3D Juump Infinite Administration Manual

[AM_EN] version 4.1

Introduction

This documentation is aimed at **system administrators**. For *CAD database specialists*, refer to the *Integration manual*.

It describes the concepts and the installation and administration procedures of 3D Juump Infinite back-end servers and their underlying third-party components.

This documentation covers the introduction installation of the 3D Juump Infinite server components:

- ∞Directory, connecting all the components of the 3D Juump Infinite - with the Web *Administration portal*
- ∞Proxy, a proximity relay for data set broadcasting
- ∞CLI, a command line tool used to process data and manage 3D Juump Infinite clusters

A Cluster administration chapter explains the security policies, presents the *Administration portal* and describes some maintenance points and procedures.

The last part is dedicated to 3D Juump client applications:

- The development and deployment of Web applications powered by 3D Juump Infinite
- The installation of the 3D Juump Infinite native client application
- The protection of third-party applications through 3D Juump Infinite sessions

For a primer installation, we suggest to start with the setup procedures of an ∞Directory and an ∞Proxy, and validate the status the cluster with the Web *Administration portal*.

Then, to construct your own data sets, please read the chapter dedicated to the data set generation procedure.

Eventually, read the last chapters to develop and publish Web applications based on 3D Juump Infinite and/or deploy the native client application on your end-user host machines.

The authentication of the users on the 3D Juump Infinite system is forwarded to an OpenID Connect authentication server (Authenticator). The ∞Directory does not hold any authentication mechanism, and only defers such authentication. There are public servers that can provide such an authentication mechanism, for example Auth0 (https://auth0.com/), Google Workspace or Microsoft Azure AD. The system administrators may choose an open-source solution hosted on their own server infrastructure such as Keycloak (https://www.keycloak.org/).

Disclaimer

1 - Documentation coverage

The information outlined in this documentation is intended to be used for the following purposes:

- installation of 3D Juump Infinite back end and front end;
- creation of specialized 3D Juump Infinite datasets specific to customer needs;
- diffusion of 3D Juump Infinite datasets by the 3D Juump Infinite back end;
- administration of 3D Juump Infinite users, groups and components.

The Licensee is liable for any damage or injury to any person or entity arising out by or in connection with its developed works (pieces of software developed for/with 3D Juump Infinite) and its data sets (generated for/by 3D Juump Infinite).

The developed works and datasets shall be conformed to the design and implementation guidelines and restrictions described in the Documentation.

The software functions available for development are documented. The Licensee shall not utilize the undocumented functionalities.

The Licensee shall be compliant with the 3D Juump Infinite administration, integration and operation recommendations stipulated in the Documentation. AKKODIS INGENIERIE PRODUIT

SAS shall not be held responsible in case of any damage caused by the non-compliance to these recommendations.

2 - System security is under the Licensee's responsibility

The Licensee's system has to be compliant with the state of the art and the Documentation's requirements, especially in terms of security. It is reminded to the Licensee that it has to anticipate safety plans and measures to minimize the consequences notably linked to a possible temporary disruption or a data loss generated by the Software or by any security breach in the Licensee infrastructure which hosts 3D Juump Infinite.

The Licensee shall be held responsible in case the developed works and datasets compromise the integrity, performance and security of 3D Juump Infinite sofwares or products.

3 - Documentation liability

The source code and template examples provided through this Documentation are only intended to illustrate. They shall not be used by the Licensee, as developed works or datasets. If they are, AKKODIS INGENIERIE PRODUIT SAS declines any responsibility and any warranty is excluded.

Although reasonable effort is made to ensure that the information in the Documentation is complete and accurate at the time of release, AKKODIS INGENIERIE PRODUIT SAS cannot assume responsibility for any existing errors. Changes and/or corrections may be incorporated in future versions.

4 - Third-party software components

The following software components are provided along with 3D Juump Infinite installers and scripts:

- Executables and plugins located in the `third-party` subfolder,
- Libraries and their dependencies (enumerated in the licensing document).

The exhaustive list of third-party components and the author's identity is made available by AKKODIS INGENIERIE PRODUIT SAS in the menu of the Software, under the link "About". The Licensee is responsible for their proper installation, configuration and usage in compliance with the license of each third-party software component and with its proper software deployment and security policies.

AKKODIS INGENIERIE PRODUIT SAS shall not assume responsibility for bug or operating disruption caused by these third-party software components. Updated versions might be incorporated in future versions.

Overview

1 - Introduction

3D Juump Infinite is a software suite allowing the users to browse an entire DMU[1] in 3D on a standard PC[2].

It relies on a management/storage server (or **∞Directory**) and a network of relays (or **∞Proxy**) for publishing and presenting data.

On the side of the end user, the 3D Juump Infinite client application, that can be summarized as a DMU browser, is simply called **3D Juump Infinite**.

[1] Digital Mock-Up

[2] Personal Computer

2 - Definitions

In order to explain the design of 3D Juump Infinite, it is mandatory to first introduce several keywords related to the DMU and CAD products.

2.1 - Digital mock-up (DMU)

The digital mock-up (DMU) is composed of multiple elements:

`Digital mock-up`

A mock-up is a partial or complete representation of a system that allows to preview, test or validate its aspects or its behavior. A digital mock-up is built from computer files storing a tree of tridimensional geometries, displayed with a 3D rendering software, in our case, 3D Juump Infinite.

`Part`

A model or template, be it an assembly or a single element. This part is referenced by the digital mock-up but is not present (*instanced*) within it. It is not localized in the digital mock-up. Ex: a wheel.

`Part Instance`

One instance of a *part*, be it an assembly or a single element. This instance has a 3D representation located in the digital mock-up. Ex: the front-right wheel.

`Product Structure`

A hierarchy of *part instances*. Ex: a car with four wheels.

`Part Number`

The unique identifier of a *part*.

`Metadata`

Any textual or numeric information decorating a *part* or *part instance*, usually presented as a `key=value` pair. Ex: `provider='WheelEx Inc.'`.

`Annotation`

Any textual or graphical information decorating a *part* or *part instance*, represented in 3D as floating labels. Ex: contextual directions for assembly, tolerancing, etc.

`Effectivity`

A global parameter or option that changes the way the DMU should be assembled. Ex: What is the drivetrain of the car? Is this car a 4-wheel or a 2-wheel drive vehicle?

`Configuration`

A set of effectivities. A digital mock-up in a given configuration is also called a configured digital mock-up. Ex: a 4-wheel, metallic paint, 150 hp engine car.

2.2 - 3D Juump Infinite infrastructure

Other keywords are specific to the 3D Juump Infinite infrastructure:

Project

An incremental set of data describing a DMU. It usually corresponds to a product (or product-line) level assembly. For instance, a vehicle manufacturer would probably opt for one project per car model.

Build

An optimized and packaged snapshot of the DMU, ready for publication. The DMU is processed by 3D Juump Infinite back-end components and served to 3D Juump Infinite front-end client applications.

Geometry Pool

A pool of pre-processed geometric data that can be mutualized between projects.

Connector

A piece of software that processes DMU data from one or various source providers, and generates builds using the ∞CLI.

DMU data source provider

Any component of your *information system* able to publish DMU data encompassing the part geometry, metadata and/or effectivity. These components could be a conjunction of your CMS[3], CRM[4], your ERP[5], your PDM[6], your PLM[7] or simply a file system folder containing all your geometry part files extracted from your CAD[8] software.

Document

A document is a JSON[9] exchange file used by the *Connector* and the 3D Juump Infinite back end to transmit and store data bound to DMU *build* generation. JSON is an open, standard, human-readable text format used to transmit data objects consisting of attribute–value pairs. Thus, this format is also used by 3D Juump Infinite third-party softwares and the 3D Juump Infinite front end to exchange data.

[3] Content Management System

[4] Customer Relationship Management

[5] Enterprise Resource Planning

[6] Product Data Management

[7] Product Lifecycle Management

[8] Computer Aided Design

[9] JavaScript Object Notation

DMU flow

A chain of components of your enterprise and 3D Juump Infinite back-end components, in charge of the DMU extraction from *DMU data source providers*, the processing and the broadcasting of DMU *builds*.

User

A person that uses 3D Juump Infinite, either through its Web *Administration portal*, through a 3D Juump Infinite client application or through one of the provided APIs. All users must be properly authenticated before being authorized to use 3D Juump Infinite.

Authenticator

A third-party server in charge with user authentication. 3D Juump Infinite relies on an OpenID Connect identity layer to delegate authentication.

Administrator

A user that can log into the ∞Directory Web *Administration portal*. Administrators can be granted various levels of administration rights. Users that have exhaustively all administration rights are labeled super-administrators. A super-administrator is able to configure the *DMU flows*, to supervise the generation of DMU *builds*, to register *users* and edit their access rights individually or through their assignation to *teams*.

Tag

A keyword used to decorate a *build*, a *user* or a server/proxy component, that defines the access rights to the DMU.

Team

A set of users. It mainly acts as a helper concept that applies a common list of tags to its users.

Asset

Any product of a 3D Juump Infinite *user* who worked on any 3D Juump Infinite client application (bookmarks, visibility layers, export configurations…). Assets can be created, manipulated and shared amongst *users* thanks to the 3D Juump Infinite back-end components.

3 - Components

3D Juump Infinite is composed of several software entities.

∞Directory

The ∞Directory is in charge of data hosting and security management. It monitors and defines the whole 3D Juump Infinite network and is responsible for access rights management. This management is accessible via a *Web Administration portal* and programmable via the ∞*Directory API*.

∞Proxy

An ∞Proxy acts as a proximity relay for DMU broadcasting. It publishes *builds* from the ∞Directory. At least one ∞Proxy is required.

3D Juump Infinite client applications

The 3D Juump Infinite technology provides various services that revolve around the DMU. Various applications, that differ in the user experience and the functionalities they provide, can be used to access those services.

3D Juump Infinite Native client

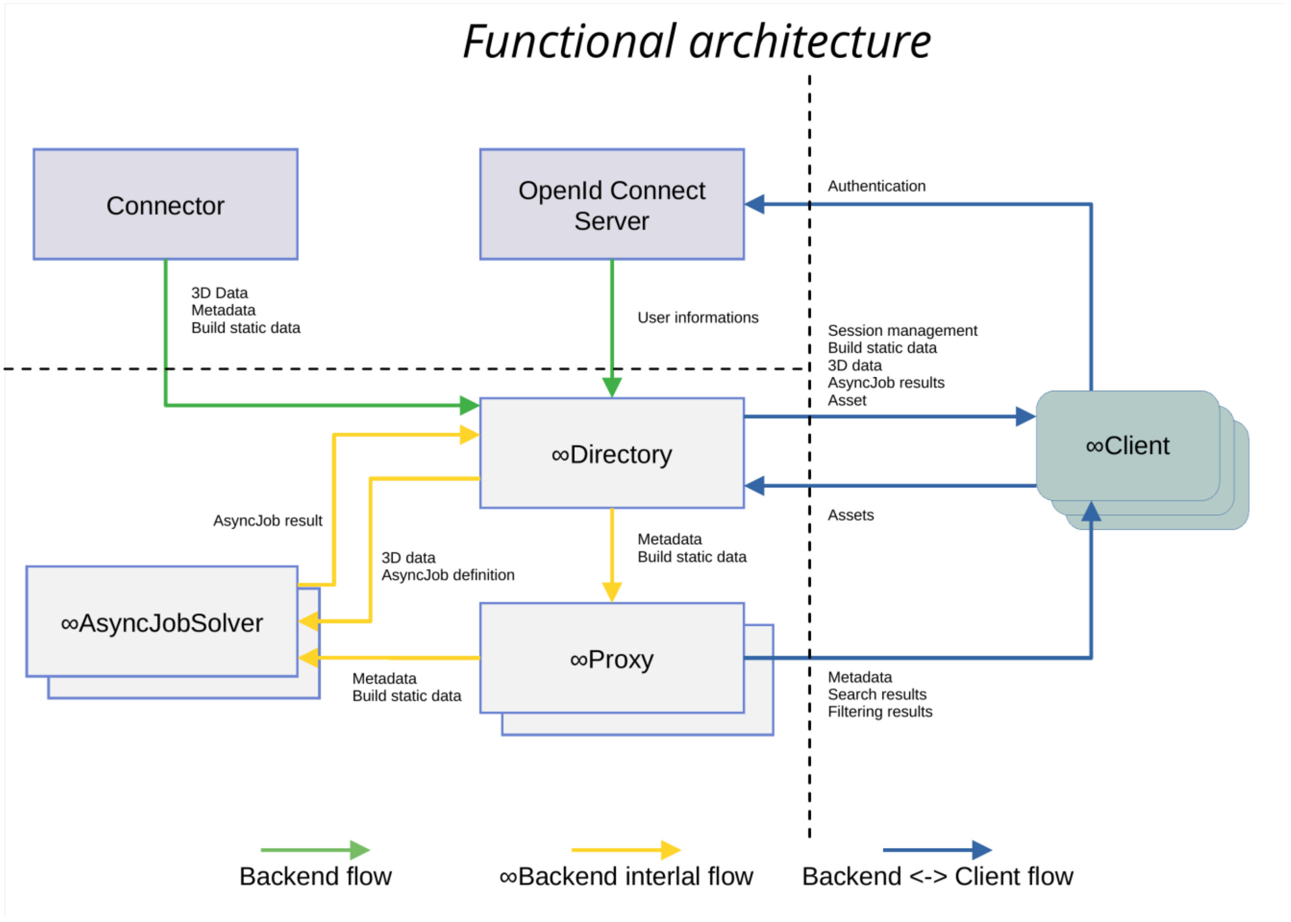
The “Native client” is a legacy native application that lets users browse the DMU with high performance and produce specific types of deliverables (geometric exports, image exports, slideshow, etc…). It is sometimes colloquially called *3D Juump Infinite*, when this name is unambiguous.

3D Juump Infinite Web applications

The 3D Juump Infinite technology can also power Web applications, that function inside a standard Web browser, making deployment easier. 3D Juump Yuzu and 3D Juump Kiwi, for instance, are Web applications powered by 3D Juump Infinite and developed by the 3D Juump team.

3D Juump Infinite Web API and custom applications

The 3D Juump Infinite technology comes with a library called the *Web API* that allows for the development of new custom Web applications, that can be fine-tuned for specific use cases.



Functional architecture

The previously presented software relies on several third-party components including databases and servers. In particular:

- a PostgreSQL service,
- an ElasticSearch service,
- an Apache HTTP service,
- an optional LM-X service.

PostgreSQL (or “Postgres”) is an SQL object-relational database management system (ORDBMS). It is used by the 3D Juump ∞Directory and ∞Proxy as the underlying database engine.

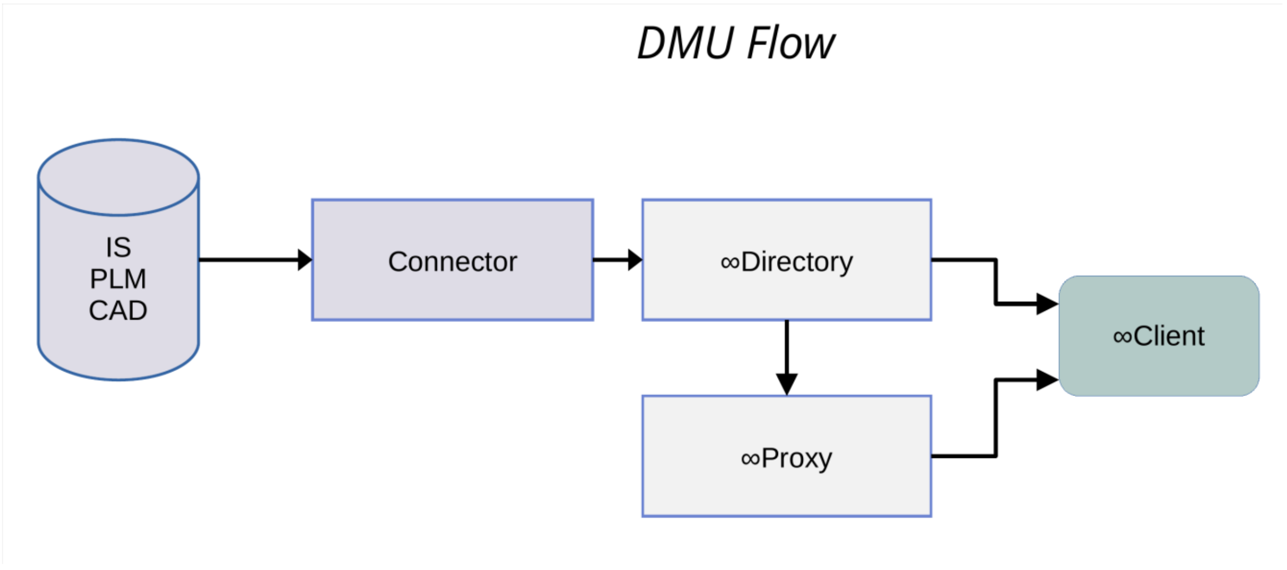
ElasticSearch is a distributed, multitenant-capable full-text search server with a RESTful Web interface and schema-free JSON documents. ElasticSearch is built on top of Apache Lucene, developed in Java and is released as open source under the terms of the Apache License. It provides full-text search capabilities to the 3D Juump Infinite client applications.

LM-X is a software licensing solution.

4 - DMU Flow

From the enterprise CAD data source to the 3D Juump Infinite client applications, the DMU follows several steps:

- First, your enterprise defined **Connector**, bound to your DMU data source providers, incrementally declares the 3D data to a *geometry pool* hosted on the ∞Directory using the ∞CLI PSConverter.
- Once all the 3D data is processed, the **Connector** can generate builds using ∞CLI generator build. The ∞CLI will retrieve the product structure using an ElasticSearch-like fetch query, that could be provided by ∞CLI generator docindexer, an ElasticSearch instance, or directly by a Web server embedded into the **Connector**.
- This *build* is then published on the **∞Directory**.
- Once finalized, the *build* will be published by **∞Proxies**.



DMU Flow

The Connector must be specified and implemented upon your requirements. 3D Juump Infinite only provides interfaces and helpers to feed the ∞Directory with such data. In the figure above, the enterprise DMU data source providers and your Connector are in gray.

For the sake of simplicity, we have described *one* DMU flow. 3D Juump Infinite lets you define *several* DMU flows, built upon your combination of Connectors, bound to your DMU data source providers and attached to the ∞Directory and several ∞Proxies.

Roles and architecture of cluster nodes

1 - ∞Directory

The ∞Directory is in charge of the security management and data storage. It monitors the whole 3D Juump Infinite network, from the ∞Proxies up to the 3D Juump Infinite client applications. None of these software entities is able to work without the explicit authorization of the ∞Directory. It is also responsible for access-rights management, though it delegates user authentication to a third-party identity service.

