichier, associé à celui de la donnée, est accessible sous la visionneuse.

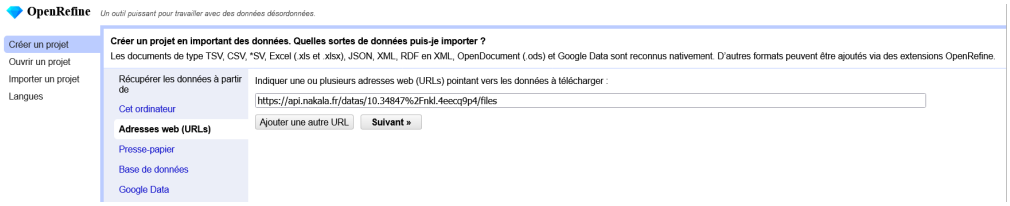
Une fois construite, cette URL peut être collée dans l'ingéreur *Image IIIF* pour ajouter un média à votre item.

> Attention, si vous souhaitez importer l'image modifiée grâce le protocole IIIF (sélection d'une région, d'une taille, d'une rotation, d'un contraste, etc.) il s'agira alors d'un fichier image, et il faudra donc passer par l'ingéreur *URL*.

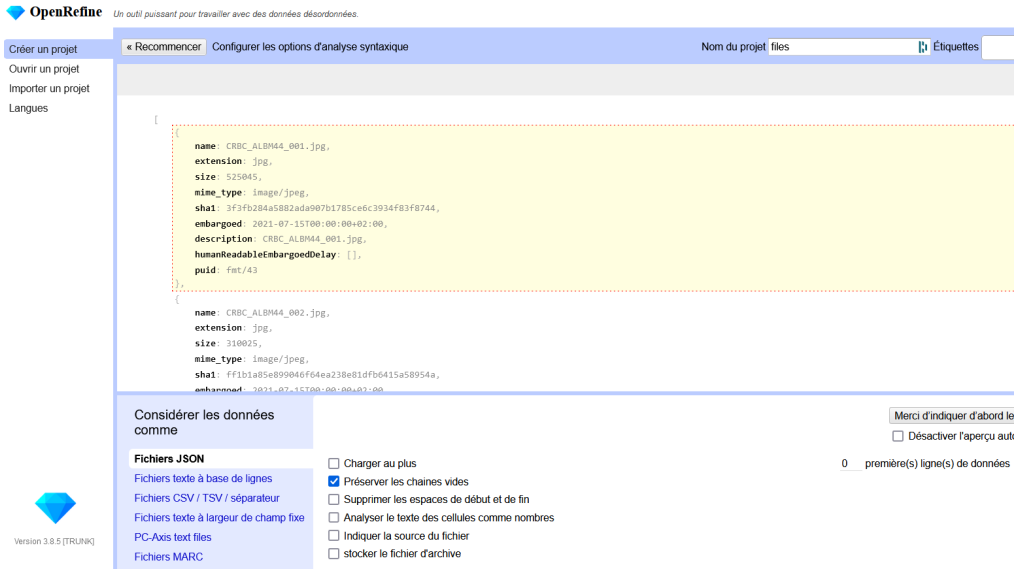
Créer les URL des manifestes Image IIIF pour l'ensemble des fichiers d'une donnée

Si vous souhaitez appeler plusieurs fichiers liés à une donnée, privilégiez le traitement en masse des urls grâce à OpenRefine.

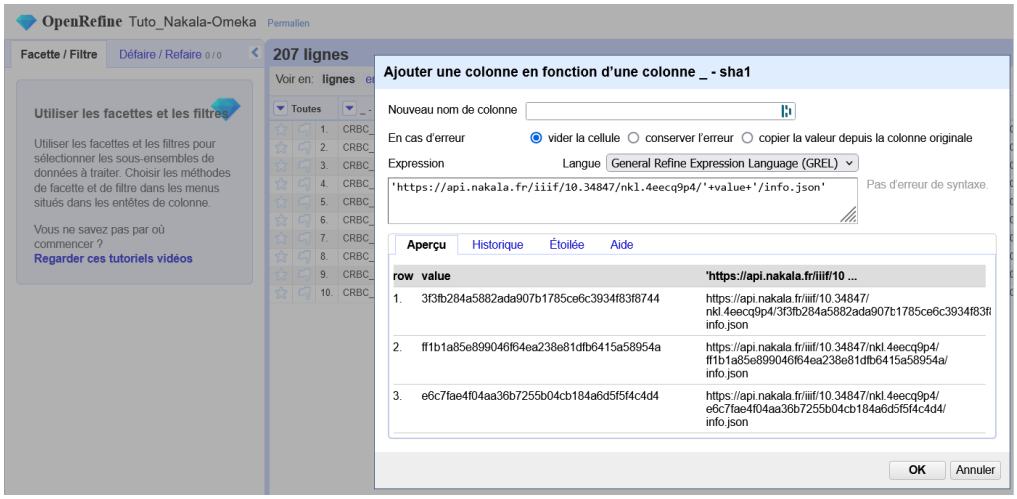
- Récupérez la liste des fichiers via l'API REST de Nakala grâce à la requête:
  `https://api.nakala.fr/datas/{identifiant de la donnée}/files`
- Entrez cette URL comme source de votre nouveau projet sur Openrefine *Adresse web (URLs)*:



- Sélectionnez le niveau de division des données du fichier JSON:



- La colonne `_ - sha1` contient l'identifiant du fichier: créez une nouvelle colonne à partir de celle-ci pour générer l'URL du manifeste Image IIIF avec l'expression suivante (n'oubliez pas les guillemets!):
  `'https://api.nakala.fr/iiif/{identifiant de la donnée}/'+value+'/info.json'`
- Ajoutez une colonne pour indiquer à chaque ligne



  l'identifiant Omeka `o:id` de l'item auquel vous souhaitez associer ces

  médias ;
  ajoutez également les métadonnées de chaque fichier si necessaire (générez un identifiant ainsi qu'un lien source vers Nakala, par

  exemple) ;
  supprimez les colonnes inutiles.

- Exportez le résultat au format CSV depuis OpenRefine, et importez-le dans votre bibliothèque numérique Omeka: précisez bien que la source de chaque média est l'ingéreur est *Image IIIF* pour la colonne concernée. Pour plus de détails, consultez intégration d'URL d'images IIIF au format JSON.

Cas d'usage n° 7 : Utiliser un serveur d'images IIIF autonome : Cantaloupe

Prérequis :

- installation Java + Cantaloupe
- installation standard d'Omeka S

Présentation de Cantaloupe

Cantaloupe est l'un des serveurs d'images les plus répandus. Il est conforme à toutes les versions de l'API Image IIIF, jusqu'à la version 3.0. Cantaloupe est un logiciel libre. Il fonctionne dans un environnement Java sous Linux ou Windows. Un conteneur Docker préconfiguré est également disponible.