The ∞Directory relies on several software components:

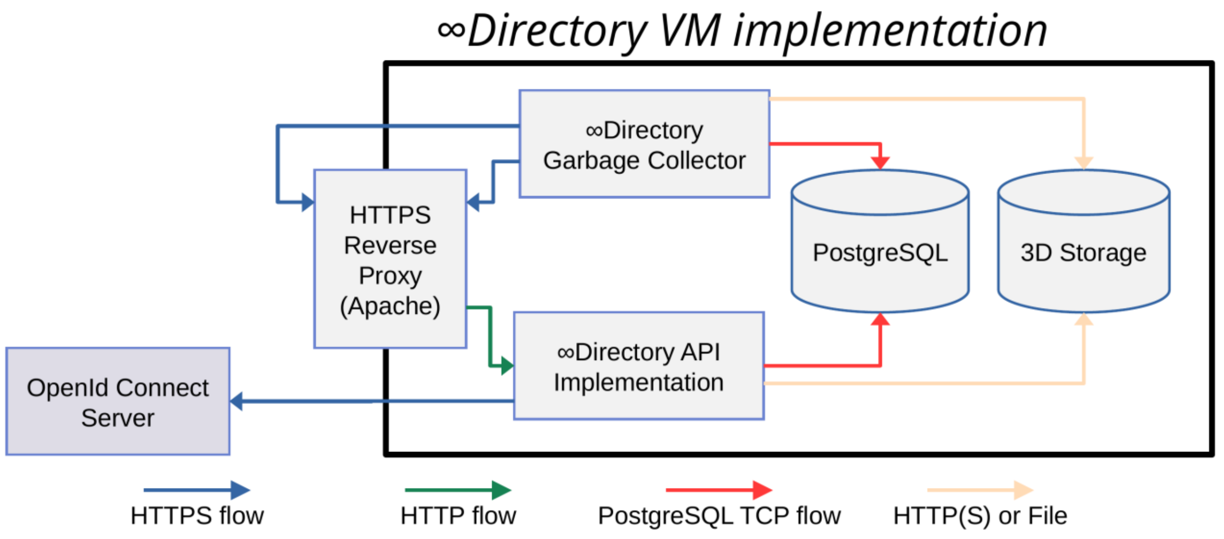
- a PostgreSQL server,
- an Apache HTTP server,
- a dedicated HTTP API implementation,
- an optional dedicated service.

The ∞Directory is operated thanks to a Web application, the *Administration portal*, built upon the **∞Directory HTTP API** (or *Directory API*), an HTTP-based REST API published by the ∞Directory. The use of the *Administration portal* is described in the administration chapter. The *Directory API* endpoints are described in a dedicated document.

The ∞Directory delegates user authentication through the *OpenID Connect* identity layer. Only authenticated users with proper access rights can operate the *Directory API* and the *Administration portal*, or open a DMU through one of the 3D Juump Infinite client applications and/or the *Directory API*.

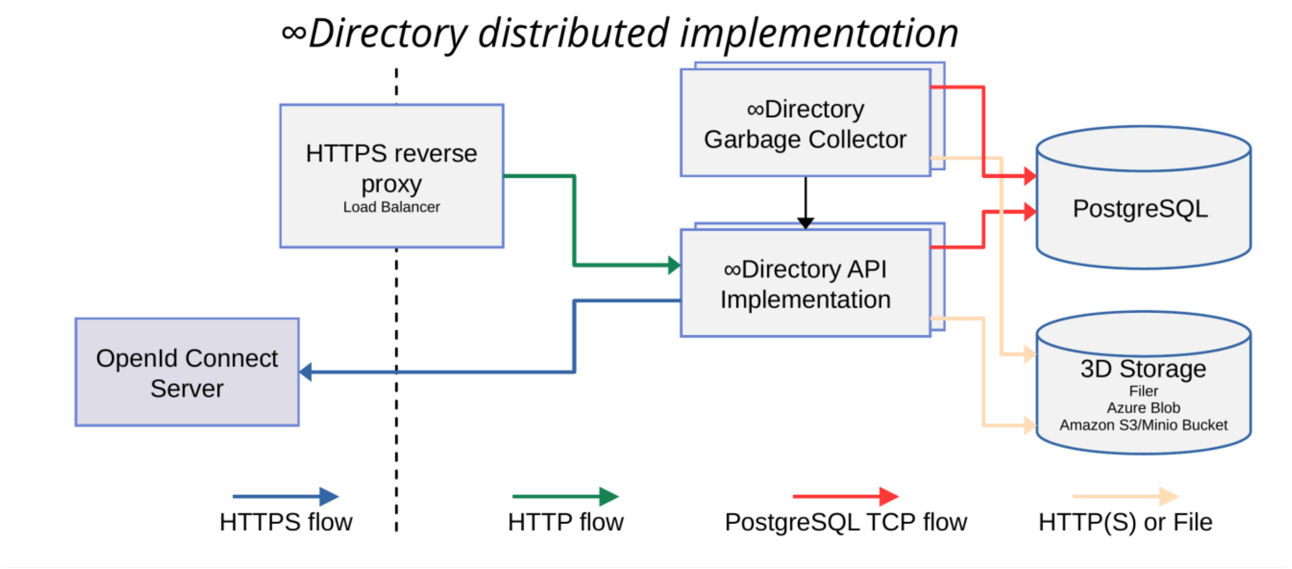
Databases located on the ∞Directory are not directly visible to client-side users. Instead, several *Directory API* endpoints are available and only accessible to authenticated users with proper access rights. Cluster-side server-to-server communication can be configured to rely either on HTTP or HTTPS.

By default the ∞Directory is run as a single virtual machine that can be installed using scripts provided within the release package (illustrated below).


∞Directory VM implementation
∞Directory Garbage Collector
HTTPS Reverse Proxy (Apache)
PostgreSQL
3D Storage
∞Directory API Implementation
OpenId Connect Server
HTTPS flow
HTTP flow
PostgreSQL TCP flow
HTTP(S) or File

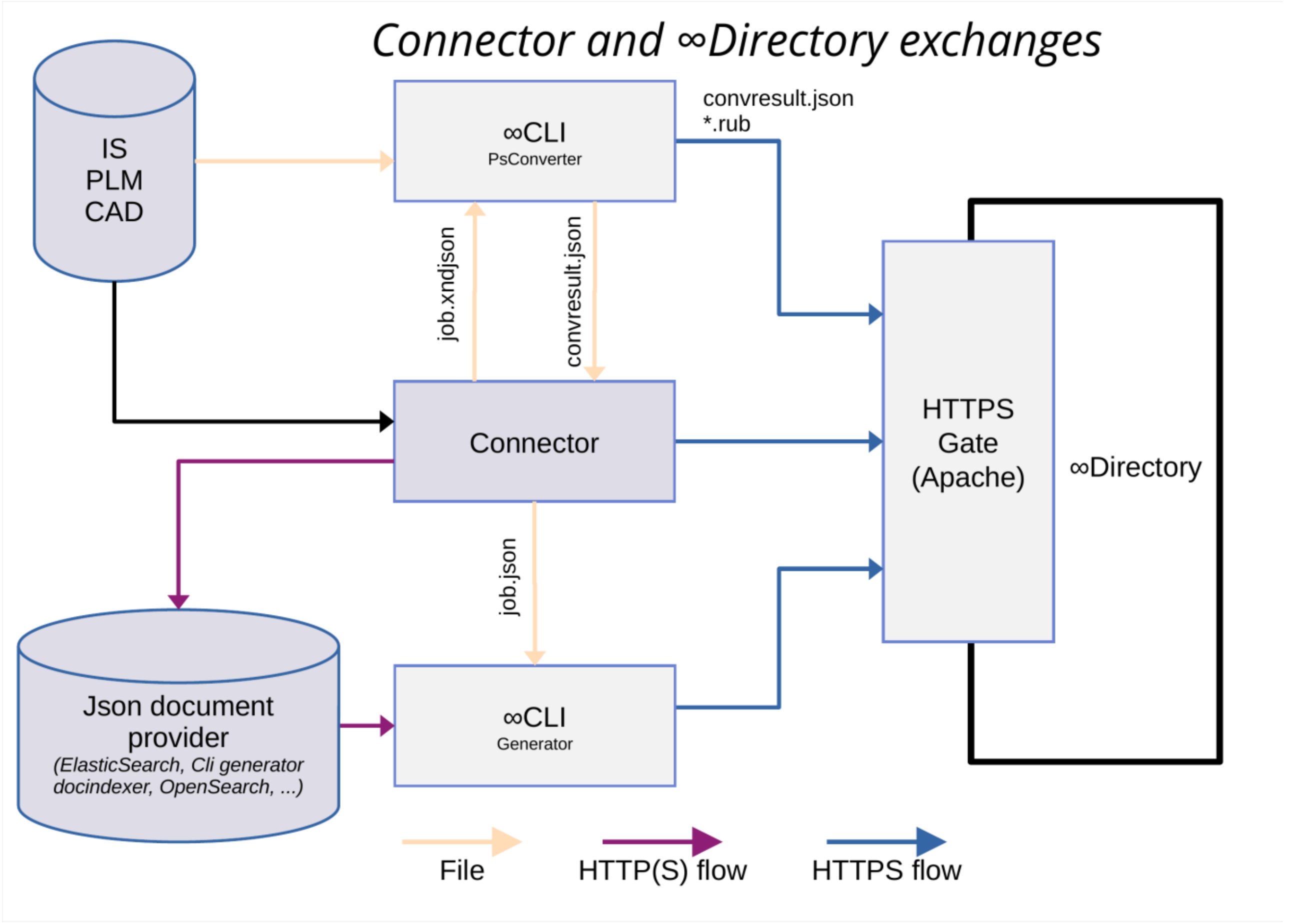
∞Directory

Alternatively, the ∞Directory can be run using distributed, redundant and scalable components (illustrated below). See.



∞Directory

The following diagram shows the dataflows between the Connector and the ∞Directory. The processing load of the generation can also be scaled by running the *PSConverter* and the *Generator* (*∞CLI generator*) processes on several servers concurrently.



Connector

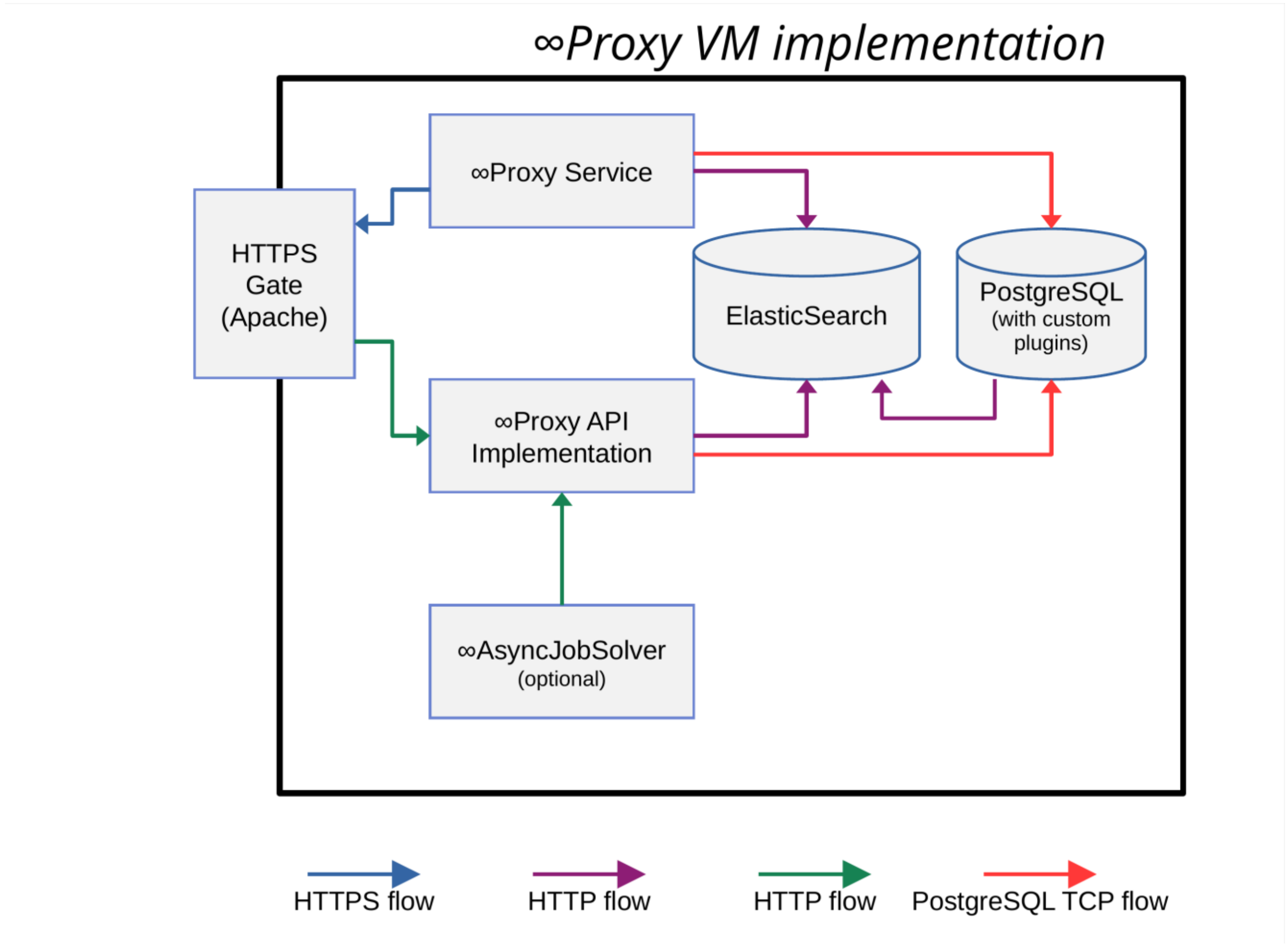
2 - ∞Proxy

Under the surveillance of the ∞Directory, an ∞Proxy acts as a proximity relay for DMU broadcasting. It publishes *builds* that were generated and stored on the ∞Directory. 3D Juump Infinite client applications connect to both the ∞Directory and ∞Proxies in order to browse DMUs. At least one ∞Proxy is needed to distribute DMU data.

The ∞Proxy relies on several software components:

- a PostgreSQL server,
- an ElasticSearch server,
- an Apache HTTP server,
- a dedicated HTTP API implementation,
- a dedicated service.

The ∞Proxy is implemented as a virtual machine containing all the components it needs to operate. Scalability and redundancy are obtained by allocating multiple virtual machines. The components of the ∞Proxy **should not be externalized or replaced** as they are optimized and customized to provide optimal performances.



∞Proxy

Databases that are located on the ∞Proxy are not directly visible to client-side users. Instead, the **∞Proxy HTTP API** (or *Proxy API*) provides several endpoints that are only accessible to authenticated users with proper access rights. Cluster-side server-to-server communication can be configured to rely either on HTTP or HTTPS.

3 - ∞AsyncJobSolver

An ∞AsyncJobSolver is an externalized unit responsible of solving asynchronous jobs requested by web applications. Each ∞Proxy as the ability to run an ∞AsyncJobSolver, however to increase capabilities ∞AsyncJobSolver might be instantiated on their own using the ∞Cli See. Note that ∞AsyncJobSolver running on a ∞Proxy will not be able to process asynchronous jobs associated to a data session that is not tied to this ∞Proxy.

If possible, it is recommended to instantiate ∞AsyncJobSolver dedicated to 2D export with dedicated GPU. If no GPU is available, rendering will be achieved using Mesa3D software renderer at a cost of low performances. Each instance of ∞AsyncJobSolver might be configured to process only specific job types.

Access rights to resources

This chapter explains the various policies relative to user access rights that are enforced at the ∞Directory and ∞Proxy level.

1 - Overview

3D Juump Infinite controls and restricts the access to:

- the different APIs (the ∞Directory API and the ∞Proxy API),
- the different nodes of the cluster,
- the DMU builds,
- portions of DMU builds,
- certain features exploiting the DMUs (export, download...),
- the assets stored on the ∞Directory,
- Web and native applications,
- the licenses.

Most of these access restrictions are configurable using tags. For instance, a user will only be able to access the builds that have tags matching his own.

2 - User

A user is identified through the OpenId Connect Server. A user can belong to teams.

3 - Teams

A team could be used to regroup users that will share same access rights or to mirror company organization.

4 - Tags

Tags are keywords that decorate the ***resources*** in the security model of 3D Juump Infinite. *Resources* can be:

- metadata documents in the product structure
- parts in the product structure
- an entire build,
- an ∞Proxy,
- a 3D Juump Infinite application or a third-party application registered on the ∞Directory.

A tag is a user-defined keyword. Please refer to the Access rights section for the exact allowed pattern.

5 - Access rights

Access rights are defined by lists of tags (or project names) regrouped under qualifying *namespaces*. *Namespaces* are predefined keywords, each corresponding to an access-controlled action. The available *namespaces* are listed and explained below. Access rights are applied to the ***consumers*** in the security model of 3D Juump Infinite:

- users
- teams
- access right collections
- applications

Please refer to the Rules chapter for the detailed mechanisms of how the access rights of teams and applications affect the access rights of the users. The access rights object is defined as follows:

Access rights

object

Lists of tags or projectids within qualifying namespaces.

- `affinity` : **array**, length (**[0;+∞]**), distinct

  A list of ∞Proxy tags. Defines on which proxies the user should be balanced in priority, if possible.

- items : **string**, length (**[1;64]**), pattern (`^[\x21\x23-\x39\x3c-\x5B\x5d-\x7e]+$`)

  Tag definition, because tags are also used as scopes limit them to scopes acceptable chars minus ':' and ';'. See https://www.rfc-editor.org/rfc/rfc6749#section-3.3.

- *`assetadmin`* : **array**

  Defines the projects on which the user is granted asset administrator rights.

  - items : **string**, length (**[36;36]**), pattern (`^prj_[a-f0-9]{32}$`)

    The unique identifier of the project.

- *`download`* : **array**, length (**[0;+∞]**), distinct

  A list of resource tags. Defines which builds the user can download in the Local DMU Manager. To allow the download, all the tags of the build (base and extra) must be granted.

  - items : **string**, length (**[1;64]**), pattern (`^[\x21\x23-\x39\x3c-\x5B\x5d-\x7e]+$`)

    Tag definition, because tags are also used as scopes limit them to scopes acceptable chars minus ':' and ';'. See https://www.rfc-editor.org/rfc/rfc6749#section-3.3.

- *`export2d`* : **array**, length (**[0;+∞]**), distinct

  A list of resource tags. Defines product structure resources from which the user can generate 2D exports. To allow the export, all the tags used when opening the datasession to view the build must be granted.

  - items : **string**, length (**[1;64]**), pattern (`^[\x21\x23-\x39\x3c-\x5B\x5d-\x7e]+$`)

    Tag definition, because tags are also used as scopes limit them to scopes acceptable chars minus ':' and ';'. See https://www.rfc-editor.org/rfc/rfc6749#section-3.3.

- *`export3d`* : **array**, length (**[0;+∞]**), distinct

  A list of resource tags. Defines product structure resources from which the user can generate 3D exports. To allow the export, all the tags used when opening the datasession to view the build must be granted.

  - items : **string**, length (**[1;64]**), pattern (`^[\x21\x23-\x39\x3c-\x5B\x5d-\x7e]+$`)

    Tag definition, because tags are also used as scopes limit them to scopes acceptable chars minus ':' and ';'. See https://www.rfc-editor.org/rfc/rfc6749#section-3.3.

- *`thirdpartyscopes`* : **array**, length (**[0;+∞]**), distinct

  A list of scopes. Defines which third application scopes are granted to a user.

    - items : **string**, length (**[1;64]**), pattern (`^[\x21\x23-\x39\x3c-\x5B\x5d-\x7e]+$`)

      Tag definition, because tags are also used as scopes limit them to scopes acceptable chars minus ‘:’ and ‘;’. See https://www.rfc-editor.org/rfc/rfc6749#section-3.3.

- `use` : **array**, length (**[0;+∞]**), distinct

  A list of application tags. Defines which applications (native and/or Web) the user is allowed to use.

    - items : **string**, length (**[1;64]**), pattern (`^[\x21\x23-\x39\x3c-\x5B\x5d-\x7e]+$`)

      Tag definition, because tags are also used as scopes limit them to scopes acceptable chars minus ‘:’ and ‘;’. See https://www.rfc-editor.org/rfc/rfc6749#section-3.3.

- `view` : **array**, length (**[0;+∞]**), distinct

  A list of resource tags. Defines which product structure resources the user is allowed to browse. To open a build, at least the base build tags must be granted. Resources protected by optional tags will be filtered out if the user does not have view rights on them.

    - items : **string**, length (**[1;64]**), pattern (`^[\x21\x23-\x39\x3c-\x5B\x5d-\x7e]+$`)

      Tag definition, because tags are also used as scopes limit them to scopes acceptable chars minus ‘:’ and ‘;’. See https://www.rfc-editor.org/rfc/rfc6749#section-3.3.

6 - Access right collection

An access right collection is a way to regroup multiple access rights that will be used on multiple ***consumers***.

7 - Data-level access rights

Unlike the other resources, builds are protected by two tag lists:

- a `base` tag list: it is the minimal required set of tags needed to access this build.
- an `extra` tag list: it is the list of extra tags that are used to restrict access to subsets of the build (metadata, product structure, annotations), referred to as *data-level filtering*.

The following resources can be decorated with tags by the *Connector* to enable data-level filtering:

- metadata documents
- instantiation links into structure documents
- attached documents
- annotation documents

- configuration definitions

The data corresponding to the protected resource will be filtered out by the ∞Directory HTTP API and will never be accessible to a user who does not have the corresponding extra tags used to protect it under his `view` access rights.

⚠ Please read the following limitations concerning data-level access rights.

- The metadata field name will be public, only the value of the field will be filtered!
- ID numbering is consecutive. Holes in the ID numbering could allow to determine that one or several parts of the product structure have been filtered!
- Configured metadata into metadata documents **ARE NOT** filtered server side and are not part of the data-level access right feature!
- Customization scripts **ARE NOT** filtered and should not contain sensitive information!

8 - Rules

Access rights management is enforced by applying a set of strict tag-matching rules.

8.1 - Access rights inheritance and validation

During the identification process, the user will inherit access rights, that will be checked in the following order:

- The user initiates a session for a particular application.
- The user inherits the access rights of all the teams he belongs to.
- Access to the application is checked by using the access rights `use`. Access to the application is granted if the application has no tags, or if all its tags are contained into the user's `use` access rights list.
- The user inherits the access rights of the application.
- The user lists builds/opens a build. Only builds of which all the base tags are contained into the user's `view` access rights list are listed and can be opened.

To set access rights on an application, it must be protected by at least one tag. This is a security rule, otherwise any enabled user would be able to use this application and inherit those access rights.

⚠ When saving assets from an application, the application tags are not transferred to the security tags of the asset. This means that application tags do not protect the data produced by an application, only the access to the application itself and the application's secure resources.

8.2 - User using an application

A user can only use an application if he bears all the matching access rights: for each *`keyword`* tag of the target application, the user must have a matching **`use`**`:`*`keyword`* access right.

8.3 - User connection to a proxy

A user will be balanced in priority to proxies with matching *`affinity`* tags.

8.4 - User access to a build

A user can access a feature of a build if he bears all the matching tags.

- Visualisation feature: for each *`keyword`* base tag of the target build, the user must have a matching ***`view:`****`keyword`* access right.
- Screenshot feature: for each *`keyword`* base tag of the target build, the user must have a matching "***`export2D:`****`keyword`* access right.
- Export feature: for each *`keyword`* base tag of the target build, the user must have a matching "***`export3D:`****`keyword`* access right.
- Download feature: for each *`keyword`* base and extra tag of the target build, the user must have a matching "***`download:`****`keyword`* access right.

8.5 - Administration of user assets

A user can administrate assets pertaining to a project only if he bears a matching ***`assetadmin:`****`prj_xxx`* access right, where *`prj_xxx`* is the project id.

8.6 - User access rights of third-party applications

Third-party backends may use directory session bearers to limit actions of users that are authenticated on the ∞Directory (using token validation or the *`directory/api/introspect`* end point of the ∞Directory API). Using the ***`thirdpartyscopes`*** namespace, it is possible to define application scopes to finely manage access rights. If a user has ***`myapplicationA`*** in its ***`thirdpartyscopes`*** access rights, and if this scope is claimed during the ∞Directory session creation, the access token will contain *`myapplicationA`* in its *`scope`* field. Note that the tag value will be percent encoded. See Third-party application protection for more information.

HTTP API security

This chapter explains the authorization mechanisms of the ∞Directory and ∞Proxy HTTP APIs.

1 - Overview

The access to the ∞Directory and ∞Proxy HTTP APIs is restricted by security schemes. The API documentations, referencing API endpoints and data schemas, is available in separate documents of the manual: “*http api/3D Juump Infinite [Directory|Proxy|CLI] API documentation [version number].html*”. A detailed view of the required authorization methods per API endpoint is available in “*3D Juump Infinite HTTP API security summary table [version number].html*”.

In the chapter Security schemes and scopes, you will find a list and descriptions of the security schemes made available by 3D Juump Infinite.

2 - Virtual hosts

2.1 - Public virtual host

For 3D Juump Infinite components, the public virtual host corresponds to its public URL, the one that will be used by client applications.

2.2 - Private virtual host

Private virtual hosts are alternative URLs that can be used to route backend traffic on a network that is faster, or that implements different security rules. Private VHosts cannot be configured to accept traffic from client applications, with the exceptions of the *Administration portal* and the *Home Page* applications.

Using backend VHosts, it is possible to add extra layers of security, like certificate authentication (mTLS) or IP filtering for certain types of traffic based on *security scheme* filtering. See Directory virtual host and Proxy virtual host for more information.

For instance, traffic coming from the Connector could be restricted to a dedicated virtual host *connector.mydirectory.com*, on which mTLS is enabled on the HTTP gate. Sample of the ∞Directory configuration file in this scenario:

```
{
    "directoryapi": {
        "publicvhost": {
            "securityschemes": {
                "blacklist": {
                  "http.m2m_bearer":["infinite.connector"]
                 },
                "whitelist": null
            },
            "url": "https://mydirectory:443/directory"
        },
        "backendvhosts": [
            {
                "securityschemes": {
                    "blacklist":null,
                    "whitelist": {
                        "http.m2m_bearer":["infinite.connector"]
                    }
                },
                "url": "https://connector.mydirectory:443/directory"
            }
        ],
        ...

    }
}
```

3 - Security schemes and scopes

3.1 - http.directory_key

Basic authentication using the master key of the ∞Directory API.

location: http header *Authorization: Basic base64(infinite:{API_KEY})*.

3.2 - http.proxy_key

Basic authentication using the master key of the ∞Proxy API.

location: http header *`Authorization: Basic base64(infinite:{API_KEY})`*.

3.3 - http.session_bearer

An HTTP bearer. Depending on the configuration of the ∞Directory, it can be an OpenId Connect access token or an infinite session bearer.

location: http header *`Authorization: Bearer {BEARER}`*.

3.4 - http.m2m_bearer

An HTTP bearer provided by the OAuth2 server using the Client Credentials Flow.

location: http header *`Authorization: Bearer {BEARER}`*.

Configurable scopes:

- *`infinite.directory`*: Protects endpoints needed by the ∞Directory.
- *`infinite.proxy`*: Protects endpoints needed by the ∞Proxy.
- *`infinite.connector`*: Protects endpoints needed by the connector (PSConverter, Generator, EvoJuump backup/restore).
- *`infinite.jobsolver`*: Protects endpoints needed by ∞AsyncJobSolver (∞CLI, ∞Proxy).
- *`infinite.asset`*: Allows to manage asset storage.
- *`infinite.admin.common`*: Refer to infinitebearer.directory_session[admin.common].
- *`infinite.admin.full`*: Refer to infinitebearer.directory_session[admin.full].
- *`infinite.admin.user`*: Refer to infinitebearer.directory_session[admin.user].
- *`infinite.admin.app`*: Refer to infinitebearer.directory_session[admin.app].

3.5 - infinitebearer.directory_session

An infinite bearer delivered by the ∞Directory when initiating a directory session.

location: http header *`x-infinite-bearer: {BEARER}`*.

Configurable scopes:

- *`base`*: Granted to all directory sessions.
- *`admin.common`*: Administration privileges that are granted to every administrator.
- *`admin.full`*: Super-admin privileges, allows to appoint other super administrators, modify the server configuration and manage data.
- *`admin.user`*: User-level administration privileges, allows to manage users and teams, with limitations on super-administrators.
- *`admin.app`*: Application-level administration privileges, allows to manage applications.
- *`client`*: Grants access to client applications.

3.6 - infinitebearer.directory_session_download_token

An infinite bearer delivered by the ∞Directory to download a resource using a secured URL. This token has a really short lifetime.

location: http query parameter *`x-infinite-download-token={BEARER}`*.

Configurable scopes:

- *`client`*: Refer to infinitebearer.directory_session.
- *`admin.full`*: Refer to infinitebearer.directory_session.
- *`admin.app`*: Refer to infinitebearer.directory_session.

3.7 - infinitebearer.directory_session_extended

An extended infinite bearer delivered by the ∞Directory when initiating a directory session.

location: http header *`x-infinite-bearer: {BEARER}`*.

3.8 - infinitebearer.data_session

An infinite bearer delivered by the ∞Directory when initiating a data session.

location: http header *`x-infinite-bearer: {BEARER}`*.

Configurable scopes:

- *`base`*: Granted to all data sessions.
- *`download`*: Granted if the user is allowed to download the whole build.
- *`export2d`*: Granted if the user is allowed to generate 2D exports. Note that the restriction could be bypassed on native client side.
- *`export3d`*: Granted if the user is allowed to generate 3D exports. Note that the restriction could be bypassed on native client side.

3.9 - infinitebearer.data_session_download_token

An infinite bearer delivered by the ∞Directory to download a resource using a secured URL. This token has a really short lifetime.

location: http query parameter *`x-infinite-download-token={BEARER}`*.

Configurable scopes:

- *`base`*: Granted to all data sessions.
- *`download`*: Refer to infinitebearer.data_session.

3.10 - infinitebearer.data_session_extended

An extended infinite bearer delivered by the ∞Directory when initiating a data session.

location: http header *`x-infinite-bearer: {BEARER}`*.

3.11 - infiniteprivate

An internal bearer used to protect calls reserved to infinite tools. This bearer has a really short duration and is locked on a full URL.

location: http header *`x-infinite-private: {BEARER}`*.

Cluster installation

1 - Installation of Infinite services

This chapter will present how to install an ∞Directory or an ∞Proxy.

1.1 - Requirements

Each 3D Juump Infinite machine must be able to contact (e.g. ping) the ∞Directory host, which is the central point of the 3D Juump Infinite architecture. Your network administrator must provide you with one or several machine(s) that can be connected to each other (in the network sense).

Before starting the installation procedure, you should retrieve:

- The software delivery package containing Windows installers, install scripts, configuration files and this documentation
- At least one machine to host your ∞Directory / ∞Proxy
- **Administrator** rights on all the machines
- Python version 3.10 or newer with (`requests`, `PyYaml` and `jsonschema` modules)
- For Windows, `wmic` tool.
- For Linux, an internet access to the 3D Juump APT repository from all the machines.

- Optionally (but strongly recommended), an SSL certificate, such a certificate may be generated by your own means
- Optionally, setup or use an existing LM-X server, configured with the 3D Juump vendor extension (such a server can be installed on the same host as the ∞Directory [LM-X])

⚠ All the logins/passwords must be limited to alphanumerical characters and these separators: space ' ', minus'-', underscore'_' and dot '.' [10].

You might require to temporarily disable your antivirus during the installation procedure. We need to modify the Windows firewall rules during installation. Some antivirus software (e.g. McAfee) may detect this as a potential threat.

1.1.1 - Server requirements

1.1.1.1 - ∞Directory, ∞Proxy and ∞AsyncJobSolver

The ∞Directory and ∞Proxy softwares run on any of the following operating systems with an **IPv6 stack** and x86-64 CPU architecture :

Operating system	System Service	Docker (Debian 13 (trixie) base images)
Microsoft Windows 11	Yes	Partial using Docker Desktop (performance issues with mounted volumes). Not recommanded due to Docker Desktop instabilities
Linux Debian 12 (bookworm)	Yes (deprecated)	Yes using docker-compose
Linux Debian 13 (trixie)	Yes	Yes using docker-compose
Linux Ubuntu 22.04 LTS (jammy)	No	Yes using docker-compose
Linux RedHat 9.4 (plow)	No	Yes using docker-compose

Other operating system might be supported using docker images, however deployment is not supported out of the box and has to be fully handled by the customer.

Minimum hardware requirements are:

- Quad-core processor (support of POPCNT x86 instruction is mandatory, support of CRC32 x86 instruction is recommended)
- For the ∞Directory and ∞Proxy: 8GB of RAM
- For the ∞AsyncJobSolver: 4GB of RAM
- For the ∞AsyncJobSolver: 8GB of disk space per worker for temporary cache
- 2GB disk space for binaries + sufficient disk space for data (depending on your data sources)
- High-speed hard drive disk or solid-state drive highly recommended

[10] For information, the matching regex is "[a-zA-Z0-9_-.]+".

- 1Gbit/s network connection between the ∞Directory and ∞Proxy
- For the ∞AsyncJobSolver: on Linux EGL 1.5 is required with at least a surfaceless platform, on Windows an hardware able to run Mesa3D
- File system supporting files of 4GB+ (NTFS, Ext4, …)
- An OpenGL implementation supporting at least OpenGL 3.0 and following OpenGL extensions (GL_ARB_vertex_array_object, GL_ARB_draw_instanced, GL_ARB_instanced_arrays)

Recommended hardware requirements are:

- Octo-core processor (support of POPCNT x86 instruction is mandatory, support of CRC32 x86 instruction is recommended)
- For the ∞Directory: 12GB of RAM
- For the ∞Proxy: 16GB of RAM
- For the ∞AsyncJobSolver: 2GB of RAM per CPU core
- For the ∞AsyncJobSolver: a dedicated GPU with 4GB of RAM

1.1.1.2 - ∞CLI: PSConverter

To obtain the best performances with the *PSConverter*, it is advised to batch as many jobs as possible per call to the *PSConverter*.

The worker count parameter has to be adjusted depending on the available CPU resources, memory resources and the complexity of the data.

Minimum hardware requirements are:

- Quad-core processor (support of POPCNT x86 instruction is mandatory, support of CRC32 x86 instruction is recommended)
- 1.5GB of available RAM per worker
- High-speed hard drive disk or solid-state drive highly recommended
- 1Gbit/s network connection to the ∞Directory

Recommended hardware requirements are:

- Octo-core processor (support of POPCNT x86 instruction is mandatory, support of CRC32 x86 instruction is recommended)
- 3GB of available RAM per worker
- High-speed hard drive disk or solid-state drive highly recommended

1.1.1.3 - ∞CLI: Generator

Depending on the complexity and volume of the data, the recommended hardware requirements might need to be increased.

The duration of the generation process will be decreased by adding more CPU cores or other resources.

Minimum hardware requirements are:

- Quad-core processor (support of POPCNT x86 instruction is mandatory, support of CRC32 x86 instruction is recommended)
- 8GB of available RAM
- High-speed hard drive disk or solid-state drive highly recommended
- 1Gbit/s network connection to the ∞Directory

Recommended hardware requirements are:

- Octo-core processor (support of POPCNT x86 instruction is mandatory, support of CRC32 x86 instruction is recommended)
- 16GB of available RAM per worker
- High-speed hard drive disk or solid-state drive highly recommended

1.2 - Versions of third-party services

3D Juump Infinite installs and/or uses third-party services. Here is a table that lists the expected versions for these components.

Third-party Software	∞Directory	∞Proxy
PostgreSQL	16.11	16.11
ElasticSearch	-	9.2.3
Java JRE Server	-	22.0.2
Apache	2.4.66	2.4.66
LM-X Server	4.9.15-2025	-

1.3 - Installation scenarios

Depending on your needs we provide the following installation scenarios:

- **directory**: will install an ∞Directory on the machine.
- **proxy**: will install an ∞Proxy on the machine.
- **directory and proxy**: will install an ∞Directory and an ∞Proxy on the machine.

The folder `install form` of the software delivery package contains the `install.py` install script and a form template `form_install_info.yaml.tpl`. First, create an install form based on `form_install_info.yaml.tpl` and fill in all the mandatory entries. You are free to update entries that are pre-filled. You will find a description of each entry in `doc_form_install_info.html`. Once you have finished editing the YAML file, open a terminal with **Administrator** rights and run the corresponding `install.py` python script. Then follow the instructions. This new form file you have filled can be provided as the first argument of the `install.py` script, otherwise the script will search for a file named `form_install_info.yaml` in `.` and `../..`.

All the install scripts support the following options:

- `-f`: force answer yes to all user input
- `--accept-eula`: automatically accept the EULA (end-user license agreement)

Notes: You may provide your own SSL certicate and private key (in `X509` format without password protection) for the TLS communication. This certificate should match the declared public URL in the *form_xxxxxxxx.yaml* file. You may then proceed with the installation and override the autogenerated certificate and private key with your own files.

1.4 - Docker installation

As an alternative to installing the service directly on the operating system, it is possible to use Docker. It allows to run different versions of the 3D Juump Infinite services on the same server. This installation scenario will not handle the HTTPS gate configuration. It will only provide an example of configuration in the install folder, it will be up to you to expose the ∞Directory and ∞Proxy API through an HTTP reverse proxy.

⚠ For production, use Docker on a Linux operating system. Docker on Windows or MacOs will have poor performances and should only be used for testing.

1.5 - Custom/Distributed installation

This chapter is dedicated to custom deployment that will not rely on provided script. It enumerates constraints that have to be respected. Implementation details could be found in provided docker compose installation scenario.

1.5.1 - Configuration files

Configuration files should be prepared using ∞Cli, see :

- ∞Directory configuration file
- ∞Proxy configuration file
- ∞Cli directory conf *
- ∞Cli proxy conf *.

1.5.2 - ∞Directory

∞Directory uses a standard PostgreSQL and could use a mutualized/managed one. It is up to you to implement load balancing, redundancy and dynamic scalability.

Resource requirements :

- Network access to PostgreSQL
- Network access to all ∞Proxy through public or backend vhost
- Read access to configuration file (consolidated)
- Read/Write access to filer storage (network or file system)

Startup sequence :

- Start PostgreSQL
- After first install or binary update : run ∞Cli directory init see
- Start ∞Directory garbage collector instances (at least one) see
- Start ∞Directory http api instances (at least one) see

1.5.3 - ∞AsyncJobSolver

Resource requirements :

- Network access to ∞Directory through public or backend vhost
- Network access to all ∞Proxy through public or backend vhost
- Read access to configuration file (consolidated)
- Read/Write access to working folder (file system)

Startup sequence :

- Start ∞Cli asyncjob see

1.5.4 - ∞Proxy

∞Proxy components (service, PostgreSQL, ElasticSearch) should not be split, externalized or shared. Scalability and redundancy should be achieved by instantiating new ∞Proxy. ∞Proxy MUST have total control on its PostgreSQL and ElasticSearch. PostgreSQL instance MUST runs with *3DJuump Infinite Postgres Plugins*.

Resource requirements :

- Network access to ∞Directory through public or backend vhost
- Read access to configuration file (consolidated)
- Read/Write access to working folder (file system)

Startup sequence :

- Start local PostgreSQL
- Start local ElasticSearch
- After first install or binary update : run ∞Cli proxy init see
- Start ∞Proxy service (3dJuumpInfiniteProxyService)

1.6 - Test of the ∞Directory Administration portal and license requests

For any scenario involving the installation of an ∞Directory, once the installation is finished, the *∞Directory Administration portal* can be accessed through *https://directory_name:${apache_https_port}/directory/admin* (see Login in the Web Administration portal). Then go to the 'Licenses' panel to download the license request file.

The Web Administration portal doesn't display? Check if the ∞Directory service has started and is approved by your network firewall rules.

2 - OpenID Connect

User authentication is deferred by the ∞Directory to a third-party server using the OpenID Connect code flow protocol. The OpenID configuration is specified during installation in *form_*.yaml*. Detailed descriptions of each field are available in */manual/api_and_service_conf/directory_conf.html*.

2.1 - Configuration

To properly configure the OpenID delegation, you have to request the creation of an *Application* to your ID provider. This *Application* redirect URI will be *https://host:443[/prefix]/directory/api/directorysession/onauthenticated*. In exchange you should obtain:

- A configuration URL OpenID Provider configuration URL
- An application ID
- An application secret
- Whether PKCE should be enabled or not
- Which kind of JWT signing algorithm will be used

See OIDC common settings, OIDC user identification for the ∞Directory and OIDC user identification for the ∞Proxy for configuration details.