Cantaloupe est fourni prêt à l'emploi et est réputé fonctionner dès son installation. Cependant un paramétrage avancé et une mise en production nécessitent des compétences en administration système.

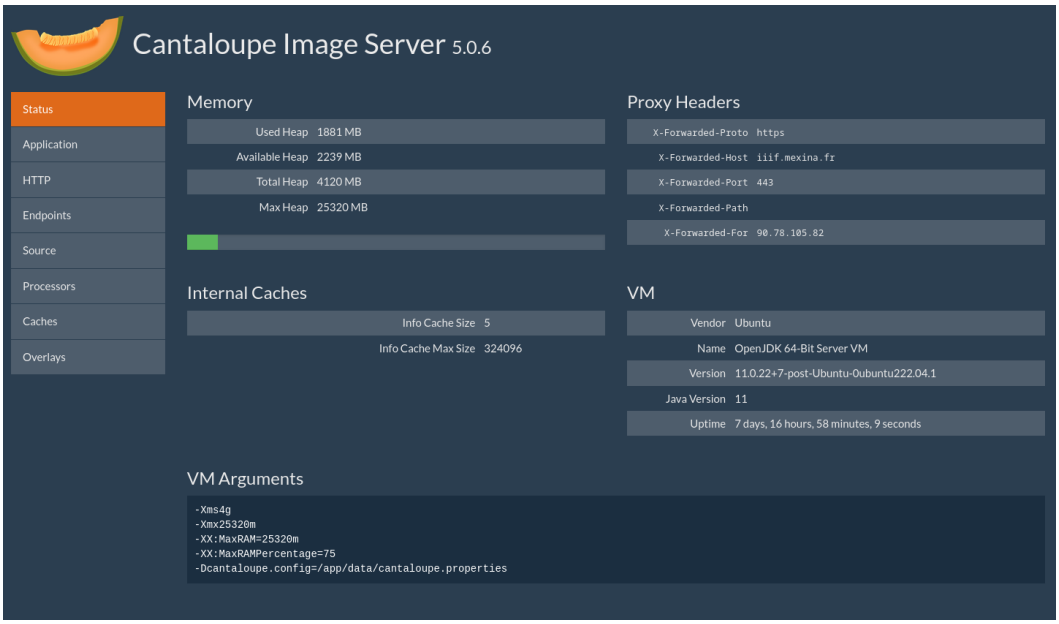
La documentation fournie est très complète et permet de faire face à un grand nombre de situations, notamment en ce qui concerne l'accès aux images, les processeurs adaptés aux différents formats et les stratégies de cache.

Liens :

- Site officiel,
- documentation (version 5.0 et précédentes).

Configuration

L'ensemble de la configuration d'une installation est rassemblée dans un seul fichier de configuration. Il est modifiable manuellement ou via une interface web conviviale, mais non traduite en français.



Interface d'administration de Cantaloupe

Procédure d'installation basique

- assurez-vous que Java (version 17 ou éventuellement supérieure) est installé sur votre machine (Linux ou Windows),
- téléchargez la dernière version de Cantaloupe (la 5.0.7. publiée en mars 2025) depuis le site officiel.
- décompressez l'archive sur votre machine,
- renommez ou copiez le fichier `cantaloupe.properties.sample` en `cantaloupe.properties`.
- ouvrez-le dans un éditeur de texte afin de
  - définir la modalité et le chemin d'accès aux images :
    - `source.static = FilesystemSource`
    - `FilesystemSource.BasicLookupStrategy.path_prefix =` chemin du dossier contenant des images,
  - activer l'interface web d'administration à l'adresse `/admin`, si vous le souhaitez :
    - `endpoint.admin.enabled = true` true/false pour activer ou désactiver,
    - `endpoint.admin.username = admin` identifiant pour accéder à l'interface,
    - `endpoint.admin.secret = sesame` mot de passe.
- lancez Cantaloupe
  - sous MacOS/Linux, avec la commande
    `java -Dcantaloupe.config=/path/to/cantaloupe.properties -Xmx2g -jar cantaloupe-5.0`
  - sous Windows, avec la commande
    `java -Dcantaloupe.config=C:\path\to\cantaloupe.properties -Xmx2g -jar cantaloupe-5`

Usage :
En supposant que vous ayez déposé une image nommée `image.tif` dans le répertoire indiqué et que vous souhaitiez utiliser la version 2 de l'API Image, accédez

- au fichier `info.json` via l'URL :
  `http://localhost:8182/iiif/2/image.tif/info.json`
- à l'image complète en résolution maximale via l'URL :
  `http://localhost:8182/iiif/2/image.tif/full/max/0/default.jpg`
- à un carré de 200 pixels de côté découpé dans l'angle supérieur gauche de l'image via l'URL : `http://localhost:8182/iiif/2/image.tif/0,0,200,200/max/0/default.jpg`

Formats d'image et processeurs d'images

Cantaloupe est principalement utilisé pour servir des images, il accepte également les documents PDF et les vidéos. Les principaux formats d'image sont pris en charge par au moins un processeur, ce qui permet de satisfaire, partiellement ou complètement, l'API image de façon plus ou moins performante. Dans l'administration de Cantaloupe, un tableau indique quels formats d'image sont pris en charge par quel processeur :

Formats d'images supportés

Les formats d'images TIFF tuilé et JPEG2000 sont considérés comme de bons choix pour Cantaloupe en termes de performance et de respect des spécifications de l'API Image. Pour le premier, la documentation conseille d'utiliser le processeur Java2d et pour le second l'un des processeurs suivants : Kakadu (licence propriétaire), Grok ou OpenJpeg.

Certains processeurs sont intégrés à l'installation standard, d'autres, comme Grok, OpenJpeg, TurboJpeg ou Kakadu, doivent être installés séparément.

Coupler un serveur d'images Cantaloupe IIIF avec Omeka

Les deux approches présentées ci-après sont mises en œuvre sans faire appel à des modules IIIF particuliers.

Les images sont stockées sur le serveur Cantaloupe

C'est la configuration par défaut de Cantaloupe (source statique `FilesystemSource`).

1. Récupérer les URL `info.json` des images sur ce modèle : https://SERVEUR/iiif/VERSION/NOM/info.json où

- SERVEUR est l'adresse du serveur,
- VERSION : 2 ou 3, version de l'API Image utilisée,
- NOM : identifiant de l'image démandée, généralement le nom du fichier.

2. Les ajouter à l'installation Omeka S :

- manuellement, via l'onglet d'ajout de medias, choix Image IIIF,
- en nombre : utilisation du module Import CSV (cf.Cas d'usage n°2 : Intégration et récupération de données).

Les images sont ailleurs

En paramétrant Cantaloupe pour récupérer les images via une URL, il est possible d'assurer une liaison transparente pour récupérer les urls `info.json`.

1. Paramétrer Cantaloupe pour utiliser des ressources en ligne, partie `Source` :

- `Strategy` : `Static`, il est aussi possible de faire appel à un script délégué afin de prendre en charge des situations plus complexes,
- `Static Sources` : `HttpSource`, et dans l'onglet du même nom :
- `Lookup Strategy` : `Basic`,
- `URL Prefix` : première partie de l'URL, avant l'identifiant de l'image, pour accéder à l'image, par exemple : `https://depot.mexina.fr/`,
- `URL Suffix` : partie de l'URL à la suite de l'identifiant, par exemple `` `/download ``,
- à titre d'exemple, la requête `https://iiif.mexina.fr/iiif/3/KISE0%2FcalAqiWA38.jpg/info.json` traite l'image téléchargeable à l'adresse `https://depot.mexina.fr/KISE0/calAqiWA38.jpg/download`

2. Pour chaque image, construire l'URL `info.json` à partir de l'identifiant du média (si l'identifiant comporte des caractères réservés : `/ ? # [ ] @ ! $ & ' ( ) * + , ; =` il est nécessaire de les encoder (URL encoding, ou encodage pourcent, par exemple / par `%2F`, comme dans l'exemple ci-dessus)
3. Les intégrer à l'installation Omeka S :

- manuellement, via l'onglet d'ajout de medias, choix Image IIIF,
- en nombre : utilisation du module Import CSV (cf.Cas d'usage n°2 : Intégration et récupération de données).

GLOSSAIRE

API

Application Programming Interface ou interface de programmation applicative : Interface logicielle permettant à des logiciels distincts de communiquer entre eux, par exemple pour échanger des données. Six APIs ont été développées par la communauté IIIF dont les principales sont l'API Image et l'API Presentation.

Canevas

(ou *Canvas*) : Vue unique au sein d'une séquence d'images. Si ce canevas correspond généralement à une image ou à une page d'un document, ce n'est pas toujours le cas, puisqu'il est par exemple possible de combiner plusieurs images au sein d'une même vue.

JSON-LD

(*JavaScript Object Notation for Linked Data*) : Format utilisé pour encoder les données structurées des Manifestes et des Collections IIIF, suivant une syntaxe simple clé/valeur. Il s'agit d'un format standardisé au W3C, organisme de normalisation du Web.

Manifeste

(*Manifest*) : Fichier unique, au format JSON-LD, récapitulant les informations relatives à la structure d'un objet numérique (séquence des images, URL des images, métadonnées…). Disposer de l'URL d'un Manifeste permet de réutiliser les images qui y sont liées, par exemple en les affichant dans n'importe quelle visionneuse compatible.

Serveur d'images

Système permettant de délivrer les images conformément à l'API Image. Cantaloupe est l'un des serveurs d'images les plus utilisés mais il en existe d'autres sur cette liste.

Tuile

(*Tile*) : Une tuile est une unité de découpage d'une image originale qui permet à celle-ci d'être servie par partie plutôt que dans son intégralité. Certains formats d'images intègrent directement toutes les tuiles possibles d'une image (Jpeg2000, Tiff pyramidal). Un serveur d'image spécialisé comme Cantaloupe génère dynamiquement des tuiles d'images pour tous les types de formats qu'il accepte. (voir doc ci-après)

Visionneuse

Outil logiciel permettant de visualiser un média ou un document.

OUTILS

Pour une liste quasi-exhaustive, se référer à la page awesome-IIIF https://github.com/IIIF/awesome-iiif.

Cantaloupe

Serveur IIIF

https://cantaloupe-project.github.io

IIPImage

serveur IIIF

https://iipimage.sourceforge.io/

Niiif-niiif

Outil de génération de manifeste IIIF à partir de données stockées dans l'entrepôt Nakala

https://gitlab.huma-num.fr/biblissima/niiif-niiif

Mirador

visualiseur

https://projectmirador.org/index.html

UniversalViewer

visualiseur

https://universalviewer.io/

Références & ressources IIIF

Cf. groupe publique Zotero IIIF francophone.

- Site officiel du consortium IIIF. (en anglais)
- Awesome IIIF, ressources de la communauté IIIF. (en anglais)
- Documentation IIIF Biblissima+
- Pasquier T., Robineau R. IIIF - La solution ouverte, pérenne et sobre pour diffuser et valoriser les images numériques, *La Lettre de l'OCIM*, n° 210, décembre 2024.
- Denoyelle M., Petermann D. Recueil de bonnes pratiques pour la diffusion en ligne des images patrimoniales, INHA, octobre 2024.
- IIIF, un outil pour visualiser les archives numérisées sur FranceArchives, France Archives, juin 2024.
- Interopérabilité des images : IIIF, France Archives, juin 2024.
- Limonade & Co, Omeka S et le protocole IIIF - Importer des métadonnées et des médias depuis une API IIIF
- Pasquier T. Créatures ou IA : consultez, manipulez & annotez les images des bibliothèques, musées… grâce à IIIF, LinuxFr, mai 2024.
- Perret, Florence. *Les manifestes IIIF et l'interopérabilité des standards*. Carnet hypothèses du consortium Huma-Num DISTAM (DIgital STudies Africa, Asia, Middle East). 2023. Consulté le 22 janvier 2025 à l'adresse https://doi.org/10.58079/npbx
- Nakala et IIIF, Biblissima.
- IIIF pour les musées de France, Ministère de la Culture, juillet 2023.
- Prunet C., Bertrand S., Chenard G., Pillorget S., Robineau R. IIIF : découverte et interopérabilité sans frontières des images patrimoniales *Culture et recherche* n°143, 2022.
- Snydman S., Sanderson R., Cramer T. The International Image Interoperability Framework (IIIF): A community & technology approach for web-based images, Archiving 2015, 2015. (en anglais)

ANNEXE 1 – OMEKA S IMAGESERVER ET LES TUILES

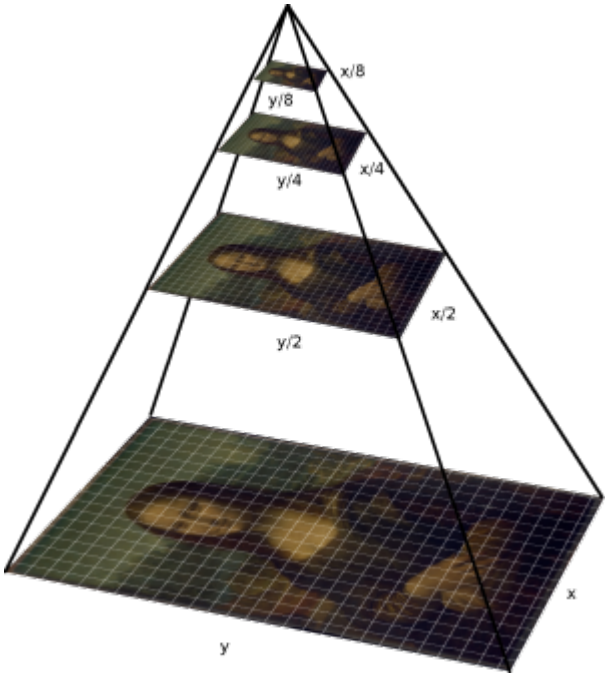
La présente annexe mélange des données théoriques ainsi que des résultats de tests pour affiner la compréhension du module *Image Server*.