2.2 - User identification

When opening a directory session on the ∞Directory, the user is identified using the OpenID Connect code flow protocol. An id_token is retrieved. Optionally, an access_token and a refresh_token could be requested depending on the *use_oidc_access_token* setting.

2.3 - ID token usage

The ID token delivered by the OpenID server is used to identify the user. The following standard fields are extracted from the id_token: *sub*, *name*, *given_name*, *family_name*, *middle_name*, *nickname*, *preferred_username*, *profile*, *picture*, *email*, *email_verified*, *zoneinfo*, *locale*, *phone_number*, *phone_number_verified*, *address*, *updated_at*.

Those fields are provided through the *directory/api/manage/users* api and are used to compute the user's display name. If delivered, the id_token contains non standard fields that can be mapped to standard fields by using the *id_token_alias* setting.

2.4 - Authentication webhook

An optional authentication webhook can be configured using the *authentication_webhook* setting. This webhook will be called before authorizing each new directory session. It could be used to update user credentials using the directory API. The webhook should expect to receive an HTTP POST request with a JSON payload ("Content-Type: application/json"):

```json
{
    "sub":"...",
    "access_token":"...",
    "challenge":"..."
}
```

- The *sub* field will contain the OpenID Connect sub id.
- The *access_token* field will contain the access token delivered by the OpenID Connect server.

- The *challenge* field specifies the challenge that must be returned in the webhook response.

To authorize the user, the webhook should respond with an HTTP response code 200 and a JSON payload (“Content-Type: application/json”):

```
{
    "challenge":"...",
    "authorized":"true"
}
```

2.5 - Access Token usage

It is possible to protect the ∞Directory and ∞Proxy APIs using access_tokens delivered by the OpenID server. This feature is enabled by *use_oidc_access_token* setting. When enabled, the OpenID server should deliver a JWT access_token and a refresh_token. The access_token will be sent to the client, the refresh_token will be kept on the ∞Directory to renew the access_token. Access token scopes can be customized using the *additional_scopes* setting. The first access_token will be retrieved using *additional_scopes/primo_token* and will only be sent to the authentication webhook. Client access_tokens will be retrieved using *additional_scopes/client_token*. The *oidc_access_token_aud* setting can be used to specify which audience should be expected in access_tokens.

If *use_oidc_access_token* is disabled, the ∞Directory and ∞Proxy APIs will be protected using internal tokens.

2.6 - Non standard query parameters

Custom query parameters can be added to OpenID endpoints calls using *additional_query_parameters*.

2.7 - Machine to Machine (M2M)

As an alternative to using an *API key* with basic HTTP authentication, it is possible to use OAuth2 access tokens to protect communications between the ∞Directory, the ∞Proxies, the ∞CLI and any external tool. Access token will be retrieved using OpenID Connect client credentials flow. The list of scopes that could be requested is listed in Security schemes and scopes. See OIDC common settings and OIDC Machine to Machine identification for configuration details.

2.8 - Example

Configuration example for Azure Active Directory:

```
{
    "openidconnect": {
        "common": {
            "configuration_endpoint": "https://login.microsoftonline.c
om/**************/v2.0/.well-known/openid-configuration",
            "http_client_configuration": {
                "client_certificate": false,
                "http_proxy": false,
                "verify_ssl_peer": true
            }
        },
        "m2m_auth": null,
        "user_auth": {
            "additional_query_parameters": {
                "authorization_endpoint": {},
                "revocation_endpoint": {},
                "token_endpoint": {}
            },
            "additional_scopes": {
                "client_token": "***************/.default",
                "primo_token": "User.Read Group.Read.All offline_acces
s"
            },
            "allowed_jwt_alg": [
                "RS256"
            ],
            "authentication_webhook": {
                "http_client_configuration": {
                    "client_certificate": null,
                    "http_proxy": null,
                    "verify_ssl_peer": null
                },
                "url": "https://127.0.0.1:444/authentication_webhook_c
gi.py"
            },
            "client_id": "***************",
            "client_secret": "***************",
            "hmac_secret": null,
            "id_token_alias": {},
            "token_aud": "***************",
            "token_azp": null,
            "token_iss": false,
            "use_PKCE": true,
            "use_oidc_access_token": true,
            "user_unique_id": "oidc"
        }
    }
}
```

Configuration example for Keycloak:

```
{
    "openidconnect": {
        "common": {
            "configuration_endpoint": "https://***************.com:443
/realms/***************/.well-known/openid-configuration",
            "http_client_configuration": {
                "client_certificate": false,
                "http_proxy": true,
                "verify_ssl_peer": true
            }
        },
        "m2m_auth": {
            "allowed_jwt_alg": [
                "RS256"
            ],
            "client_id": "***************",
            "client_secret": "***************",
            "token_aud": "***************"
        },
        "user_auth": {
            "additional_query_parameters": {
                "authorization_endpoint": {
                    "prompt": "login"
                },
                "revocation_endpoint": {},
                "token_endpoint": {}
            },
            "additional_scopes": {
                "client_token": "***************",
                "primo_token": ""
            },
            "allowed_jwt_alg": [
                "RS256"
            ],
            "authentication_webhook": null,
            "client_id": "***************",
            "client_secret": "***************",
            "hmac_secret": null,
            "id_token_alias": {},
            "token_aud": "***************",
            "token_azp": null,
            "token_iss": null,
            "use_PKCE": true,
            "use_oidc_access_token": true,
            "user_unique_id": "oidc"
        }
    }
}
```

3 - LM-X server (optional, for borrow support)

3.1 - Run an LM-X server (for borrow support)

An LM-X server is needed only to enable the support of the 'borrow' functionality within the local DMU Manager.

If you do not already have an available LM-X server (version 4.9.15-2025), please run the corresponding installer (provided in the installation package) and refer to the LM-X documentation. Provide the 3D Juump vendor extension file (*liblmxvendor.so* on Linux or *liblmxvendor.dll* on Windows) when required during the installation.

Note: it is not necessary that the LM-X server runs on the same physical computer as the ∞Directory as long as it is visible (in a network sense) by 3D Juump Infinite client application stations.

3.2 - Install the LM-X borrow license file

Copy the license file (`*.lic`) provided by 3D Juump to the LM-X server folder.

Cluster administration

This chapter provides hints for the following administrative tasks:

- manage 3D Juump Infinite through its HTTP API and the *Administration portal* Web application
- enumerate all the started services and opened TCP/IP ports, to assist you in properly installing 3D Juump Infinite in conformance with the security policies of your enterprise
- download the activity report for billing purposes

1 - HTTP API

1.1 - Introduction

The 3D Juump Infinite ∞Directory and ∞Proxy serve a group of well-known URLs. A documentation of the exposed endpoints is available in the `manual/HTTP API` folder of the release package.

- `3D Juump Infinite [Directory|Proxy|CLI] API documentation [version number].html` files contain the OpenAPI documentation of the public endpoints
- `3D Juump Infinite HTTP API security summary table [version].html` file list all the exposed endpoints

- *resources/3D Juump Infinite [Directory|Proxy|CLI] API documentation [version number].yaml* files contain the OpenAPI documentation of the public endpoints in yaml format
- *resources/3D Juump Infinite [Directory|Proxy|CLI] API security summary table [version].json* files list all the exposed endpoints in json format

Three HTTP APIs are documented:

- The ∞Directory API, exposed by ∞Directory servers
- The ∞Proxy API, exposed by ∞Proxy servers
- The Document indexer API, exposed by the ∞CLI for generation

1.2 - Authentication

Authentication methods vary depending on HTTP API endpoints, see the *3D Juump Infinite <*> API documentation* documents for details.

Some endpoints do not require any authentication at all.

1.3 - mTLS: Mutual authentication

On top of HTTP authentication, it is possible to enable **Client certificate authentication**. When enabled, each client will have to present a valid SSL certificate to the server in order to be authorized.

1.3.1 - Backend configuration

To enable mTLS at the installation, you have to provide a root certificate (*provided_mTLS_root_ca*) that will be used by Apache to validate client certificates. This will set the Apache setting *SSLVerifyClient* to *require*. The ∞Directory and ∞Proxy will be configured to present their own certificate. Note that every server certificate must be derived from the mTLS root certificate.

1.3.2 - Client configuration

The ∞CLI can be configured using the *client_certificate* field in the HTTP client configuration.

The 3D Juump Infinite native client can be configured to use a certificate using the fields *Security/SSLClientCertificate*, *Security/SSLClientKey* and *Security/SSLClientKeyPassword* within the *Settings.ini* file.

Web browsers like Chrome will prompt a dialog to select which certificate should be used.

2 - Administation portal

First, make sure that the ∞Directory has been installed properly and is running along with its dependent back-end services. The ∞Directory publishes a Web application interface that lets the administrator(s) manage the 3D Juump Infinite network, data and user access rights. This Web application is accessible directly through the ∞Directory Web server, by browsing the corresponding URL:

```
https://directory_name:${apache_https_port}/directory/admin
```

Please display the *Administration portal* with a supported browser: Microsoft Edge, Google Chrome or Mozilla Firefox.

2.1 - Main features

This documentation will broadly list the general features of the *Administration portal*, to give a sense of its organization. The user interface is meant to be self-explatory.

Many icons, buttons or other items of the *Administration portal* show tooltips on hover, which give extensive details or explanations on what you see! Don't hesitate to take advantage of this!

For an exhaustive list of capabilities, please refer to the documentation of the HTTP API `directory/api/manage/...` endpoints.

2.1.1 - Users and Teams

- Pre-register a user by providing its OpenID Connect User ID
- Manage teams of users
- Enable/disable global access to 3D Juump Infinite
- Promote/demote administrator users
- Manage the access rights of users and teams

2.1.2 - Applications

- Install/update an application package
- Register a third-party application
- Manage application tags
- Manage application access rights

2.1.3 - Cluster management

- Monitor the status of the ∞Directory, ∞Proxies and ∞AsyncJobSolvers of the cluster
- Manage the tags of the ∞Proxies
- Revoke all the user sessions opened on an ∞Proxy
- Send cleanup requests to an ∞Proxy

2.1.4 - Cluster computing

- Monitor asynchronous jobs

2.1.5 - Projects

- Create/recreate projects
- Monitor the existing projects and the build statuses
- Manage builds
- Restore (install) and download EvoJuump files.
  When restoring or downloading large or multiple EvoJuumps, it is advised to use the ∞CLI to obtain better performances.
- Monitor the progress of build generation/restoration

2.1.6 - Data Management

- Monitor geometry pools, metadata DocStores and asset stores

- Request the release of unused geometry data

2.1.7 - Settings

- Manage the dynamic settings of the ∞Directory

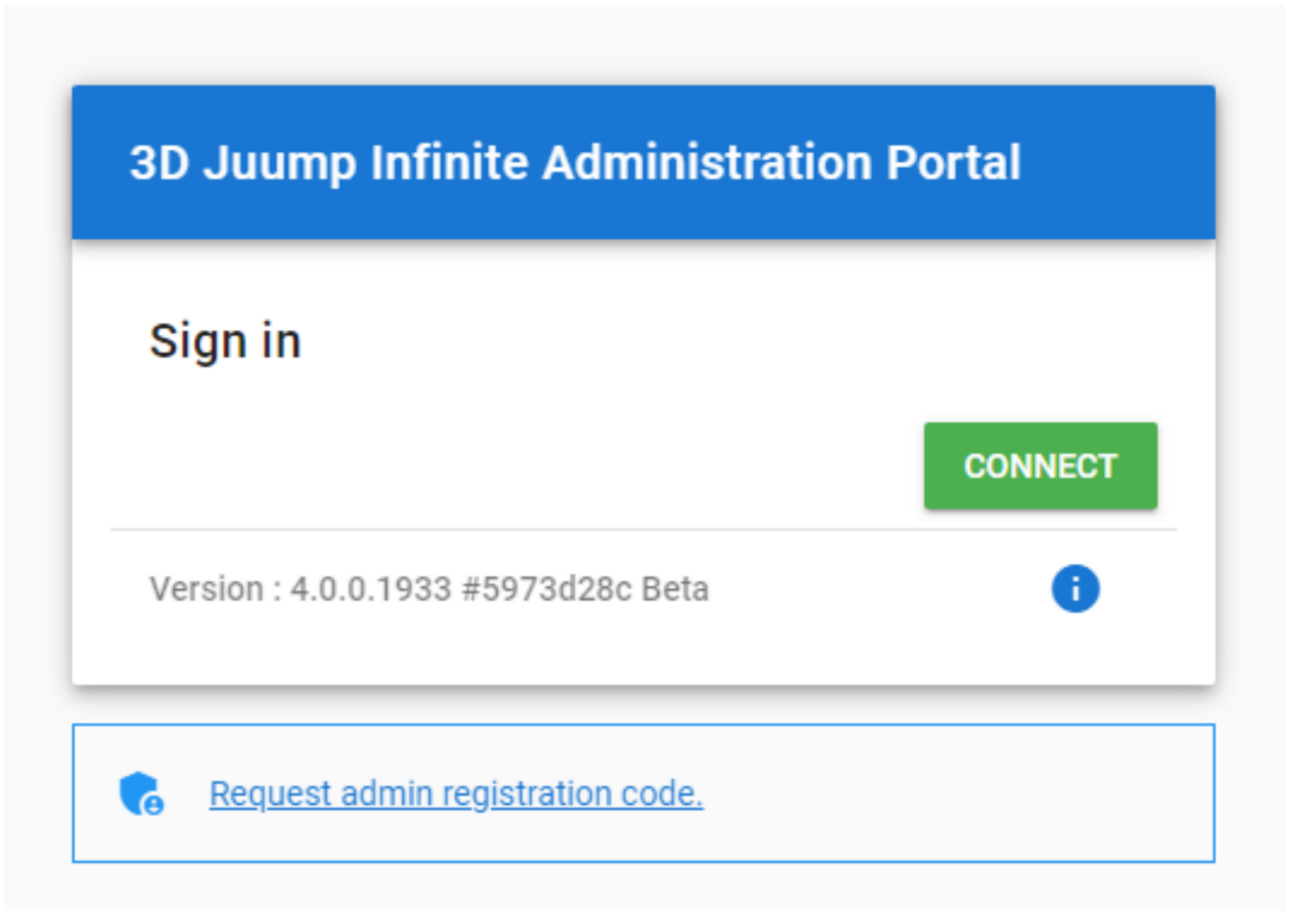
2.1.8 - Licenses

- Monitor the status of your license (capabilities, slot count, periods of validity)
- Download a license request file
- Install a new license
- Download the activity report

2.2 - Administrator registration

The access to the *Administration portal* is limited to Administrator users. There are various types of administrators. Those that have exhaustively all administration rights are labeled super-administrators. Other types of administrators can only perform a limited set of operations.

To become a super-administrator, a user can navigate to the *Administration portal* URL (as in the image below) and click on the link reading `Request admin registration code`.


3D Juump Infinite Administration Portal

Sign in

Version : 4.0.0.1933 #5973d28c Beta

Request admin registration code.

Login

The user will then obtain a code that must be used with the command of the ∞CLI directory registersuperadmin and the API-key to register themself as a super-administrator.

Endpoint responsible of generating registration code is disable by default. See ∞Directory configuration file enablegetsuperadminregistrationcode to enable it.

A super-administrator can then promote other users to super-administrators or other types of administrators.

3 - Monitoring

3.1 - Loki/Grafana log handler

The log messages generated by the 3D Juump Infinite services can be automatically pushed to a Loki instance, allowing to automate error monitoring. Please refer to the ∞Directory configuration file and the ∞Proxy configuration file

3.2 - HTTP API

At anytime, you may probe the status of any 3D Juump Infinite node (∞Directory, ∞Proxy) using, respectively, the `getversion` endpoint of the ∞Directory or ∞Proxy API with the URL parameter `status=true`.

In the case of an ∞Proxy, the returned `status` indicates the status of the node itself. In the case of the ∞Directory, the returned `status` will give an idea of the general status of the 3D Juump Infinite network registered to this directory:

- `green`: All the nodes are running correctly.
- `yellow`: The Infinite network is partly functional, but some nodes may be down or some services may be unusable.
- `red`: The 3D Juump Infinite network is totally unusable.

An external script or service could be configured to probe the status of the various known 3D Juump Infinite nodes and alert the administrator when a failure is detected.

See the corresponding ∞Directory or ∞Proxy API documentation for full details.

4 - Asynchronous job

An `Asynchronous job` or `Async job` is a compute and bandwith intensive task that cannot be executed in a web browser.

Each `Async Job` is associated to a directory or data session, and this session should remain alive during the whole solving process.

Depending on type and complexity `Async Job` will be scheduled on an ∞AsyncJobSolver to be solved.

An `Async Job` will be limited in time, cpu and memory and might fail if it is too complex (see Limits).

5 - Data access control

3D Juump Infinite **securely** broadcasts DMUs to users. Data protection is deeply rooted into the design of 3D Juump Infinite, through all the DMU flow.

In this paragraph, we summarize the 3D Juump and third-party services required by the 3D Juump Infinite back end.

5.1 - TCP port usage

3D Juump Infinite brings its own service but also installs third-party servers. Make sure your network administrator properly configures these servers in compliance with your network policy.

We summarize which installed third-party servers are visible from outside. TCP port values are the default ones and may vary depending on your configuration.

This table presents a list of the TCP ports published by the ∞Directory and the ∞Proxy.

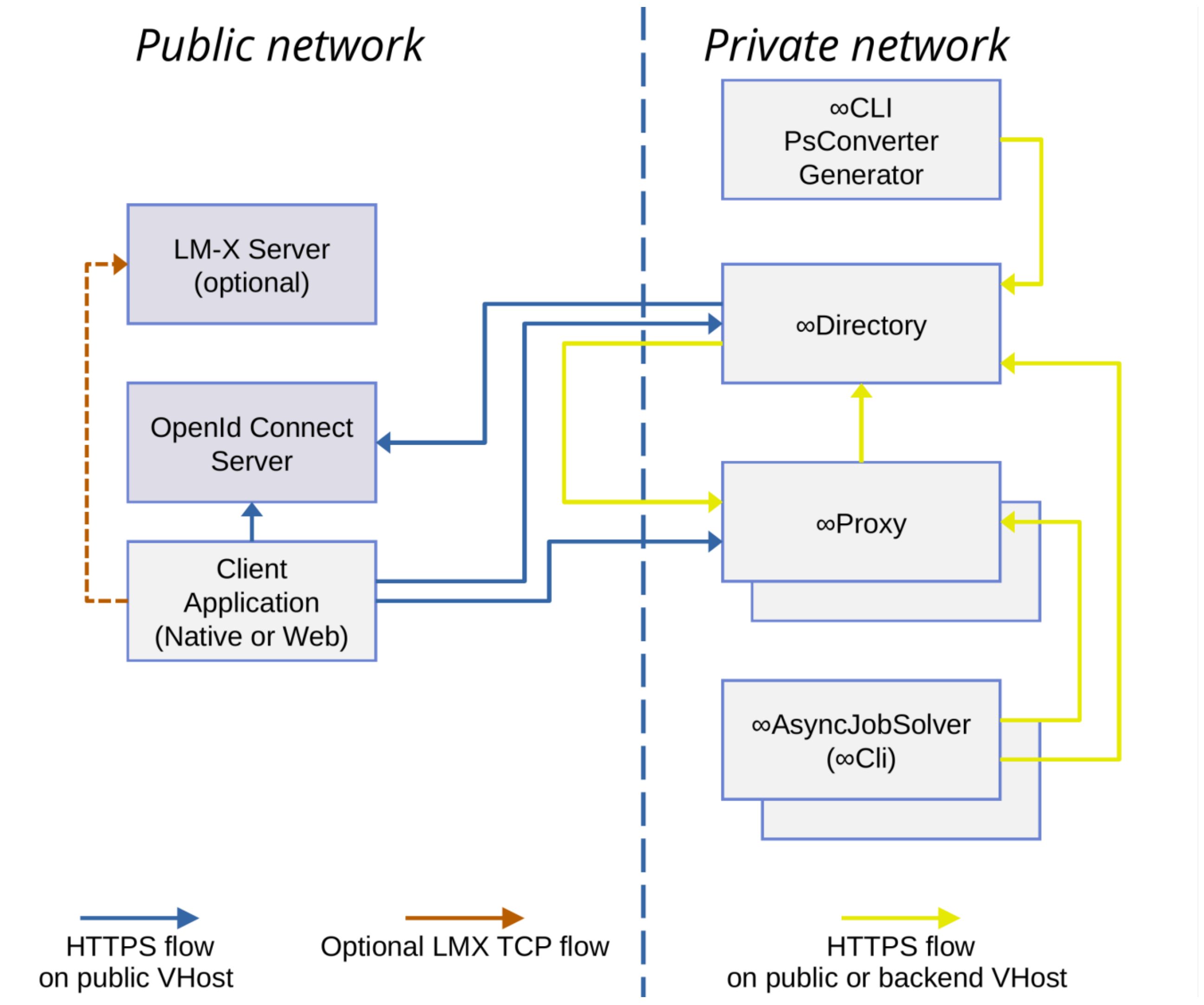
Service	∞Directory	∞Proxy	LM-X Server
Apache (HTTPS)	443	443	-
LM-X (optional)	-	-	6200

This table presents additional TCP ports used on loopback by the ∞Directory and the ∞Proxy.