Que fait le module Image Server ?

Le module Image server effectue deux opérations liées mais distinctes: 1) Il implémente l'API Image IIIF, au sens où il sait répondre aux requêtes du type : * modèle d'URL: `{scheme}://{server{/prefix}/{identifier}/{region}/{size}/{rotation}/{quality}.{format}` * exemple : https://api.nakala.fr/iiif/10.34847/nkl.8e31p9d2/014f975b5550100f7a5b977ae409d4c51f3ae263/full/full/0/default.jpg

2. Il effectue des traitements sur les images pour optimiser ou transformer les images qu'il envoie lorsqu'il répond à une requête API Image IIIF. Parmi ces traitements, un des plus importants est le tuilage.

Qu'est ce que l'opération de tuilage ?



Source : IIPImage

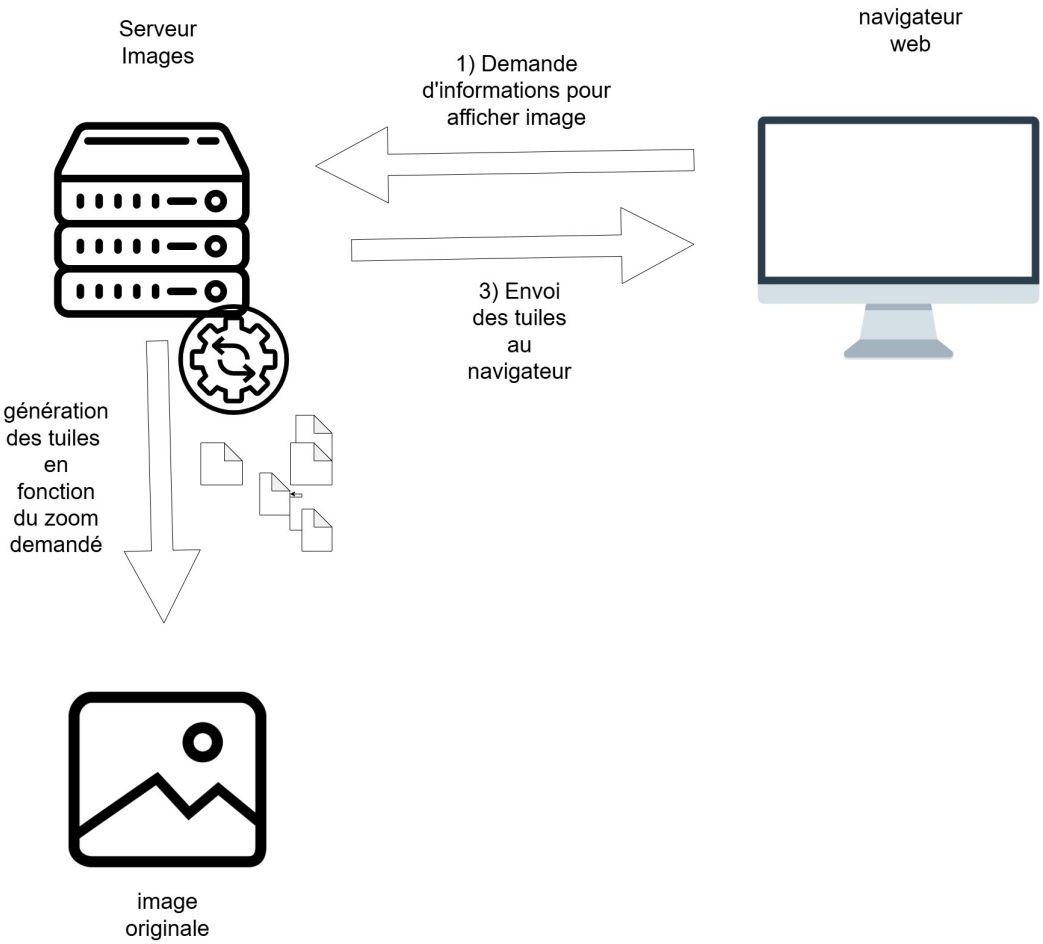
Le tuilage consiste à découper une image de grande taille en de plus petites images (appelées tuiles) qui seront affichées uniquement lorsque le niveau de zoom requis le nécessitera. Cela permet d'optimiser le temps de chargement des images ainsi que de minimiser les données transférées entre un serveur web hébergeant les images et le navigateur web d'un utilisateur.

Tuilage dynamique *vs* tuilage statique

Cette opération peut s'effectuer selon deux modes:

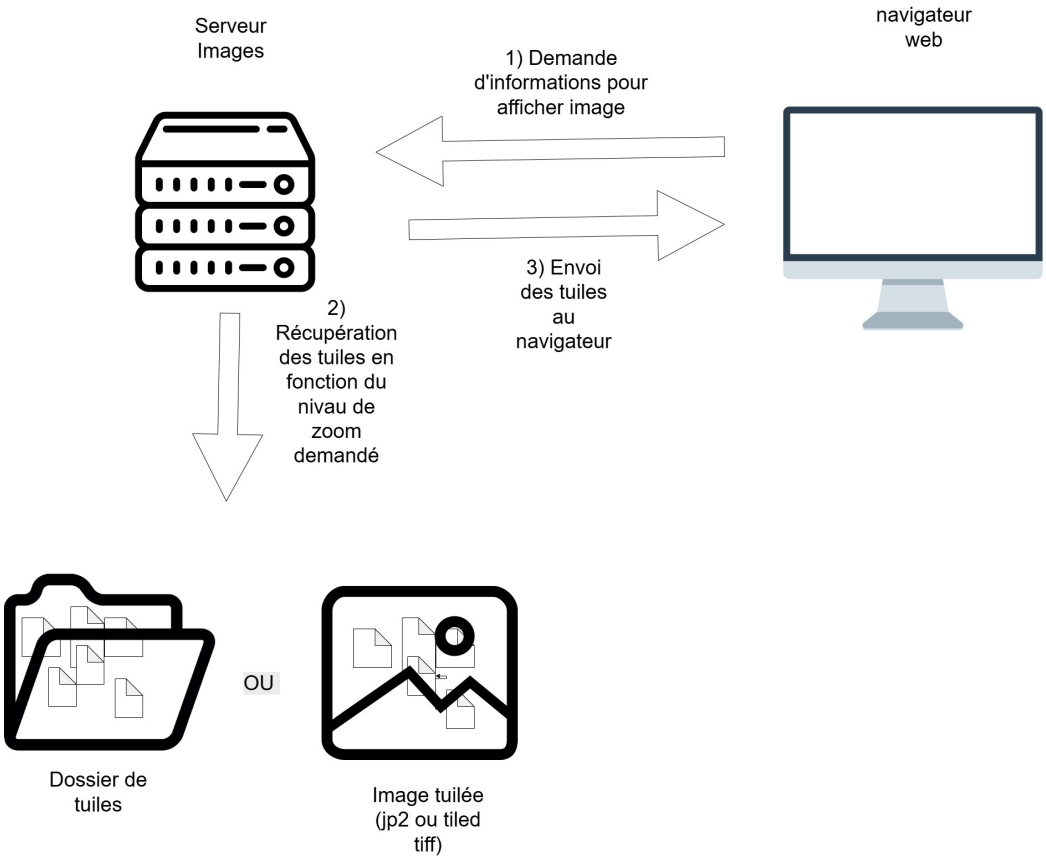
- **dynamique** : les tuiles sont générées au moment où la requête de zoom est demandée par le navigateur. Cela évite de stocker un nombre de tuiles important sur le disque d'un serveur via un travail de prégénération de tuiles.