Protocol	∞Directory	∞Proxy	LM-X Server
∞Service	2702	2703	-
Apache (HTTPS)	443	443	-
PostgreSQL	5440	5440	-
ElasticSearch	-	9200	-

The native ∞Client uses the HTTPS port 443 to communicate with the ∞Directory, and the ∞Proxy. In addition to that, the native ∞Client may use the LM-X Server port 6200 to borrow licenses. The ∞Directory and the ∞Proxy use the HTTPS port 443 to communicate with other network elements.

This diagram shows the dataflows between components.



∞Directory

5.2 - List of service startup users

5.2.1 - Windows

The Microsoft Windows startup user is the user able to start the current service. The name in parenthesis is for a French version of Microsoft Windows.

Name of service	Windows Service startup user
3djuump-infinite-directory	Network Service (Service Réseau)
3djuump-infinite-proxy	Network Service (Service Réseau)
Apachehttp3djuumpInfinite	Local System Account (Système Local)
postgresql-x64-16	Network Service (Service Réseau)
elasticsearch-service-x64	Local System Account (Système Local)

Apachehttp3djuumpInfinite startup user could be changed using `apache_win/user` and `apache_win/password`

5.2.2 - Linux

For Linux, the startup user and group are configured by the *general/user* field of the *form_install_info.yaml*.

5.3 - Data encryption

In order to ensure security, all transfers between the nodes of the 3D Juump Infinite network are encrypted using SSL/TLS protocols[11]. SSL is now deprecated, and Apache servers are configured by the install scripts to use only the protocols TLS v1.3 and above. For the sake of simplicity, we use the acronym *SSL* to designate both SSL and TLS protocols. To establish this encryption, a SSL certificate is mandatory.

5.4 - Data caching policy

The 3D Juump Infinite client applications can retrieve the *builds* from the ∞Proxy, with the proper authorizations of the ∞Directory. For a secure access to the data, the ∞Proxy cannot be accessed directly. It should be registered in a *DMU flow* and always remain in contact with the ∞Directory.

5.5 - Folder architecture

In case you followed the default folder architecture during the installation, files are organised as follows:

Folder	Usage
${install_basepath}	all the files related to the infinite cluster (except binaries) are located there.
${install_basepath}/service	all the files related to the ∞Directory and the ∞Proxy.
${install_basepath}/service/proxy	all the files related to data for the ∞Proxy.
${install_basepath}/service/filer	all the files related to the ∞Directory filer storage.
${install_basepath}/service/logs	logs of the ∞Directory and ∞Proxy services.
${install_basepath}/postgresql	PostgreSQL cluster data and logs.
${install_basepath}/elasticsearch	all the files related to the ElasticSearch service for the ∞Proxy.
${install_basepath}/elasticsearch/data	ElasticSearch database.
${install_basepath}/elasticsearch/logs	ElasticSearch logs.
${install_basepath}/ssl	private key and SSL certificate files.
${install_basepath}/ssl_pg	on Linux, certificate files used by Apache cannot have the same access rights. *ssl_pg* is a copy of the SSL folder with different access rights.
${install_basepath}/www	all the files related to the apache service for the ∞Directory and the ∞Proxy.
${install_basepath}/www/logs	Apache logs.

[11] Secure Sockets Layer and its successor Transport Layer Security (TLS) are cryptographic protocols that provide communication security over TCP/IP networks such as the Internet. wikipedia

${install_basepath}/www/empty	empty folder used in Apache configuration.
${install_basepath}/www/empty/public	public part of the empty folder.

5.6 - ∞Directory

5.6.1 - Administration data

Administration data is stored in a secure database that no unauthorized users can access:

- nor remotely:
  - the PostgreSQL server requires authentication, and is not exposed on the client side
- nor locally:
  - thanks to the operating system authentication procedure.

The ∞Directory service stores its own credentials (for accessing local servers) with a well-known cryptographic algorithm.

5.6.2 - DMU data

The data pushed by a Connector into an ∞Directory is stored in secured databases or filers that no unauthorized users can access:

- nor remotely:
  - the PostgreSQL server requires authentication, and is not exposed on the client side
  - the filer is not exposed on the client side
  - an authentication procedure is enforced by the HTTPS servers
- nor locally:
  - thanks to the operating system authentication procedure.

The 3D data pushed by the ∞CLI to the ∞Directory is stored in a proprietary binary format in a repository that can only be accessed by authorized users. It requires a valid system account with read-access. Remote access is permitted by the HTTPS API to authorized users only.

5.6.3 - Assets

Assets are stored in secure databases that no unauthorized users can access:

- nor remotely:
  - the PostgreSQL server requires authentication, and is not exposed on the client side
  - an authentication procedure is enforced by the HTTPS servers
- nor locally:
  - thanks to the operating system authentication procedure.

Remote access is permitted by the HTTPS API to authorized users only.

5.7 - ∞Proxy

5.7.1 - Administration data

The ∞Proxy service stores its own credentials (for accessing local servers) with a well-known cryptographic algorithm.

5.7.2 - DMU data

Replicated data is stored in secure databases as specified for the ∞Directory.

Build data is stored on disk in a proprietary binary format, in a repository that can only be accessed by authorized users. It requires a valid System account with read-access. Remote access is ensured by the HTTPS server for authorized users only.

5.8 - 3D Juump Infinite native client application

5.8.1 - Connected mode

At start-up, the 3D Juump Infinite native client application requires the user to log into the ∞Directory. No connection with any other node of the 3D Juump Infinite network is initiated unless the ∞Directory has authorized it. Credentials delivered by the ∞Directory for the session are temporary and not reusable.

The 3D Juump Infinite native client application itself possesses no credentials to access the ∞Directory or ∞Proxies.

Cached data is stored on disk in the user’s temporary folder.

5.8.2 - Standalone mode

Once a DMU file has been installed to the 3D Juump Infinite Local DMU Manager or downloaded from an ∞Directory, its content is made available to any user authenticated to the local computer. 3D data are stored on disk in binary containers (proprietary format, tessellated geometries only).

5.9 - Communications

All network communications between the ∞Directory, ∞Proxies and 3D Juump Infinite native client application is secured using the HTTPS protocol.

5.10 - DMU files

DMU files (*EvoJuump*) are encrypted and password-protected.

6 - Fault tolerance and preventive maintenance

3D Juump Infinite is designed to ensure the availability of the DMU to the connected users. However, like every system, it is subject to fault and breakdown. This chapter gathers a non-exhaustive list of procedures that administrators may choose to apply to keep a high availability of services.

6.1 - Temporary network failure

3D Juump Infinite client application is tolerant to temporary network failure: it does not rely on keep-alive connections.

Temporarily losing the network connectivity will only result in temporarily inoperative functions (search, id-card…) and 3D views will be limited to the high-quality 3D data that is already cached locally. After a short period (minutes), the user is forced to quit, see the Administration page settings for more information.

6.2 - ∞Directory failure

To circumvent a lasting ∞Directory failure, the ∞Directory should be deployed on a distributed architecture.

The ∞Directory heavily relies on the PostgreSQL database and the filer storage (filesystem, network filer, Azure blob, Amazon S3 bucket), those third-party services should be backed up to allow a fast recovery.

6.2.1 - Asset databases

The assets database is hosted on a regular PostgreSQL server on the ∞Directory, and thus supports classical PostgreSQL preventive maintenance procedures. Particularly, it is good practice to maintain a secondary PostgreSQL server and initiate a continuous master-slave replication or scheduled punctual replications for future restoration.

A periodical backup could be implemented using the ∞Directory asset HTTP API.

6.3 - ∞Proxy failure

6.3.1 - Redundancy by ∞Proxy

To circumvent a lasting ∞Proxy failure, one can rely on redundant ∞Proxies.

The ∞Directory will route new connection requests to the redundant ∞Proxy and users will thus still be able to access and browse the DMU.

6.3.2 - Build backups

To backup a particular *build*, use the EvoJuump packing procedure. This *EvoJuump* can later be restored to an ∞Directory.

6.3.3 - Project backups

To backup a project, just save its `projectid`. You can the use it to recreate the project on the ∞Directory.

6.4 - LM-X failure

3D Juump Infinite servers do not directly interact with the LM-X software protection system. An LM-X failure will only impact the borrow feature of the Local DMU Manager. If you need the borrow feature, please carefully read the LM-X documentation for information regarding maintenance and best practice.

7 - Migration procedures

7.1 - Cluster migration

To migrate a cluster:

1. Retrieve relevant data from the ∞Directory
   - Projects
   - Users
   - Teams
   - …
2. Pack and download any useful build (as EvoJuumps) from the projects (from the ∞Directory).
   - If you are migrating many builds, it is advised to use the command ∞CLI evojuump dump to pack the EvoJuumps instead of the *Administration portal*.
   - Check the integrity of the EvoJuump packs using the command ∞CLI evojuump checkintegrity.
3. Uninstall previous version (binaries and data)
4. Run a fresh install of the new version
5. Restore relevant data to the ∞Directory
6. Migrate the builds using the command ∞CLI evojuump migrate (see the corresponding chapter in the Integration Manual).
7. Restore (upload) the migrated builds on the new ∞Directory. If you are restoring many builds, it is advised to use the command ∞CLI evojuump restore to restore the EvoJuumps instead of the *Administration portal*.

7.2 - Assets migration

⚠ Migrating from 3.0 and below requires that all assets are patched to reflect the new authentication mechanisms. The assets' security descriptors have to be updated to reflect the new team and user identifiers.

1. Before starting the procedure, make sure all asset databases are properly backed up
2. Retrieve the internal team identifiers from the old cluster
3. Migrate the cluster (see above)
4. Recreate the teams
5. Prepare a map of all users, matching the previous user identifier to the corresponding subid from the new *Authenticator*
6. Prepare a map of all teams, matching the previous team identifier to the corresponding team identifier from the new ∞Directory
7. Patch all the assets, replacing previous user and team identifiers in the security descriptor.

⚠ In versions of 3D Juump Infinite 3.2 and below, assets were stored in a CouchDB database. Starting with 3D Juump Infinite 3.3, assets are stored in the PostgreSQL database of the ∞Directory. To migrate assets from a 3.2 version to a 3.3 of higher version, we advise to follow these steps:

1. Before starting the procedure, make sure all asset databases are properly backed up.
2. Download all the assets (CouchDB documents) from the CouchDB database while your previous ∞Directory 3.2 is running.

3. Within the security descriptor of each asset, replace 3.2 team and user identifiers by their new counterparts. Make sure that no old identifier remains, or it will result in inaccessible assets.
4. Upload the patched assets to the newly installed ∞Directory, using the ∞Directory asset API (in particular, the *`directory/api/assets/push`* endpoint). See the corresponding ∞Directory API documentation for full details.

Executable command line manual

1 - ∞CLI: Command Line Interface

1.1 - Introduction to the Infinite CLI

The **3D Juump Infinite CLI** is a binary executable that performs various common operations related to 3D Juump Infinite servers components, like configuring backends, managing Evojuumps, converting CAD data, generating builds… One could compare it to a remote control room or a toolbox, provided to system administrators and data integrators of 3D Juump Infinite. Depending on the tasks that are undertaken, it can be used either from the user's personal machine or from the server machine, using a command line shell. It can serve for:

- 3D Juump Infinite Directory or Proxy configuration and initialization *(on the server machine)*
  - directory conf
  - directory init
  - proxy conf
  - proxy init
- 3D Juump Infinite Directory administration *(from the administrator's machine)*
  - directory registersuperadmin
  - directory registeroidc

- Data integration: generating and managing builds *(from the administrator's machine)*
  - generator ...
  - psconverter ...
  - evojuump ...
- Running side services *(on the machine destined to run the service)*
  - generator docindexer
  - asyncjob ...
- Managing and configuring the Infinite CLI itself
  - cli conf
  - cli ...
  - directory list
  - directory registerapikey

The **3D Juump Infinite CLI** is shipped alongside the **∞Directory** and **∞Proxy** binaries, so that you can perform initialization operations with the server packages.

As usual with command-line tools, a detailed "usage" is available in the executable itself. We invite you to consult it for exact and up-to-date information about commands: - Executing *`./3dJuumpInfiniteCli.exe help -d`* will list all the available commands with their arguments and a short description. - You may notice that commands are grouped by topic. Executing *`./3dJuumpInfiniteCli.exe pscoverter`* will list all the subcommands grouped under the *`psconverter`* topic.

Many commands require that you configure the **∞CLI** in order to contact the **∞Directory** you want to interact with. The Infinite CLI stores the specified credentials only locally, with encryption.

For instance, a first step would be to execute...

```
./3dJuumpInfiniteCli.exe directory registerapikey MyServerNickname 'https://p
reprod.3djuump.com:443/directory' 'MySecurePassword'
```

...before running various tasks on that server, for instance restoring on an Evojuump:

```
./3dJuumpInfiniteCli.exe evojuump restore 'C:\MyEvojuump.evojuump' 'my_evojuu
mp_password' MyServerNickname --worker-count 4
```

You may also refer to the detailed documentation below for a description of commands.

1.2 - ∞Directory

Commands starting with ***`directory ...`***

Regroups the various commands operating the ∞Directory.

1.2.1 - directory conf ...

Manages the ∞Directory configuration file.

1.2.1.1 - *directory conf init*

Initializes the ∞Directory configuration file with default fields.

```
directory conf init [--conf-file|-c <val>]
```

Optional arguments:

- *`--conf-file|-c <val>`* : Path to the ∞Directory configuration file. If left empty, it will default to '*`%PROGRAMDATA%/3djuump-infinite-directory/conf_4_1.json`*' on Windows or '*`/etc/3djuump-infinite-directory/conf_4_1.json`*' on Linux.
  default: ""
  maxLength: 1024
  minLength: 0

1.2.1.2 - *directory conf consolidate*

Validates the content of the ∞Directory configuration file and ciphers all the sensitive fields. See the *`cipher`* CLI option to pre-cipher values.

```
directory conf consolidate [--conf-file|-c <val>]
```

Optional arguments:

- *`--conf-file|-c <val>`* : Path to the ∞Directory configuration file. If left empty, it will default to '*`%PROGRAMDATA%/3djuump-infinite-directory/conf_4_1.json`*' on Windows or '*`/etc/3djuump-infinite-directory/conf_4_1.json`*' on Linux.
  default: ""
  maxLength: 1024
  minLength: 0

1.2.2 - directory init

Initializes/Updates ∞Directory resources (PostgreSQL database, folders, …). If the PostgreSQL database does not exists, a connection to the 'postgres' database will be made to create it. This command should be executed after each binary update.

```
directory init [--conf-file|-c <val>] [--pg-wait-timeout <val>]
```

Optional arguments:

- *`--conf-file|-c <val>`* : Path to the ∞Directory configuration file. If left empty, it will default to '*`%PROGRAMDATA%/3djuump-infinite-directory/conf_4_1.json`*' on Windows or '*`/etc/3djuump-infinite-directory/conf_4_1.json`*' on Linux.
  default: ""
  maxLength: 1024
  minLength: 0
- *`--pg-wait-timeout <val>`* : The maximum wait duration when testing the availability of the PostgreSQL server.
  default: 10
  maximum: 360
  minimum: 10

1.2.3 - directory registersuperadmin

Registers a user on the ∞Directory as a super administrator.

```
directory registersuperadmin <directory_nickname> <registration_code>
```

Cancelable using CTRL+C/SIGTERM events.

Arguments:

- *`<directory_nickname>`* : A nickname used to reference an ∞Directory/credentials pair previously registered on the CLI.
  example: "my_directory"
  maxLength: 64
  minLength: 1
  pattern: *`\S+`*
- *`<registration_code>`* : A registration code retrieved from the endpoint *`/directory/api/directorysession/requestsuperadminregistrationcode`*.
  example: "0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef"
  minLength: 1

1.2.4 - directory registerapikey

Registers API key credentials in the configuration file to access an ∞Directory.

```
directory registerapikey <directory_nickname> <directory_url> <api_key> [--lo
cation|-l <.|userscope|systemscope|auto>] [--nosslcheck] [--mtls-cert-p12 <va
l>] [--mtls-cert-pwd <val>]
```

Arguments:

- *`<directory_nickname>`* : A nickname used to reference an ∞Directory/credentials pair previously registered on the CLI.
  example: "my_directory"
  maxLength: 64
  minLength: 1
  pattern: *`\S+`*
- *`<directory_url>`* : Directory API URL for public usage. The port has to be explicited as this attribute is part of the license.
  example: "https://127.0.0.1:443/directory"
  maxLength: 1024
  pattern: *`^https?:\/\/[^@\/A-Z]+?(:[1-9][0-9]{0,4})(\/.*)?\/directory$`*
- *`<api_key>`* :
  minLength: 1

Optional arguments:

- *`--location|-l <.|userscope|systemscope|auto>`* : Predefined location of the configuration file. '*`.`*' will be in the current working directory. '*`userscope`*' will be in *`%APPDATA%`* on Windows and *`~/`* on Linux. '*`systemscope`*' will be in *`%PROGRAMDATA%`* on Windows and *`/etc/`* on Linux. '*`auto`* will use search path to find configuration file and will default to userscope if no existing file was found.

default: “auto”
enum: [‘.’, ‘userscope’, ‘systemscope’, ‘auto’]

- `--nosslcheck` : Disable SSL peer verification for directory calls.
  default: False
- `--mtls-cert-p12 <val>` : Path to a mTLS p12 client certificate.
  default: ""
- `--mtls-cert-pwd <val>` : The mTLS p12 password.
  default: ""

1.2.5 - directory registeroidc

Registers OpenId Connect credentials in the configuration file to access an ∞Directory.

```
directory registeroidc <directory_nickname> <directory_url> <oidc_conf_url> <
client_id> <client_secret> [--location|-l <.|userscope|systemscope|auto>] [--
nosslcheck] [--mtls-cert-p12 <val>] [--mtls-cert-pwd <val>] [--noproxyforoidc
]
```

Arguments:

- `<directory_nickname>` : A nickname used to reference an ∞Directory/credentials pair previously registered on the CLI.
  example: “my_directory”
  maxLength: 64
  minLength: 1
  pattern: `\S+`
- `<directory_url>` : Directory API URL for public usage. The port has to be explicited as this attribute is part of the license.
  example: “https://127.0.0.1:443/directory”
  maxLength: 1024
  pattern: `^https?:\/\/[^@\/A-Z]+?(:[1-9][0-9]{0,4})(\/.*)?\/directory$`
- `<oidc_conf_url>` : OpenID Provider configuration URL (https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderConfigurationRequest).
  example: “https://myapp.auth0.com/.well-known/openid-configuration”
  maxLength: 1024
  pattern: `^https:\/\/([^\/]*?)\/.*$`
- `<client_id>` :
  minLength: 1
- `<client_secret>` :
  minLength: 1

Optional arguments:

- `--location|-l <.|userscope|systemscope|auto>` : Predefined location of the configuration file. ‘`.`’ will be in the current working directory. ‘`userscope`’ will be in `%APPDATA%` on Windows and `~/` on Linux. ‘`systemscope`’ will be in `%PROGRAMDATA%` on Windows and `/etc/` on Linux. ’`auto` will use search path to find configuration file and will default to userscope if no existing file was found.

default: “auto”
enum: [‘.’, ‘userscope’, ‘systemscope’, ‘auto’]

- `--nosslcheck` : Disable SSL peer verification for directory calls.
  default: False
- `--mtls-cert-p12 <val>` : Path to a mTLS p12 client certificate.
  default: ""
- `--mtls-cert-pwd <val>` : The mTLS p12 password.
  default: ""
- `--noproxyforoidc` : Disables the use of an HTTP proxy for directory calls.
  default: False

1.2.6 - directory unregister

Unregisters from the configuration file the credentials associated to an ∞Directory.

```
directory unregister <directory_nickname> [--location|-l <.|userscope|systems
cope|auto>]
```

Arguments:

- `<directory_nickname>` : A nickname used to reference an ∞Directory/credentials pair previously registered on the CLI.
  example: “my_directory”
  maxLength: 64
  minLength: 1
  pattern: `\S+`

Optional arguments:

- `--location|-l <.|userscope|systemscope|auto>` : Predefined location of the configuration file. ‘`.`’ will be in the current working directory. ‘`userscope`’ will be in `%APPDATA%` on Windows and `~/` on Linux. ‘`systemscope`’ will be in `%PROGRAMDATA%` on Windows and `/etc/` on Linux. ’`auto` will use search path to find configuration file and will default to userscope if no existing file was found.
  default: “auto”
  enum: [‘.’, ‘userscope’, ‘systemscope’, ‘auto’]

1.2.7 - directory list

Lists ∞Directories credentials registered in the configuration file.

```
directory list [--location|-l <.|userscope|systemscope|auto>] [--ext-info|--v
erbose|-v]
```

Optional arguments:

- `--location|-l <.|userscope|systemscope|auto>` : Predefined location of the configuration file. ‘`.`’ will be in the current working directory. ‘`userscope`’ will be in `%APPDATA%` on Windows and `~/` on Linux. ‘`systemscope`’ will be in `%PROGRAMDATA%` on Windows and `/etc/` on Linux. ’`auto` will use search path to find configuration file and will default to userscope if no existing file was found.

default: “auto”
enum: [‘.’, ‘userscope’, ‘systemscope’, ‘auto’]

- `--ext-info|--verbose|-v` : Print extended informations.
  default: False

1.3 - ∞Proxy

Commands starting with `proxy ...`

Regroups the various commands operating the ∞Proxy.

1.3.1 - proxy conf ...

Manages the ∞Proxy configuration file.

1.3.1.1 - *proxy conf init*

Initializes the ∞Proxy configuration file with default fields.

```
proxy conf init [--conf-file|-c <val>]
```

Optional arguments:

- `--conf-file|-c <val>` : Path to the ∞Proxy configuration file. If left empty, it will default to ‘`%PROGRAMDATA%/3djuump-infinite-proxy/conf_4_1.json`’ on Windows or ‘`/etc/3djuump-infinite-proxy/conf_4_1.json`’ on Linux.
  default: ""
  maxLength: 1024
  minLength: 0

1.3.1.2 - *proxy conf consolidate*

Validates the content of the ∞Proxy configuration file and ciphers all the sensitive fields. See the `cipher` CLI option to pre-cipher values.

```
proxy conf consolidate [--conf-file|-c <val>]
```

Optional arguments:

- `--conf-file|-c <val>` : Path to the ∞Proxy configuration file. If left empty, it will default to ‘`%PROGRAMDATA%/3djuump-infinite-proxy/conf_4_1.json`’ on Windows or ‘`/etc/3djuump-infinite-proxy/conf_4_1.json`’ on Linux.
  default: ""
  maxLength: 1024
  minLength: 0

1.3.2 - proxy init

Initializes/Updates ∞Proxy resources (PostgreSQL database, folders, ...). If the PostgreSQL database does not exists, a connection to the ‘postgres’ database will be made to create it. This command should be executed after each binary update.

```
proxy init [--conf-file|-c <val>] [--pg-wait-timeout <val>]
```

Optional arguments:

- *--conf-file|-c <val>* : Path to the ∞Proxy configuration file. If left empty, it will default to '*%PROGRAMDATA%/3djuump-infinite-proxy/conf_4_1.json*' on Windows or '*/etc/3djuump-infinite-proxy/conf_4_1.json*' on Linux.
  default: ""
  maxLength: 1024
  minLength: 0
- *--pg-wait-timeout <val>* : The maximum wait duration when testing the availability of the PostgreSQL server.
  default: 10
  maximum: 360
  minimum: 10

1.4 - ∞AsyncJobSolver

1.4.1 - asyncjob solver

Start an instance of asynchronous job solver. Note that if used directory API key could only be used as credentials if all proxies also have an API key configured.

```
asyncjob solver <directory_nickname> <tmp_folder> [--maxmemorymb <val>] [--ma
xcpu <val>] [--jobtypes <val>] [--openglprovider <none|system|mesa>]
```

Cancelable using CTRL+C/SIGTERM events.

Arguments:

- *<directory_nickname>* : A nickname used to reference an ∞Directory/credentials pair previously registered on the CLI.
  example: "my_directory"
  maxLength: 64
  minLength: 1
  pattern: *\S+*
- *<tmp_folder>* : Path to a temporary folder in which worker will write temporary files.

Optional arguments:

- *--maxmemorymb <val>* : Maximum amount of memory that could be used to solve jobs. A minimum of 2048MB is recommended. If 0 80% of RAM will be used.
  default: 0
  minimum: 0
- *--maxcpu <val>* : Maximum number of cpu that could be used to solve jobs. A minimum of 2 is recommended. If 0 all cpu will be used.
  default: 0
  minimum: 0
- *--jobtypes <val>* : Comma separated list of accepted job types, if empty all supported job types will be accepted.
  default: "“
  example:”test,export3d"

- *`--openglprovider <none|system|mesa>`* : Allows to choose which OpenGL implementation should be used. *`none`* : will not load OpenGL support, making some export jobs unprocessable. *`system`* : will use OpenGL implementation provided by the OS. *`mesa`* : will use Mesa3D LLVM software renderer (only available on Windows).
  default: “system”
  enum: [‘none’, ‘system’, ‘mesa’]

1.5 - PSConverter

Commands starting with **`psconverter ...`**

Performs the conversion of CAD/3D file.

1.5.1 - psconverter convert

Processes the given PSConverter job file.

```
psconverter convert <job_file> <directory_nickname>
```

Cancelable using CTRL+C/SIGTERM events.

Arguments:

- *`<job_file>`* : An x-ndjson file containing a settings JSON, followed by a list of jobs that should be processed by the PSConverter. See PSConverter section in Integration manual for details.
  example: C:/psconverter_input_file.x-ndjson
  minLength: 1
- *`<directory_nickname>`* : A nickname used to reference an ∞Directory/credentials pair previously registered on the CLI.
  example: “my_directory”
  maxLength: 64
  minLength: 1
  pattern: `\S+`

1.5.2 - psconverter version

Returns the version of the PSConverter.

```
psconverter version
```

1.5.3 - psconverter supportedfileformats

Returns list of supported file extensions.

```
psconverter supportedfileformats
```

1.6 - Generator

Commands starting with **`generator ...`**

Regroups various operations relative to the build generation process.

1.6.1 - generator canbuild

Checks that the specified project exists and that the licence allows this tool to generate builds.

```
generator canbuild <project_id> <directory_nickname> [--ifmissingcreatewithna
me <val>]
```

Cancelable using CTRL+C/SIGTERM events.

Arguments:

- *`<project_id>`* : The unique identifier of the project.
  example: “prj_091d232caeae2c0d2e4e63b031534c5b”
  maxLength: 36
  minLength: 36
  pattern: *`^prj_[a-f0-9]{32}$`*
- *`<directory_nickname>`* : A nickname used to reference an ∞Directory/credentials pair previously registered on the CLI.
  example: “my_directory”
  maxLength: 64
  minLength: 1
  pattern: *`\S+`*

Optional arguments:

- *`--ifmissingcreatewithname <val>`* : If project does not exists, it will be created with provided name.
  default: ""
  maxLength: 128
  minLength: 1

1.6.2 - generator build

Launches the generation of a build.

```
generator build <build_description_file> <index_url> <directory_nickname>
```

Cancelable using CTRL+C/SIGTERM events.

Arguments:

- *`<build_description_file>`* : A file containing the various build parameters. See Generator section in Integration manual for details.
  example: C:/generator_input_file.json
  minLength: 1
- *`<index_url>`* : Full URL to the JSON document provider index (elasticsearch, CLI doc indexer, ...).
  example: “http://127.0.0.1:9200/prj_195df47ecc0c2f7d2d6ab832b28a3e84_connector”
  minLength: 1
  pattern: *`^https?:\/\/.*$`*
- *`<directory_nickname>`* : A nickname used to reference an ∞Directory/credentials pair previously registered on the CLI.

example: “my_directory”
maxLength: 64
minLength: 1
pattern: `\S+`

1.6.3 - generator syncmetadata

Imports new metadata from a connector index into a project DocStore.

```
generator syncmetadata <index_url> <directory_nickname> <projectid> [--worker
-count|-w <val>]
```

Cancelable using CTRL+C/SIGTERM events.

Arguments:

- *`<index_url>`* : Full URL to the JSON document provider index (elasticsearch, CLI doc indexer, ...).
  example: “http://127.0.0.1:9200/prj_195df47ecc0c2f7d2d6ab832b28a3e84_connector”
  minLength: 1
  pattern: `^https?:\/\/.*$`
- *`<directory_nickname>`* : A nickname used to reference an ∞Directory/credentials pair previously registered on the CLI.
  example: “my_directory”
  maxLength: 64
  minLength: 1
  pattern: `\S+`
- *`<projectid>`* : The unique identifier of the project.
  example: “prj_091d232caeae2c0d2e4e63b031534c5b”
  maxLength: 36
  minLength: 36
  pattern: `^prj_[a-f0-9]{32}$`

Optional arguments:

- *`--worker-count|-w <val>`* : The maximum number of concurrent workers.
  default: 4
  maximum: 16
  minimum: 1

1.6.4 - generator docindexer

Runs a local in-memory document indexer that can act as a substitute to ElasticSearch in its role of input for build generation.

```
generator docindexer <listeningport> [--close-with-parent] [--log-file <val>]
[--validate-documents] [--compress] [--http-compress] [--swap-folder <val>] [
--max-memory-before-swap-mb <val>] [--dump-out <val>] [--dump-in <val>] [--du
mp-format <dataonly|_index|folder>]
```

Cancelable using CTRL+C/SIGTERM events.

Arguments:

- *`<listeningport>`* :
  maximum: 65535
  minimum: 1025

Optional arguments:

- *`--close-with-parent`* : If set, the process will exit if its parent process is closed.
  default: False
- *`--log-file <val>`* : Overrides the location of the log file specified in the CLI configuration.
  default:
- *`--validate-documents`* : Enable JSON document validation when inserting them into the index. This option SHOULD NOT BE used in production as it will reduce performances.
  default: False
- *`--compress`* : Compresses documents, it will reduce memory usage but will be slower to index.
  default: False
- *`--http-compress`* : Allow compression of HTTP response. This will reduce bandwith but will cost CPU and add latency.
  default: False
- *`--swap-folder <val>`* : If set JSON data will be dumped into this folder to limit memory usage. Only index will reside in memory. Using swap file will obviously reduce indexation and fetch speed.
  default:
- *`--max-memory-before-swap-mb <val>`* : If –swap-folder is specified, will be the maximum amound of document data kept in memory before starting storing data on swap file. Note that total used memory will be greater than this limit as index will be kept in memory.
  default: 32
  minimum: 0
- *`--dump-out <val>`* : File location where index content will be dumped on exit or first search.
- *`--dump-in <val>`* : File location from which to initialize doc indexer.
- *`--dump-format <dataonly|_index|folder>`* : Dump file format.
  `_index` : same format as /_index API endpoint.
  `dataonly` : an x-ndjson containing only documents. It is more suitable for interoperability but will be slower to import.
  `folder` : a folder containing a JSON file per document. Folder will be scanned recursively. Only supported as input format.
  default: "_index"
  enum: [‘dataonly’, ’_index’, ‘folder’]

1.6.4.1 - generator geometrypool dump

Dump generation data from the geometry pool. This dump is intended for debug purposes, to be transfered to 3D Juump Infinite support team.

```
generator geometrypool dump <output_file> <geometry_pool_id> <directory_nickn
ame>
```

Cancelable using CTRL+C/SIGTERM events.

Arguments:

- `<output_file>` : A file that will contain dumped data.
  example: C:/geometry_pool_content.dump
  minLength: 1
- `<geometry_pool_id>` : The unique identifier of a geometry pool.
  example: “geom_091d232caeae2c0d2e4e63b031534c5b”
  maxLength: 37
  minLength: 37
  pattern: `^geom_[a-f0-9]{32}$`
- `<directory_nickname>` : A nickname used to reference an ∞Directory/credentials pair previously registered on the CLI.
  example: “my_directory”
  maxLength: 64
  minLength: 1
  pattern: `\S+`

1.6.5 - generator elasticsearch dump

Dump an elasticsearch index content, to a format compatible with docindexer dump. This dump is intended for debug purposes, to be transfered to 3D Juump Infinite support team.

```
generator elasticsearch dump <output_file> <index_url>
```

Cancelable using CTRL+C/SIGTERM events.

Arguments:

- `<output_file>` : A file that will contain dumped data.
  example: C:/geometry_pool_content.dump
  minLength: 1
- `<index_url>` : Full URL to the JSON document provider index (elasticsearch, CLI doc indexer, ...).
  example: “http://127.0.0.1:9200/prj_195df47ecc0c2f7d2d6ab832b28a3e84_connector”
  minLength: 1
  pattern: `^https?:\/\/.*$`

1.7 - Evojuump

Commands starting with **`evojuump ...`**

Regroups various operations relative to EvoJuump files.

1.7.1 - evojuump info

Displays the properties of the given EvoJuump. Integrity will be checked if correct password is provided.

```
evojuump info <evojuump> [--password <val>]
```

Cancelable using CTRL+C/SIGTERM events.

Arguments:

- `<evojuump>` : An EvoJuump file.
  example: C:/mybuild.evojuump
  minLength: 1

Optional arguments:

- `--password <val>` : The password to test, could be omitted.
  default: ""
  minLength: 0

1.7.2 - evojuump restore

Restores the specified EvoJuump to an ∞Directory.

```
evojuump restore <srcevojuump> <passphrase> <directory_nickname> [--worker-co
unt|-w <val>]
```

Cancelable using CTRL+C/SIGTERM events.

Arguments:

- `<srcevojuump>` : An EvoJuump file.
  example: C:/mybuild.evojuump
  minLength: 1
- `<passphrase>` : The passphrase of the EvoJuump.
  example: “mypassphrase1”
  minLength: 0
- `<directory_nickname>` : A nickname used to reference an ∞Directory/credentials pair previously registered on the CLI.
  example: “my_directory”
  maxLength: 64
  minLength: 1
  pattern: `\S+`

Optional arguments:

- `--worker-count|-w <val>` : The maximum number of concurrent workers.
  default: 2
  maximum: 8
  minimum: 1

1.7.3 - evojuump migrate

Migrates an old EvoJuump to the current version of the CLI. The migration log will be saved next to the resulting EvoJuump.

```
evojuump migrate <srcevojuump> <passphrase> <dstevojuump> <postgresurl> <tmpf
older> [--script <val>] [--diag-script] [--worker-count|-w <val>]
```

Cancelable using CTRL+C/SIGTERM events.

Arguments:

- `<srcevojuump>` : An EvoJuump file.
  example: C:/mybuild.evojuump
  minLength: 1
- `<passphrase>` : The passphrase of the EvoJuump.
  example: “mypassphrase1”
  minLength: 0
- `<dstevojuump>` : The destination for the migrated EvoJuump file.
  example: C:/mybuild_migrated.evojuump
  minLength: 1
- `<postgresurl>` : The URL to the PostgreSQL cluster of an ∞Proxy, that will be used for the migration.
  example: “pg://login:pwd@127.0.0.1:5400”
  pattern: `^(pg|postgres|postgresql):\/\/(.+@)?([^@]+?):[1-9][0-9]{0,4}$`
- `<tmpfolder>` : A temporary folder used to unpack the EvoJuump.
  minLength: 1

Optional arguments:

- `--script <val>` : A migration script to update metadata documents.
  default:
  minLength: 1
- `--diag-script` : If set each script call will be logged (as DEBUG) with input and output. This operation will be slow and will generate a huge amount of log.
  default: False
- `--worker-count|-w <val>` : The maximum number of concurrent workers.
  default: 2
  maximum: 32
  minimum: 1

1.7.4 - evojuump checkintegrity

Checks the integrity of a EvoJuump without need of password. Only EvoJuump generated by version 4.0 or upper are supported. For older EvoJuump use ‘EvoJuump info’.

```
evojuump checkintegrity <srcevojuump>
```

Arguments:

- `<srcevojuump>` : An EvoJuump file.
  example: C:/mybuild.evojuump
  minLength: 1

1.7.5 - evojuump dump

Generates an EvoJuump (pack file) of a build. It is recommended to perform this operation when no build generation or metadata update is in progress on the associated project.

```
evojuump dump <dstevojuump> <passphrase> <directory_nickname> <buildid> [--wo
rker-count|-w <val>] [--checksum-file <val>] [--no-generation-stats]
```

Cancelable using CTRL+C/SIGTERM events.

Arguments:

- `<dstevojuump>` : The destination for the EvoJuump file.
  example: C:/mybuild.evojuump
  minLength: 1
- `<passphrase>` : The passphrase of the EvoJuump.
  example: “mypassphrase1”
  minLength: 0
- `<directory_nickname>` : A nickname used to reference an ∞Directory/credentials pair previously registered on the CLI.
  example: “my_directory”
  maxLength: 64
  minLength: 1
  pattern: `\S+`
- `<buildid>` : The unique identifier of a build.
  example: “{48577a06-4733-4510-a49a-ce2202134617}”
  maxLength: 38
  minLength: 38
  pattern: `^\{[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}\}$`

Optional arguments:

- `--worker-count|-w <val>` : The maximum number of concurrent workers.
  default: 2
  maximum: 8
  minimum: 1
- `--checksum-file <val>` : The file where to store checksums of the EvoJuump file.
  default:
  minLength: 1
- `--no-generation-stats` : When set to true, does not pack the files `generation_statistics.json` and `generation_statistics.html` within the EvoJuump.
  default: False

1.7.6 - evojuump unpack

Unpacks the given EvoJuump to allow data processing by a connector.

```
evojuump unpack <srcevojuump> <passphrase> <directory_nickname>
```

Cancelable using CTRL+C/SIGTERM events.