Tuilage dynamique



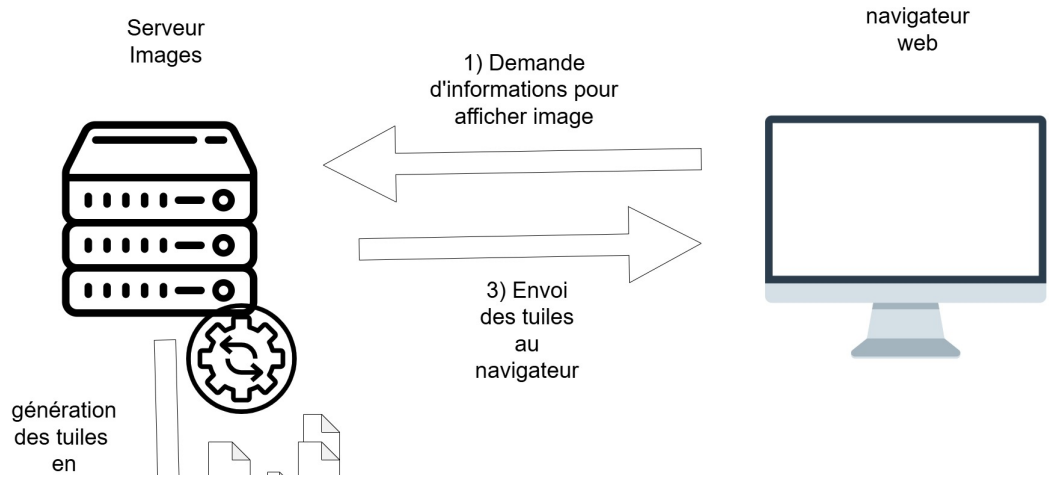
- **statique** : les tuiles sont générées en amont de l'affichage et stockées directement sur le serveur. Plusieurs formats de tuiles existent parmi lesquels : DZI (DeepZoom), Zoomify, Tiled Tiff, JPEG 2000. Les deux premiers formats (DZI et Zoomify) créent et organisent les tuiles à l'intérieur de répertoires quand les deux autres les encapsulent dans un seul fichier (JPEG 2000 et Tiled Tiff).

Tuilage statique



Pour ces deux derniers formats, l'embarquement des tuiles dans un seul fichier devrait améliorer grandement les temps d'affichage des images de grande taille, mais cela demande d'utiliser des logiciels qui soient capables de lire les tuiles à l'intérieur de ces fichiers au gré des demandes effectuées via le visualiseur IIIF (Openseadragon, Mirador, etc.). Cela ne semble pas être le cas avec le module **Image Server** qui, d'après les tests que nous avons effectués, génère dynamiquement les tuiles à partir de ces deux formats tuilés **Tiled Tiff** et **JPEG 2000**. Pour de grandes images et des ressources serveur limitées, le résultat est bien plus lent qu'avec des tuiles statiques en format **DeepZoom**.

Image server - Scenario jp2 et tiff pyramidal

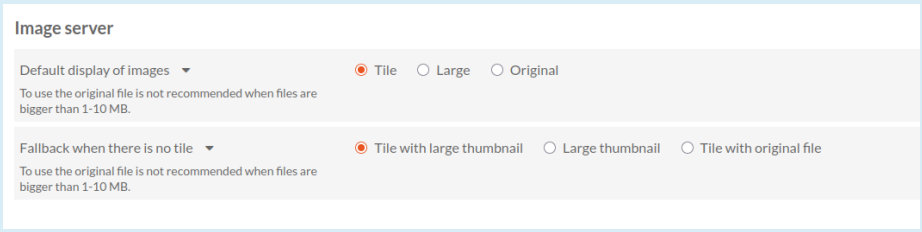


A notre connaissance, le module **Image Server** génère des tuiles statiques uniquement pour les formats **DeepZoom** et **Zoomify** et uniquement pour les besoins standards de zoom sur une image. En revanche, il utilise le tuilage dynamique pour des requêtes IIIF de modifications d'images (rotation, transformation N&B, sélection de régions spécifiques) qui ne peuvent être anticipées en amont. Cela permet d'équilibrer vitesse d'affichage (avec tuiles prégénérées) et souplesse fonctionelle (génération de tuiles dynamiques uniquement quand c'est néecssaire). Comme expliqué au dessus, le module semble aussi générer dynamiquement les tuiles lors de l'utilisation de fichiers prétuilés **JPEG 2000** et **Tiled Tiff**, même pour un zoom "normal", ce qui peut avoir un impact significatif sur les performances.

Tests de différentes configurations du module Image Server

> Première observation en effectuant les tests: avec PHP7.4, le tuilage ne s'effectuait pas sur notre serveur mais il n'y avait aucun message d'erreur relatif à la version de PHP qui permettait de le comprendre. En basculant à PHP8.1 le tuilage fonctionne.

Bonne nouvelle, même dans un cas d'usage où le tuilage ne s'effectue pas, les images peuvent tout de même s'afficher en IIIF via le module **Image Server** grâce à des mesures de contournement prévues par le module en cas d'absence de tuiles. On peut voir et configurer ces mesures dans les paramètres généraux d'Omeka-S rubrique **Image Server**:



Processeurs d'image

Les tests ont été effectués sur Ubuntu 24.04.

Le processeur d'image **Vips** a été installé préalablement aux tests qui suivent.