Arguments:

- `<srcevojuump>` : An EvoJuump file.
  example: C:/mybuild.evojuump
  minLength: 1

- `<passphrase>` : The passphrase of the EvoJuump.
  example: “mypassphrase1”
  minLength: 0
- `<directory_nickname>` : A nickname used to reference an ∞Directory/credentials pair previously registered on the CLI.
  example: “my_directory”
  maxLength: 64
  minLength: 1
  pattern: `\S+`

1.8 - ∞CLI misc

1.8.1 - version

Outputs the version of the tool on the standard output (stdout).

```
version [--output-file|-o <val>]
```

Optional arguments:

- `--output-file|-o <val>` : Instead of stdout, specifies a destination file within which the version string should be written.
  default:

1.8.2 - help

List all CLI commands. Command detail documentation is available by calling desired command without arguments.

```
help [--description|-d]
```

Optional arguments:

- `--description|-d` : Display command description.
  default: False

1.8.3 - systeminfo

Generates a system/hardware info report on the standard output.

```
systeminfo [--output-file|-o <val>]
```

Optional arguments:

- `--output-file|-o <val>` : Instead of stdout, specifies a destination file within which the report should be written.
  default:

1.8.4 - cipher

Ciphers a value for use within configuration files (like `conf consolidate`). If called without -v, command will be interactive.

```
cipher [--value|-v <val>]
```

Optional arguments:

- *`--value|-v <val>`* :
  maxLength: 1024
  minLength: 1
  pattern: *`^(?!cipher=).*$`*

1.8.5 - cli conf ...

Manages the configuration file of the ∞CLI.

1.8.5.1 - cli conf init

Initializes the configuration file with default fields.

```
cli conf init [--location|-l <.|userscope|systemscope|auto>]
```

Optional arguments:

- *`--location|-l <.|userscope|systemscope|auto>`* : Predefined location of the configuration file. '`.`' will be in the current working directory. '*`userscope`*' will be in *`%APPDATA%`* on Windows and *`~/`* on Linux. '*`systemscope`*' will be in *`%PROGRAMDATA%`* on Windows and *`/etc/`* on Linux. '*`auto`* will use search path to find configuration file and will default to userscope if no existing file was found.
  default: "auto"
  enum: ['.', 'userscope', 'systemscope', 'auto']

1.8.5.2 - cli conf consolidate

Validates the content of the configuration file and ciphers all the sensitive fields. See the *`cipher`* CLI option to pre-cipher values.

```
cli conf consolidate [--location|-l <.|userscope|systemscope|auto>]
```

Optional arguments:

- *`--location|-l <.|userscope|systemscope|auto>`* : Predefined location of the configuration file. '`.`' will be in the current working directory. '*`userscope`*' will be in *`%APPDATA%`* on Windows and *`~/`* on Linux. '*`systemscope`*' will be in *`%PROGRAMDATA%`* on Windows and *`/etc/`* on Linux. '*`auto`* will use search path to find configuration file and will default to userscope if no existing file was found.
  default: "auto"
  enum: ['.', 'userscope', 'systemscope', 'auto']

1.8.5.3 - cli conf searchpath

Shows the current search path for the configuration file.

```
cli conf searchpath
```

1.9 - Diagnostic tools

1.9.1 - diag httpapi accesslog

Analyse the access logs of the HTTP API implementation (*.log and* .log_[0-9]+.back) to extract usage statistics.

```
diag httpapi accesslog <outputfolder> <filebasename> [--date-min <val>] [--no
auto] [--logfolder|-f <val>] [--no-recurse] [--workers|-w <val>]
```

Cancelable using CTRL+C/SIGTERM events.

Arguments:

- *<outputfolder>* : Result file path.
  example: C:/mystatsoutput
- *<filebasename>* : Base name of result files.
  minLength: 1

Optional arguments:

- *--date-min <val>* : Ignore log entries whose date is lower than this value.
  default: “1970-01-31”
  pattern: *[0-9]{4}-[0-9]{2}-[0-9]{2}*
- *--noauto* : Disable automatic log discovery (from conf_x_y.json files and default log locations).
  default: False
- *--logfolder|-f <val>* : Specify folder containing logs.
  default:
- *--no-recurse* : Disable recurse search for logs.
  default: False
- *--workers|-w <val>* : The amount of concurrent workers.
  default: 0

2 - ∞Directory HTTP API implementation

2.1 - version

Outputs the version of the tool on the standard output (stdout).

```
version [--output-file|-o <val>]
```

Optional arguments:

- *--output-file|-o <val>* : Instead of stdout, specifies a destination file within which the version string should be written.
  default:

2.2 - httpserver

Start an instance of ∞Directory API HTTP server.

```
httpserver [--conf-file|-c <val>] [--close-with-parent]
```

Cancelable using CTRL+C/SIGTERM events.

Optional arguments:

- `--conf-file|-c <val>` : Path to the ∞Directory configuration file. If left empty, it will default to '`%PROGRAMDATA%/3djuump-infinite-directory/conf_4_1.json`' on Windows or '`/etc/3djuump-infinite-directory/conf_4_1.json`' on Linux.
  default: ""
  maxLength: 1024
  minLength: 0
- `--close-with-parent` : If set HTTP server will exit if parent process is closed.
  default: False

2.3 - garbagecollector

Runs directory db and filer cleanup routines, this CLI should run with a maximum interval of 25 minutes.

```
garbagecollector [--conf-file|-c <val>] [--single-run] [--no-data-cleanup] [-
-force-data-cleanup|-f] [--close-with-parent] [--max-cleanup-workers|-w <val>
]
```

Cancelable using CTRL+C/SIGTERM events.

Optional arguments:

- `--conf-file|-c <val>` : Path to the ∞Directory configuration file. If left empty, it will default to '`%PROGRAMDATA%/3djuump-infinite-directory/conf_4_1.json`' on Windows or '`/etc/3djuump-infinite-directory/conf_4_1.json`' on Linux.
  default: ""
  maxLength: 1024
  minLength: 0
- `--single-run` : If set, garbage collector will run only once, else it will keep running waiting for changes.
  default: False
- `--no-data-cleanup` : If set DocStore and geometry pool cleanup will not be handled, usefull to ensure a quick exit as those operations are time consuming.
  default: False
- `--force-data-cleanup|-f` : If set cleanup will be executed even for storage whose cleanup is not scheduled (on first loop).
  default: False
- `--close-with-parent` : If set garbage collector will exit if parent process is closed.
  default: False
- `--max-cleanup-workers|-w <val>` : The maximum number of concurrent workers.
  default: 4
  maximum: 32
  minimum: 1

3 - ∞Directory Service

3.1 - run

Runs ∞Directory.

```
run [--conf-file|-c <val>]
```

Cancelable using CTRL+C/SIGTERM events.

Optional arguments:

- *`--conf-file|-c <val>`* : Path to the ∞Directory configuration file. If left empty, it will default to '*`%PROGRAMDATA%/3djuump-infinite-directory/conf_4_1.json`*' on Windows or '*`/etc/3djuump-infinite-directory/conf_4_1.json`*' on Linux.
  default: ""
  maxLength: 1024
  minLength: 0

3.2 - version

Outputs the version of the tool on the standard output (stdout).

```
version
```

3.2.1 - winservice run

Runs ∞Directory as a service.

```
winservice run
```

Available only on : windows.

3.2.2 - winservice register

Registers ∞Directory as a service.

```
winservice register <servicename> [--user-name <val>]
```

Arguments:

- *`<servicename>`* : Name of the service.

Optional arguments:

- *`--user-name <val>`* : Name of the user that will run the service.

Available only on : windows.

3.2.3 - winservice unregister

Unregisters ∞Directory as a service.

```
winservice unregister <servicename>
```

Arguments:

- *`<servicename>`* : Name of the service.

Available only on : windows.

4 - ∞Proxy Service

4.1 - run

Runs ∞Proxy.

```
run [--conf-file|-c <val>]
```

Cancelable using CTRL+C/SIGTERM events.

Optional arguments:

- *--conf-file|-c <val>* : Path to the ∞Proxy configuration file. If left empty, it will default to '*%PROGRAMDATA%/3djuump-infinite-proxy/conf_4_1.json*' on Windows or '*/etc/3djuump-infinite-proxy/conf_4_1.json*' on Linux.
  default: ""
  maxLength: 1024
  minLength: 0

4.2 - version

Outputs the version of the tool on the standard output (stdout).

```
version
```

4.2.1 - winservice run

Runs ∞Proxy as a service.

```
winservice run
```

Available only on : windows.

4.2.2 - winservice register

Registers ∞Proxy as a service.

```
winservice register <servicename> [--user-name <val>]
```

Arguments:

- *<servicename>* : Name of the service.

Optional arguments:

- *--user-name <val>* : Name of the user that will run the service.

Available only on : windows.

4.2.3 - winservice unregister

Unregisters ∞Proxy as a service.

```
winservice unregister <servicename>
```

Arguments:

- *`<servicename>`* : Name of the service.

Available only on : windows.

Web applications

title: 3D Juump Infinite
author: -
language: en
version: 2

Web applications are applications accessible through a Web browser, that are able to exploit the capabilities of 3D Juump Infinite using the 3D Juump Infinite Web API.

Third-party Web applications have to be registered or installed on the ∞*Directory Administration portal*. Then, the access to these Web applications will be controlled by the ∞Directory, and can be fine-tuned per user using tags.

1 - Package installation

The ∞Directory allows to install and serve ‘React-like Web applications’, without using a dedicated server. Web applications can be installed and then updated by uploading a zip package using the ∞*Directory Administration portal*. See Packaging a Web application for detailed information. Note that the resources of the application will be publicly available.

2 - External hosting

When the application is hosted on a dedicated server, the final redirect URL needs to be registered on the ∞Directory. In this situation, there is no constraint on the structure of the application but for a better integration it is recommended to follow the structure described in Packaging a Web application.

3 - Include Web API JavaScript file

It is preferable to use the JavaScript Web API delivered by the directory to automatically tak advantage of bug fixes without having to rebuild/redeploy your application. The JavaScript Web API is available on the following URL `https://directory_name:${apache_https_port}/{optionnal_prefix}directory/api/getwebapi?version=4.1`. You may (and should) adjust the `version` URL parameter to fetch the version needed by your application.

4 - Packaging a Web application

4.1 - Content

The zip package of your Web application must contain at least the following files:

- `index.html`
- `favicon.ico`
- `manifest.webmanifest`

4.2 - Name and URL

The **Home** and the **Authentication Redirect URL** will be in the form `https://directory_name:${apache_https_port}/{optionnal_prefix}directory/app/{webappurlname}/`, were `webappurlname` is the `applicationid` specified when installing the package on the *Administration portal*. The name and the description will be retrieved from the file `manifest.webmanifest`.

The name will be extracted from `short_name` if available, otherwise from `name`.

It is advised to include versioning information in the `description` field. If enclosed between C-style comment markers `/* my versioning information */`, this information will be hidden on the *Home page*.

4.3 - Technical constraints

To allow serving the Web application without knowing the base path and remain compatible with `ReactRouter`, a set of behaviors and rules are enforced.

4.3.1 - HTTP server

The Web server will have the following specific behaviors:

- If the app URL references an unexisting resource, `index.html` will be served instead of returning a 404 error. This allow to use `ReactRouter`.

- A file named *infinite.config.js* will be dynamically generated.
- When serving *index.html*, the following patterns will be automatically replaced:
  - */APP_BASE_URL/* by */{optionnal_prefix}directory/app/{webappurlname}/*
  - */DIRECTORY_BASE_URL/* by */{optionnal_prefix}directory/*
  - */getwebapi?version=API_VERSION* by */getwebapi?version=4.1*

The dynamically generated *infinite.config.js* file will allow to configure the Web application to function with the host ∞Directory at runtime.

```
// 3D Juump Infinite standard infinite.config.js file for self hosted
Web applications
// URL and Path will always endswith a slash '/'
// directory base URL
const directoryURL = 'https://directory_name:${apache_https_port}/{opt
ionnal_prefix}directory/'
// application home URL and trusted redirect URL
const appHomeURL = 'https://directory_name:${apache_https_port}/{optio
nnal_prefix}directory/app/{webappurlname}'
const appHomePath = '/{optionnal_prefix}directory/app/{webappurlname}'
const appName = '{webappurlname}'
window.config = { directoryURL , appHomeURL, appHomePath }
```

4.3.2 - Application structure

The following requirements should be met:

- *index.html* should contain exactly *<base href="/APP_BASE_URL/"*
- *index.html* should contain exactly *<link rel="manifest" href="./manifest.webmanifest"*
- *index.html* should contain exactly *<link rel="icon" href="./favicon.ico"*
- all the resources should be referenced using relatives path (eg : *./favicon.ico*)
- the fields *name* and *description* in the *manifest.webmanifest* file should not be empty nor missing
- the field *start_url* in the *manifest.webmanifest* file should be *./*
- the file size is limited to 4MB (after compression if supported)
- the file count is limited to 128
- the total file size is limited to 64MB (after compression if supported)

It is recommended to use the JavaScript Web API delivered by the ∞Directory by adding *<script src="/DIRECTORY_BASE_URL/api/getwebapi?version=API_VERSION"></script>* in *index.html*. Or a compatibility one by adding *<script src="/DIRECTORY_BASE_URL/api/getwebapi?version=X.Y"></script>*.

4.3.3 - React application tips

ReactRouter can be configured like this:

```
<Router basename= { window.config.appHomePath.length > 1 ? window.conf
ig.appHomePath.substring(0,window.config.appHomePath.length - 1) : '/'
}>
  <div className="overflow-hidden">
```

```
      <Routes>
        <Route path="*" element={<Navigate to='/'/>} />
      </Routes>
      ...
    </div>
  </Router>
```

5 - Application secure resources

It is possible to attach secure resources on application. Those resources will only be available to users that are authenticated on the application. This could be used to customize Kiwi with customer data that will be retained across software updates.

Third-party application protection

When deploying a webservice based on the Infinite Web API you may want to add a security layer to limit access to your Web application and data. This can be done by using a `directory session` access token or a `data session` session access token delivered by the ∞Directory.

1 - Defining user access rights

Add tags to users or teams that will be allowed to access your Web application. See [User access rights of third-party applications]()

For example, you can define two scopes destined to allow respectively read and write operations:

```
thirdpartyscopes:myapplication.read thirdpartyscopes:myapplication.wri
te
```

2 - Request scope

When opening an ∞Directory session from the Web API, add the desired application claims.

```
let lDirectorySession : DirectorySessionInterface = ...;
lDirectorySession.getPopupBasedAuthenticationUrl(...,['myapplication.r
ead','myapplication.write']);
```

3 - Protect your Web application

Use the Directory session access token returned by

```
let lDirectorySession : DirectorySessionInterface = ...;
lDirectorySession.getAuthenticationBearer();
```

or the Data session access token returned by

```
let lDataSession : DataSessionInterface = ...;
lDataSession.getDataSessionBearer();
```

in HTTPS calls to your backend (with the header *Authorization: Bearer <token>* or as a query parameter).

4 - Validate token and claims

First, validate the received token using */directory/api/.well-known/jwks.json* or *directory/api/instrospect* endpoints. Then ensure that the *scope* field contains the expected claims. Access tokens signature could be validated using the public key available on Json Web Key Store or by introspection OAuth 2.0 Token Introspection.

Directory session access tokens will only contain application scopes. It allows you to ensure that the user is logged on the ∞Directory and has the expected claims. In addition, Data session access tokens contain extra information related to the Project and Build, that may be used to limit access based on project access rights.

Example of a Directory session access token

```
{
  "access:client": true,
  "exp": 1609866581,
  "iat": 1609866282,
  "ip": "...",
  "iss": "https://xxxx:443/directory",
  "native": false,
  "nbf": 1609866282,
  "oidcsub": "...",
  "salt": "7b4pzw",
  "scope": "myapplication.read myapplication.write",
  "session_eol": 1609952681,
  "sessionid": "1ab88920ebc14f3baee8ac4eb5aa301e",
  "tokentype": "TT_Directory"
}
```

Example of a Data session access token

```
{
  "aud": "https://xxxx:443/directory",
  "buildgrantedtags": [
    "MyTag"
  ],
  "buildid": "{00000000-0000-0000-0000-000000000000}",
  "datasessionid": "745f5e0ca8714968a8bd9e467dfa71ec",
  "directorysessionid": "ba1061a601fa431bb8cf632295251619",
  "exp": 1685980994,
  "geometrypoolid": "geom_00000000000000000000000000000000",
  "iat": 1685977394,
  "iphash": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
  "iss": "https://xxxx:443/directory",
  "nbf": 1685977394,
  "projectid": "prj_00000000000000000000000000000000",
  "salt": "WewHWg",
  "scope": "myapplication.read myapplication.write",
  "tokentype": "TT_Data_Session"
}
```

Native client

The ∞Client is composed of several software components:

- the *Hub* is a central element responsible for user management, token management, add-on management, etc.
- the *Browser* is a mandatory add-on that is launched from the *Hub* and lets the (authenticated) user visualize and interact with the chosen DMU.
- the *Local DMU Manager* is an optional add-on that adds the ability to install data locally in a *standalone*, offline mode (see below).

1 - Modes

The ∞Client supports two distinct modes:

- In *online* mode, the ∞Client is connected to a remote ∞Directory and ∞Proxy and browses an *online DMU*, i.e. data streaming from the 3D Juump Infinite cluster.
- In *standalone* mode, the ∞Client browses a *standalone DMU*, i.e. local data installed from *EvoJuump* files.

1.1.1 - Online mode

The *online* mode requires:

- that the user can contact (in the network sense) a remote ∞Directory and an ∞Proxy,
- that the user has valid credentials for this ∞Directory,
- that the credentials give access to at least one *online DMU*,
- that the conditions of the licence installed on the ∞Directory allow it to open a new datasession.

Once the user is properly authenticated and once a DMU has been selected, the *Hub* launches the *Browser*. One datasession is locked on the ∞Directory until the *Browser* is closed. Should the local computer lose its datasession while the *Browser* is running (due to a network error, for instance), the *Browser* will forcibly close a few minutes after issuing a warning. The user will still be able to save any content they produced.

1.1.2 - Standalone mode

The *standalone* mode requires:

- that the *Local DMU Manager* add-on is installed and running on the local computer,
- that a valid non-expired *node-locked license* is installed on the local computer or a non-expired *borrowed token* from an LM-X server,
- that the user has installed a *standalone DMU*,
  - by importing a valid EvoJuump file to the *Hub*,
  - by providing the password used to cipher the EvoJuump file.

Once a *standalone DMU* has been selected, the *Hub* launches the *Browser*. Should the *node-locked license* expire while the *Browser* is running, the *Browser* will forcibly close a few minutes after issuing a warning. The user will still be able to save any content they produced.

2 - Installation

2.1 - Installing the *Hub* and *Browser* add-on

Note: the *Browser* add-on is automatically installed with the *Hub*.

2.1.1 - Requirements

The client application can run on a broad range of CPUs, from a mobile to a high-end x86-64 central processing unit (CPU). The amounts of CPU power and required RAM depend on the complexity of your DMU and the availability of a dedicated graphic processing unit (GPU).

A dedicated graphic card (GPU) is not required but when it is available, it should be installed with an OpenGL 3.1 and above driver. Any NVIDIA GeForce 8 and above, AMD Radeon 4xxx and above or Intel HD4000 and above may fit. Without a dedicated GPU, the 3D Juump client application is able to run with the integrated GPU.

Eventually, you should also reserve enough hard disk space to store your digital mock-up data and workspaces. The disk space required for the installation of the ∞Client application and the Local DMU Manager is only a negligible fraction of the occupied size, respectively 200MB and 800MB.