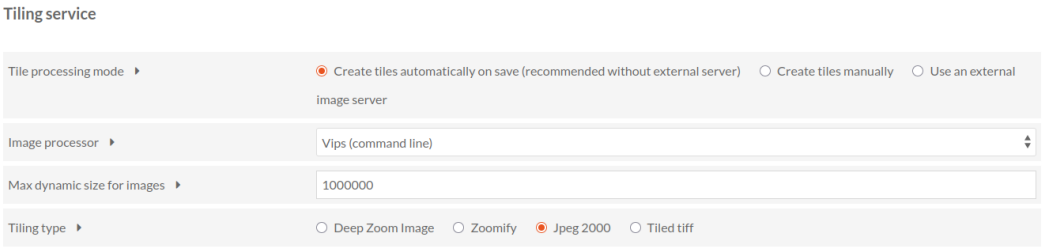
Sur un système comme Ubuntu son installation est très simple:

```
sudo apt install libvips
sudo apt install libvips-tools
```

Idem pour le processeur **Image Magick** :

```
sudo apt install imagemagick
```

Configuration avec un format de tuilage JPEG 2000



Le format de tuilage sélectionné, ici JPEG 2000, est utilisé quel que soit le format initial du fichier image. Dans les exemples ci-dessous les format de fichiers images associés à mon item de test dans Omeka S sont JPG, PNG, JP2. Le fait d'associer des images JPEG 2000 ou Tiled Tiff à un item Omeka S n'empêchera donc pas le module de générer ses propres fichiers de tuiles embarquées JPEG 2000 et Tiled Tiff.

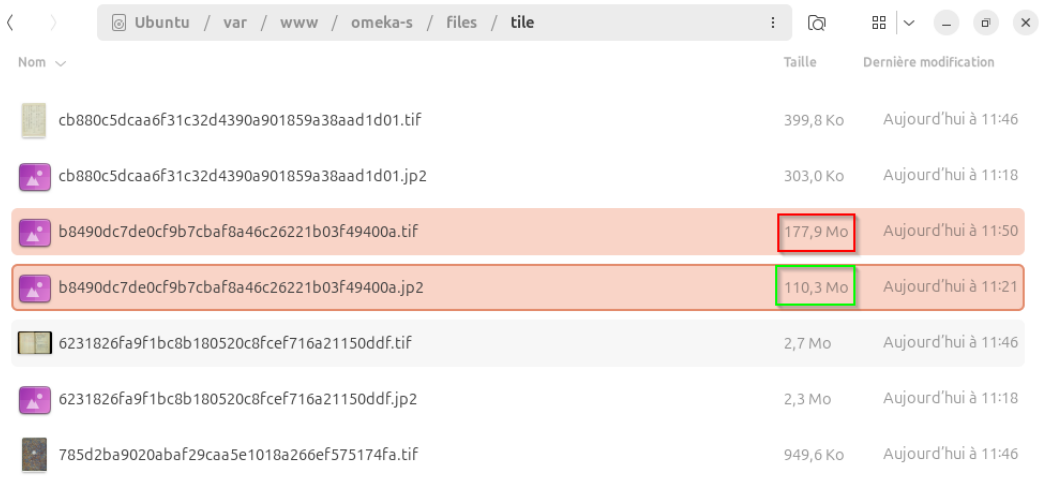
> Si le paramètre **create tiles automatically** est coché et si vous modifiez le format de tuilage dans la configuration du module, l'opération de génération du nouveau format de tuilage sera déclenchée à la prochaine modification d'un item, qu'il s'agisse de charger un nouveau média ou bien seulement de modifier une métadonnée. Cela peut donc avoir un effet non négligeable sur les ressources serveurs lors de modifications d'items en lots.

Une fois les tuiles générées, on observe bien dans le répertoire `/files/tile` les fichiers JPEG 2000 générés:

```
htl@Ubuntu24:/var/www/omeka-s/files$ ls tile/
273ff613b293ef46bcde1985c23833c72b58c949.jp2  785d2ba9020abaf29caa5e1018a266ef575174fa.jp2
2e59f572493398fbe018e86597f0464a1e110efd.jp2  b8490dc7de0cf9b7cbaf8a46c26221b03f49400a.jp2
417c2be0ab75fc9b4b5f5d46bbb5c354c166db7c.jp2  cb880c5dcaa6f31c32d4390a901859a38aad1d01.jp2
5cb923bdafa90bb5edd4f2a4bb3229abe9c3dfc0.jp2  index.html
6231826fa9f1bc8b180520c8fcef716a21150ddf.jp2
```

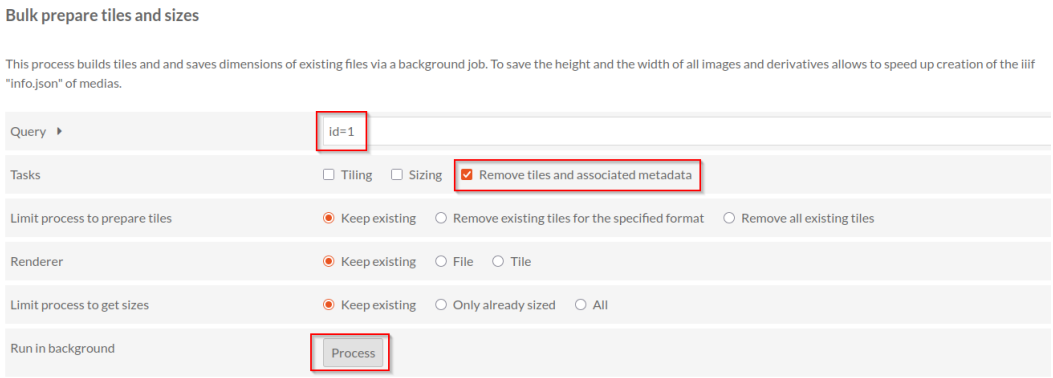
Avec un format de tuilage Tiled tiff

Comme indiqué précédemment, le format configuré ayant été modifié, lors de la modification d'un item, les medias associés sont tuilés sous forme de Tiled Tiff.



> La génération automatique des tuiles ne permet pas de préciser si on souhaite effacer ou non les tuiles d'un autre format déjà présentes. On peut donc observer un doublon dans les fichiers de tuiles présents dans le répertoire `tile`. Nous avons en effet un fichier `.jp2` et un fichier `.tif` pour une même image originale. Conserver plusieurs formats prend de la place. Il peut donc être utile de nettoyer le répertoire `/files/tile` lorsque vous observez ce type de "doublons" lorque vous passez en production.

Fonctionnalité à manier avec précautions!! Il est possible de supprimer les tuiles déjà générées avant d'en créer de nouvelles, via l'onglet de configuration `Bulk prepare tiles and sizes` du module **Image Server**. Pour ce faire, il suffit de cocher `Remove tiles and associated metadata` puis de cliquer sur `Process`. Si vous ne remplissez pas le champ `query` toutes les tuiles associées à l'ensemble de vos images seront supprimées. Il est possible de réduire le *scope* de suppression par exemple en sélectionnant uniquement les tuiles liées à l'item d'id 1 via la requête `id=1`.



> De même, les fichiers **Tiled Tiff** générées sont plus volumineux que les fichiers **JPEG 2000**. Il peut donc être intéressant de préférer ce dernier format au détriment du **Tiled Tiff** qui est plus lourd.

> Il est important de déterminer le contexte d'implémentation afin de choisir le format de tuilage le plus judicieux pour votre usage. Par exemple, comme expliqué plus haut, les tests effectués sur notre serveur ont révélé de mauvaises performances avec les formats de tuile **JPEG 2000** et **Tiled Tiff** pour afficher des images volumineuses (> 100Mo). Dans notre cas de test, le format le plus rapide est sans conteste **DeepZoom**.

Avec un format de tuilage DeepZoom

Comme indiqué plus haut, **DeepZoom** a montré de meilleures performances dans les conditions de ressources machines que nous avons à disposition. Nous ne testerons pas l'autre format de tuiles statiques **Zoomify** car il est déconseillé par le développeur du module et semble avoir été conservé pour des raisons historiques liées au développement du module.

```
htl@Ubuntu24:/var/www/omeka-s/files$ ls tile
273ff613b293ef46bcde1985c23833c72b58c949.dzi    6231826fa9f1bc8b180520c8fcef716a21150ddf_files
273ff613b293ef46bcde1985c23833c72b58c949_files  785d2ba9020abaf29caa5e1018a266ef575174fa.dzi
2794a52f6c43c94ebf84358bb7c7b5a6549ccfb7.dzi    785d2ba9020abaf29caa5e1018a266ef575174fa_files
2794a52f6c43c94ebf84358bb7c7b5a6549ccfb7_files  b8490dc7de0cf9b7cbaf8a46c26221b03f49400a_files
2e59f572493398fbe018e86597f0464a1e110efd.dzi    c51843cbe76dc5c413b8399786bbde05224bb2ab.dzi
2e59f572493398fbe018e86597f0464a1e110efd_files  c51843cbe76dc5c413b8399786bbde05224bb2ab_files
5cb923bdafa90bb5edd4f2a4bb3229abe9c3dfc0.dzi    cb880c5dcaa6f31c32d4390a901859a38aad1d01.dzi
5cb923bdafa90bb5edd4f2a4bb3229abe9c3dfc0_files  cb880c5dcaa6f31c32d4390a901859a38aad1d01_files
6231826fa9f1bc8b180520c8fcef716a21150ddf.dzi    index.html
htl@Ubuntu24:/var/www/omeka-s/files$
```

Un format DeepZoom est constitué de 2 parties: - Un répertoire qui accueille toutes les tuiles liées au même média - Un fichier en `.dzi` qui décrit la taille de chaque tuile et le format de fichier image des tuiles tel que décrit par la documentation de Deepzoom. Par exemple:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Image xmlns="http://schemas.microsoft.com/deepzoom/2008"
  Format="jpg"
  Overlap="0"
  TileSize="256"
  >
  <Size
    Height="860"
    Width="1605"
  />
</Image>
```

Dans le répertoire, les tuiles sont organisées par niveau de zoom (de 0 à 13 ci-dessous):

```
htl@Ubuntu24:/var/www/omeka-s/files$ ls tile/c51843cbe76dc5c413b8399786bbde05224bb2ab_files
0  1  10  11  12  13  2  3  4  5  6  7  8  9  vips-properties.xml
htl@Ubuntu24:/var/www/omeka-s/files$ cat tile/c51843cbe76dc5c413b8399786bbde05224bb2ab_files/vips-properties.xml
<?xml version="1.0"?>
<image xmlns="http://www.vips.ecs.soton.ac.uk//dzsave" date="2025-05-16T16:39:46.464753+02" version="8.15.1">
  <properties>
    <property>
      <name>width</name>
      <value type="gint">4386</value>
    </property>
    <property>
      <name>height</name>
      <value type="gint">5768</value>
    </property>
    <property>
      <name>bands</name>
      <value type="gint">3</value>
    </property>
    <property>
```

> le fichier `vips-properties.xml`, qui est situé dans le répertoire des tuiles, contient les métadonnées techniques de l'image originale. Il est généré automatiquement par **Vips**.

Annexe 2 – Cas d'usage Omeka Classic

Si vous souhaitez utiliser IIIF sous Omeka Classic, un certain nombre de plugins et de ressources existent :

- IIIF Toolkit développé par la bibliothèque de l'université de Toronto,
- IIIF Toolkit Embed
- Outils IIIF pour Omeka Classic
    - iiif-MaGe-for-nakala pour générer un manifest IIIF à partir d'une donnée Nakala.
    - iiif2tei Divers scripts pour aller du IIIF vers la TEI.
    - tei2iiif Divers scripts pour aller de la TEI vers du IIIF.

Ils correspondent plutôt à des cas d'usage avancés. Pour simplement afficher des ressources IIIF dans un Omeka Classic, vous pouvez suivra la procédure suivante (merci à la plateforme EMAN pour ce retour d'expérience).

1. Récupérer le lien du manifeste IIIF de la ressource extérieure que l'on souhaite afficher, le copier et le coller dans un champ de métadonnées choisi au préalable (par exemple dublin core "source").
2. Installer le plugin permettant d'utiliser la visionneuse Universal Viewer dans Omeka Classic.
3. Paramétrer Universal Viewer pour qu'il visionne du IIIF.

Pour cela il faut indiquer la métadonnée utilisée pour le manifeste IIIF dans les paramétres du plugin Universal Viewer sous l'option « Alternative Manifest Source ».

À noter qu'il est possible d'utiliser la propriété sélectionnée pour stocker le manifeste IIIF pour afficher également d'autres informations. Il est alors par contre nécessaire que le manifeste soit la valeur en première position dans le champ pour que la ressource IIIF s'affiche bien dans la visionneuse.

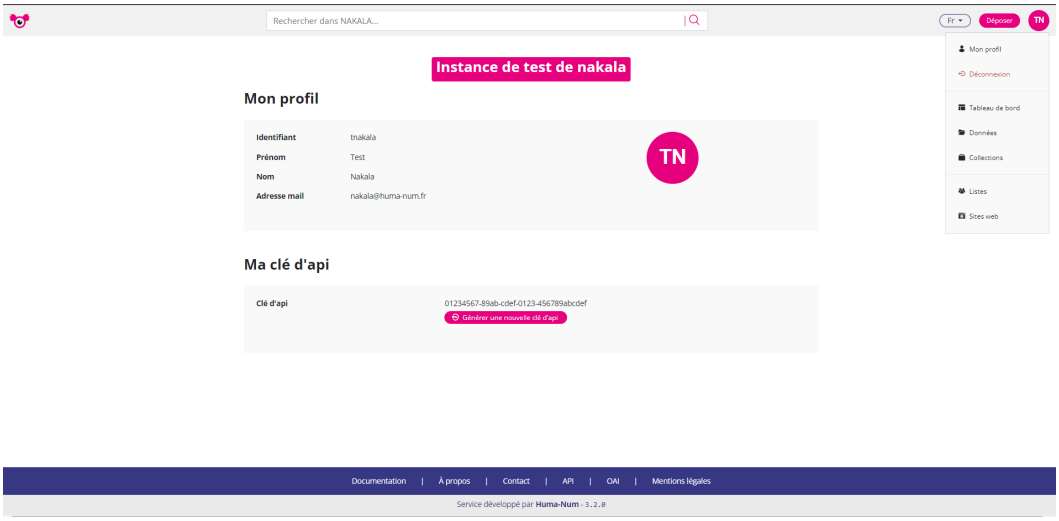
Il peut être nécessaire de modifier le thème utilisé (fichier `items/show.php`) pour forcer l'affichage de la visionneuse Universal Viewer même en absence de fichier image sur le serveur (les ressources étant disponibles à distance via IIIF)

Annexe 3 - Générer un manifeste pour une donnée Nakala

Comme nous l'avons dit dans le cas d'usage n°6, Nakala ne génère pas de manifestes IIIF. Mais il peut tout à fait en accueillir ! En effet, le json peut être un fichier lié à la donnée. C'est le choix qu'ont fait plusieurs projets et bibliothèques numériques comme Fontegaia ou Banyan.

Des scripts ont été conçus pour générer des manifestes à partir de fichiers déposés sur Nakala comme niiif-niiif. La version d'origine de Jean-Baptiste Pressac sauvegarde automatiquement le manifeste sur Nakala, contrairement à la version plus développée de Biblissima (qui permet une création en masse sur toute une collection). Un autre outil, développé par Elan à Grenoble est iiif-MaGe-for-nakala ou encore la génération en lots de manifestes avec annotations IIIF développé dans le cadre du projet Collex IndexAngkor: https://archive.softwareheritage.org/browse/origin/directory/?origin_url=https://github.com/vpaillusson/niiif-niiif-annotations

Attention : pour déposer des contenus sur Nakala, il faut demander un accès à la plateforme en justifiant de votre mission de recherche. De même, pour utiliser ces outils semi-automatisés, il vous faut une clé d'API que vous pouvez générer une fois cet accès octroyé.



Source : Huma-Num

Ces scripts nécessitent une certaine aisance avec Python, soit à partir de lignes de commande soit d'un gestionnaire d'environnement. Suivez les instructions développées sur les gitlab pour arriver à générer vos manifestes.

Une fois le manifeste généré et déposé sur Nakala ou conservé sur un autre entrepôt accessible, vous pouvez l'appeler soit en important un média de type `IIIF Présentation`, soit comme métadonnée de l'item dans la "propriété fournissant un manifeste tiers" (par défaut le champs dcterms:hasFormat) configurée dans IIIF Server.

Liste de sites Omeka avec IIIF

Liste non exhaustive de sites Omeka S qui utilisent le format IIIF.

Si vous souhaitez ajouter votre site à la liste, merci de compléter ce formulaire ou envoyez nous un message.

- 1886 - collections patrimoniales numérisées de l'université Bordeaux Montaigne
- Banyan, la bibliothèque numérique de l'École française d'Extrême-Oriente
- Bibliothèque numérique de la Sorbonne Nouvelle
- Bibliothèque numérique de l'Observatoire de Paris - PSL)
- Bina - collections patrimoniales numérisées de la Bibliothèque universitaire des langues et civilisations (BULAC)
- Corpus - Musée du Louvre
- Digital Muret - Institut national d'histoire de l'art (INHA)
- Genovefa, la bibliothèque numérique de la bibliothèque Sainte-Geneviève
- Horae Pictavenses : origines et provenances des manuscrits poitevins étudiés dans le texte et l'image
- Humathèque Condorcet - Bibliothèque Numérique du Campus Condorcet
- Lillonum - bibliothèque numérique patrimoniale de l'Université de Lille
- NuBIS, la bibliothèque numérique de la Bibliothèque interuniversitaire de la Sorbonne
- PSL explore
- Transcrire
- UCL Archives - patrimoine de l'université catholique de Louvain)

LISTE DES CONTRIBUTRICES ET CONTRIBUTEURS

- Anne Bugner (BULAC)
- Olivier Ghuzel (Université Paris Cité)
- Laurent Nabias (Bibliothèque de l'Université Sorbonne Nouvelle)
- Vincent Paillusson (HTL)
- Thierry Pasquier (Espace Mendès France)
- Pauline Rivière (Bibliothèque Sainte-Geneviève)
- Régis Robineau (Biblissima)
- Alyx Taounza-Jéminet (Humathèque Condorcet)
- Elisa Thomas (Université PSL)
- Pierre Willaime (Archives Henri-Poincaré)

L'ASSOCIATION DES USAGERS FRANCOPHONES DE OMEKA

L'Association des usagers francophones de Omeka (AUFO) a été créée le 21 septembre 2021.

Ses objectifs sont les suivants :

- encourager l'entraide entre les utilisateurs francophones d'Omeka grâce au partage de l'information, de la documentation et des ressources ;
- entreprendre tout type d'action de valorisation et de formation pour promouvoir l'usage d'Omeka dans les communautés académiques, associatives et culturelles francophones ;
- se positionner comme un interlocuteur reconnu du Digital Scholar et d'autres acteurs publics ou privés intéressés par Omeka ;
- appuyer les demandes de développement du logiciel Omeka répondant en particulier aux besoins des utilisateurs francophones ;
- fédérer toute initiative pour améliorer l'environnement francophone d'Omeka et ses différents outils ;
- et plus largement promouvoir l'utilisation des logiciels libres et l'ouverture des données culturelles et scientifiques.

Son forum de discussion est ouvert à toutes et tous, adhérent·e·s ou pas.

Site de l'AUFO : https://omeka.fr

CHANGEMENTS DANS LE VADE-MECUM-IIIF

Pour le détail des changements, voir :

- Changelog, changements passés et de la version en cours (non encore publiée).
- Notes de versions

8 novembre 2025

- Amélioration de l'ergonomie du site
- Ajout des versions (expérimentales) PDF et EPUB
- Ajout d'un annuaire de sites Omeka utilisant IIIF

10 juillet 2025

- Annonce publique du Vadémécum Omeka S & IIIF

Table des matières