Minimal requirements of the ∞Client:

- **Windows 11 64-bit version**
- Dual Core CPU x86-64
- 4GB RAM
- 4GB disk space for binaries and caches
- GPU with OpenGL 3.1 and GLSL 140 support with 3GB RAM
- 1280 x 800 pixel display

Recommended requirements of the ∞Client:

- Windows 11 64-bit version
- Quad Core CPU x86-64
- 6GB RAM
- 8GB disk space for binaries and caches (SSD)
- GPU with OpenGL 3.1 and GLSL 140 support with 4GB dedicated RAM
- 1920 x 1080 pixel display

Minimal requirements for Web Applications / Web Browser:

- Dual Core CPU x86-64
- 8GB RAM
- WebGL 2.0 support
- `deflate` and `br` decompression support
- 64 bit browser, Chromium based, Firefox or Safari (tested on latest Chromium)

Recommended requirements for Web Applications / Web Browser:

- Quad Core CPU x86-64
- Dedicated GPU
- 64 bit Chromium based browser

A keyboard and a mouse are mandatory for fully using 3D Juump Infinite. Touchscreen support is limited to navigation.

When using the Local DMU Manager, it is recommended to add **+2 cores** to the CPU and add **+2GB** of RAM, for the minimal and recommended requirements above.

⚠ In order to install the Local DMU Manager on the user host, you must have an **administrator** account. Note the *administrator* account is only required for the installation. Once the application is deployed, the user can have access to their data with their regular account.

2.1.2 - Installation procedure

To install the 3D Juump Infinite *Hub* and *Browser*:

- Double click on the `3DJuump Infinite X64-setup-admin-2.x.x.xxx.exe` installer (requires administrator privileges).
- Validate the licence agreement.
- If required, change the installation folder.

- Then, if needed, modify the program folder in the start menu and choose if you want to create a shortcut on the desktop.
- Click on the `Install` button to start the installation.

The installer will deploy the *Hub* application and its mandatory *Browser* add-on, plus any dependencies (including *Visual C++ Redistributable*).

Note: an alternative installer is also available for standard users lacking administrator privileges. This installer does not deploy any *Redistributable*, hence it is only usable on computers already equipped with the following:

- Visual C++ Redistributable 2015 x64
- Visual C++ Redistributable 2008 x64 SP1 ATL

2.1.3 - Default settings configuration

2.1.3.1 - Location

The settings of the *Browser* application are stored in `settings.ini` files found in the following locations:

- `%PROGRAMDATA%\3DJuumpInfiniteX64\settings.ini` (*global* settings for all users)
- `%APPDATA%\3DJuumpInfiniteX64\settings.ini` (*current user*'s settings)

2.1.3.2 - Resolution order

For a given settings key, the value is read from the global settings location first and then, if the key is not present, it is loaded from the current user's settings. It means that the administrator is allowed to force a settings for all users. For example, the administrator can ship a predefined global `settings.ini` file with his installation program.

2.1.3.3 - Useful settings keys

Several settings can (and probably *should*) be set by the administrator:

- `Security/verifySSLPeer`: Tells 3D Juump Infinite whether it should verify the SSL certificates of the ∞Directory and ∞Proxy it connects to. Turn it off when using internal self-signed certificates.
- `Security/enableHttpProxy`: Tells 3D Juump Infinite whether it should use the system-defined HTTP proxy for communicating with the 3D Juump Infinite cluster.
- `Security/SSLClientCertificate`, `Security/SSLClientKey` and `Security/SSLClientKeyPassword`: Tells 3D Juump Infinite the location of the client certificate that should be used for mTLS support. The format can be PEM or P12 (`Security/SSLClientKey` should be empty).
- `HUB/allowLocalAccount`: Tells the *Hub* whether it should display the local account. Turn it off if you don't ever want your users to work in standalone mode.

2.1.3.4 - Pre-registering an ∞Directory

An ∞Directory may be pre-registered per user by adding an entry in the `%APPDATA%/3DJuumpInfiniteX64/directories_4_1.json` file, or globaly by adding an entry in the

%PROGRAMDATA%/3DJuumpInfiniteX64/directories_4_1.json file. Global entries can’t be removed from the *Hub*.

```
[
    {
        "directoryName": "MyDirectory",
        "directoryUrl": "https://hostname/directory",
        "licenseServer": "",
        "serverLicenseFeature": ""
    }
]
```

2.2 - Installing the *Local DMU Manager* add-on

2.2.1 - Requirements

To run the *Local DMU Manager*, the local computer must meet some minimal requirements:

- Windows 11 64-bit version
- Quad Core CPU x86-64
- 6GB RAM
- at least 20GB disk space for binaries and local DMUs

Note: depending on the size and number of your local DMUs, the *Local DMU Manager* may require more disk space.

Note: the *Local DMU Manager* will take advantage of an SSD to store the local DMUs.

2.2.2 - Installation procedure

To install the 3D Juump Infinite *Local DMU Manager*:

- Double click on the *3DJuump Local DMU Manager X64-setup-2.x.x.xxx.exe* installer (requires administrator privileges). Please note that the versions of the Local DMU Manager and the Hub must be the same.
- Validate the licence agreement.
- If required, change the installation folder.
- Then, if needed, modify the program folder in the start menu and choose if you want to create a shortcut on the desktop.
- Click on the *Install* button to start the installation.

The installer will deploy the *Local DMU Manager* add-on plus any dependencies, including the following *Visual C++ Redistributable*:

- Visual C++ Redistributable 2015 x64
- Visual C++ Redistributable 2010 x86
- Visual C++ Redistributable 2008 x64 SP1 ATL

2.2.3 - Local Web application

It is possible to run a single page Web application in *Standalone mode*. Edit *%PROGRAMDATA%\3DJuumpLocalDMUManagerX64\Settings.ini* to add the following entry

`LocalWebApps=path_to_web_app_folder` in the *[localserver]* section. The Web application folder should contain a subfolder per Web application. Each subfolder must be organized in the same fashion as a Web application package destined to an ∞Directory.

⚠ see Local Web Application limits

2.2.4 - ElasticSearch configuration

Depending on computer available RAM and size of DMU it might be needed to adjust amount of memory dedicated to ElasticSearch instance. This is achieved by editing `%AppData%/3DJuumpLocalDMUManagerX64/databases_4.1/elastisearch/config/jvm.options`. See ElasticSearch documentation for details.

Troubleshooting

This chapter provides hints for error troubleshooting

1 - Backend or Native Client executable error

When an executable returns an error following steps may help to understand what gone wrong :

- Check if return code matches a known ProcessReturnCodes.
- Check associated log for error details.
- Enable diagnostic mode in settings and restart execution. Depending on executable setting might differ. See loglevel for backend components and enablediag in Settings.ini for Native Client.
- If process crashes, check system logs (`eventvwr` on Windows, `syslog` on Linux), application logs and `std::cerr` stream.

2 - HTTP API or Web Client error

When a HTTP API call returns an error following steps may help to understand what gone wrong :

- Identify source of the error. All Infinite backend components returns a json object error.
- Check logs of components which has raised the error. Use timestamp or error uuids to locate corresponding events.

3 - Process return codes

Code	Name	Description
0	PRC_Ok	Execution went successfull.
232	PRC_NoDirectoryCredentials	Invalid directory credentials.
233	PRC_SystemError	A system error occurs during process initialization. Check process and/or system logs for details.
234	PRC_InitError	An error occurs during process initialization. Check process and/or system logs for details.
235	PRC_MissingOrInvalidConfFile	Configuration file is missing or invalid.
236	PRC_FailToOpenFile	Fails to open file. Ensure that file path is correct and that process has access rights.
237	PRC_OutOfMemory	Process runs out of memory.
239	PRC_Unauthorized	Operation requires an authorization from the Infinite Directory. Ensure that the process can contact the Directory and that your license allows this operation.
240	PRC_ConvertFailToInitializeCoreTechnologie	Fail to initialize CoreTechnologie CAD plugin.
241	PRC_InvalidJobFile	Invalid input job file. Check documentation for details.
242	PRC_FailToAcquireLock	Fail to acquire exclusive lock.
243	PRC_ConfFileSaveError	Fail to save configuration file. Check log for details.
244	PRC_ConfFileLoadError	Fail to load configuration file. Check log for details. Ensure that configuration file was consolidated with same cli version.
245	PRC_ArgumentParseError	Fail to parse argument. Check documentation for details.
246	PRC_ArgumentUnexpectedValue	Unexpected argument value. Check documentation for details.
247	PRC_ArgumentWrongCount	Invalid arguments count. Check documentation for details.
248	PRC_UninitializedError	Error code was not set.
249	PRC_Error	Generic error code. Check process logs for details.
250	PRC_Canceled	Execution was canceled by caller or external source.
251	PRC_FailToResolveExeAbsPath	Fail to resolve process absolute path.
252	PRC_FailToRetriveArguments	Fail to parse command line arguments. Check process logs and documentation for details.
253	PRC_GetProcAddressFailed	Fail to locate library entry points. This denote a binary incompatibility error. Try to reinstall the software.
254	PRC_LoadLibraryFailed	Error while loading binaries (.dll, .so, ...). This denote a dependency error. Try to reinstall the software.
255	PRC_Crash	Sometimes on crash, system will return this value. Investigate process and system logs to gather more details.

4 - HTTP API error object

HTTP API endpoints error

object

Error object return by HTTP API endpoints.

- `source` : **string**, length (**[0;32]**)

  Information on error emitter.

- `error` : **boolean**, *REQUIRED*, values (**true**)
- `datetime` : **string**
- `errorinfo` : **array**
  - items : **string**
- `code` : **string**, *REQUIRED*, values (**E_BadContentType**, **E_BadMethod_405**, **E_Conflict_409**, **E_Disabled_403**, **E_Forbidden_403**, **E_InternalError**, **E_InternalServerError_500**, **E_InvalidBody**, **E_InvalidHttpMethod**, **E_InvalidIpStack**, **E_InvalidLicense_403**, **E_InvalidParameter**, **E_InvalidRequest**, **E_Missing_404**, **E_No_Error**, **E_ServiceUnavailable_503**, **E_TooManyRequests_429**, **E_Unauthorized_401**, **E_UnknownError**)

  `E_BadContentType`: `E_BadMethod_405`: `E_Conflict_409`: `E_Disabled_403`: `E_Forbidden_403`: `E_InternalError`: `E_InternalServerError_500`: `E_InvalidBody`: `E_InvalidHttpMethod`: `E_InvalidIpStack`: `E_InvalidLicense_403`: `E_InvalidParameter`: `E_InvalidRequest`: `E_Missing_404`: `E_No_Error`: `E_ServiceUnavailable_503`: `E_TooManyRequests_429`: `E_Unauthorized_401`: `E_UnknownError`:

- `uuid` : **string**, *REQUIRED*

  An identifier to retrieve more information from backend logs, if provided by the client this will contain the value of the X-Request-ID header.

5 - 3D Juump Infinite node status codes

3D Juump Infinite ∞Directory, ∞Proxy and ∞AsyncJobSolver will report a status `ok`, `warning`, `critical`. This status is completed by enums that describe issues.

Status reason	Description
AsyncJobSolversNeedAttention_Warning	Some 3D Juump ∞AsyncJob Solvers need attention. Some 3D Juump ∞AsyncJob Solvers are in a 'critical' or 'warning' status. Check the Administration Portal and 3D Juump Infinite logs.
DatabaseBadVersion_Critical	The Database version is incorrect. The Database schema does not match the schema expected by the 3D Juump Infinite binaries.
DevVersion_Warning	This component is using a development version. This component's version is not a stable version. It should not be maintained in production.

ElasticSearchBadStatus_Critical	The ElasticSearch status is not ok. The ElasticSearch server status is not ‘green’ (ok) or ‘yellow’ (warning). Check the ElasticSearch logs.
ElasticSearchBadVersion_Critical	The ElasticSearch server version is incorrect. The ElasticSearch server version is not compliant with 3D Juump Infinite requirements.
ElasticSearchNetworkError_Critical	The ElasticSearch server is not responding. Check the 3D Juump ∞Proxy and ElasticSearch logs.
ElasticSearchOutdated_Warning	The ElasticSearch server is outdated. The ElasticSearch server version is older than the one packaged with the release of 3D Juump Infinite.
GarbageCollectorDown_Critical	The ∞Directory garbage collector is down. It does not send any life signal. Check the 3D Juump Infinite logs.
HeterogeneousVersions_Warning	Backend components binaries version mismatch. This could lead to incompatibilities or denote missing updates.
HttpApiBadVersion_Critical	HTTP API version incompatibility. Usually implied by a component that was not updated alongside the others.
LogLevelDebug_Warning	The log level of this component is set to DEBUG. This could negatively impact the performances or disclose sensitive data. This log level should not be maintained in production.
NoAsyncJobSolver_Warning	No 3D Juump ∞AsyncJob Solvers are available. Export features will not be available on the Web applications.
NoLicense_Critical	The ∞Directory has no valid license. Check the Administration Portal and 3D Juump Infinite logs.
NoProxy_Critical	No 3D Juump ∞Proxies are available. No 3D Juump ∞Proxies are registered, or they are all down. Check the Administration Portal and 3D Juump Infinite logs.
PostgreSQLBadVersion_Critical	The PostgreSQL server version is incorrect. The PostgreSQL server version is not compliant with 3D Juump Infinite requirements.
PostgreSQLFetchError_Critical	Failed to fetch data from the PostgreSQL server. Check the 3D Juump ∞Proxy and PostgreSQL logs.
PostgreSQLNetworkError_Critical	The PostgreSQL server is not responding. Check the 3D Juump ∞Proxy and PostgreSQL logs.
PostgreSQLOutdated_Warning	The PostgreSQL server is outdated. The PostgreSQL server version is older than the one packaged with the release.
ProxiesNeedAttention_Warning	Some 3D Juump ∞Proxies need attention. Some 3D Juump ∞Proxies are in a ‘critical’ or ‘warning’ status. Check the Administration Portal and 3D Juump Infinite logs.
ProxyContactLost_Critical	The connection to the ∞Proxy is down. The ∞Directory cannot contact this ∞Proxy. The ∞Proxy service may be down, or it may be an HTTP(s) connection problem.
SSLVerifyDisabled_Warning	The SSL certificate validation is disabled. It is disabled for at least one scope (default, backend to backend, OpenId Connect). It should not be maintained in production.
ServiceBadVersion_Critical	Service internal version incompatibility. Usually implied by a component that was not updated alongside the others.
ServiceDown_Critical	The service is down. It does not send any life signal. Check the 3D Juump Infinite logs.
Shutdown_Critical	This component is currently shut down. Part of your Infinite services may be unavailable.
Unknown_Critical	Unknown reason.

Annexes

These annexes contain:

- the description of ∞Directory, ∞Proxy and ∞CLI configuration files
- the release notes,
- the security guidelines,
- the list of third-party licenses,
- the list of known limitations.

1 - Configuration files

1.1 - Generalities on configuration files

Configuration files are JSON documents. They can be initialized using ∞CLI * conf init, and consolidated (to cipher sensitive data) using ∞CLI * conf consolidate.

see JSON limitations

1.2 - ∞Directory configuration

The ∞Directory configuration file is located in `/etc/3djuump-infinite-directory/conf_4_1.json` on Linux and in `%PROGRAMDATA%\3djuump-infinite-directory\conf_4_1.json` on Windows.

1.2.1 - Schema of the configuration file

∞Directory configuration file

object

- *`$schema`* : **string**, length (**[1;64]**), pattern (`^\./.*\.schema\.json$`)
- *`directoryapi`* : **object**, *REQUIRED*
  - *`apikey`* : **oneOf**, *REQUIRED*
    - **null**

      If null HTTP.*_key authentication methods will be disabled.
    - **string**, length (**[1;+∞]**)

      API key secret used to protect API endpoints, this secret will be used as HTTP Basic authentication with 'infinite' login. It will be preferred over OAuth2 method if both are specified.
  - *`backendvhosts`* : **array**, length (**[0;7]**)
    - items : Optional directory API vhost for backend or administration usage, if specified this vhost will not accept client security schemes. Using separate vhosts allows to implement custom security rules (mTLS, ip filtering, ...) on the HTTP gate based on API usage. see Directory VHost
  - *`bindport`* : **integer**, *REQUIRED*, range(**]1;65535]**)

    Bind port for ∞Directory API HTTP implementation.
  - *`httpaccesslog`* : **boolean**, default (**true**)

    Enables log of received HTTP requests.
  - *`jwtcachelifetime`* : **integer**, *REQUIRED*, range(**]0;500]**)

    How long (in secondes) a bearer is kept in cache.
  - *`publicbind`* : **boolean**, default (**false**)

    Specify if directory API HTTP implementation should listen on any addresses, if false only loopback will be bound.
  - *`publicvhost`* : Main directory vhost always used by client applications. If a backendvhost security schemes is defined, implicit restrictions will be applied to this vhost. Should use HTTPS ! see Directory VHost
  - *`hostsearchorder`* : Define which headers should be used to determine host and port used by the client.Unfortunately 2 sets of headers ('Forwarded' and 'X-Forwarded-*') exist for reverse proxy. So depending on your infrastructure you might need to change evaluation order. Evaluation will stop on the first header found. If no headers

were found the 'Host' header will be used. Sometimes exotic configuration (like AWS) may preserve the Host header, add X-forwarded-port and discard X-forwarded-Host, in this case, the policy host-with-x-forwarded-port may be used. see Host search order

  - *disableauthforgetversion* : **boolean**, default (**false**)

    If true, authorization and vhost check will be disabled for /api/getversion endpoint. This is useful to run healthcheck of containers behind a load balancer were load balancer will use internal IP or hostname.

- *http_client_configuration* : HTTP configuration. see Http client global configuration
- *log* : see Log configuration
- *openidconnect* : **object**, *REQUIRED*
  - *common* : Common OpenId Connect settings. see OpenID connect common settings
  - *user_auth* : Configure user identification and session access token using OpenId Connect code flow. see OpenID Connect user identification ∞Directory settings
  - *m2m_auth* : **oneOf**, *REQUIRED*
    - **null**

      No OAuth2 configuration for machine to machine communication, HTTP.m2m_bearer will be disabled and API key will be used.

    - Configure OAuth2 machine to machine identification using OpenId Connect client credentials flow. See HTTP.m2m_bearer authentication method. Those settings will be used to acquire a token and to validate received tokens. see OpenID Connect M2M settings
- *postgres* : see PostgreSQL configuration
- *filerstorage* : **oneOf**, *REQUIRED*
  - **object**

    Filer storage implemented using OS or network share filesystem. Used folder should be dedicated to this ∞Directory as it will automatically create/delete files !

    - *type* : **string**, *REQUIRED*, const (**filesystem**)
    - *folder* : **string**, *REQUIRED*, length (**[1;+∞]**)

      Folder in which data will be stored. If relative, will be resolved relative to job file.

  - **object**

Filer storage implemented using a Microsoft Azure storage account. This storage account should be dedicated to this ∞Directory as it will automatically create/delete blob containers !

- *`http_client_configuration`* : HTTP configuration for calls to Azure blob rest API. see Http client override configuration
- *`type`* : **string**, *REQUIRED*, const (**azureblob**)
- *`url`* : **string**, *REQUIRED*, length (**[1;+∞]**)

  Azure storage account URL.
- *`storage_account`* : **string**, *REQUIRED*, length (**[1;+∞]**)

  Azure storage account name. Storage account name could not be deduced from the URL as URL format might differ due to use of reverse proxy or azurite. Example is the default storage account for Azurite emulator.
- *`shared_key`* : **string**, *REQUIRED*, length (**[0;4194304]**), pattern (*`^(?:[A-Za-z0-9+\/]{4})*(?:[A-Za-z0-9+\/]{2}==|[A-Za-z0-9+\/]{3}=)?$`*)

  Azure storage shared key. Example is the default shared_key for Azurite emulator.

- **object**

  Filer storage implemented using an Amazon S3 or Minio bucket storage. This bucket should be dedicated to this ∞Directory !

  - *`http_client_configuration`* : HTTP configuration for calls to bucket storage rest API. see Http client override configuration
  - *`type`* : **string**, *REQUIRED*, const (**s3bucket**)
  - *`url`* : **string**, *REQUIRED*, length (**[1;+∞]**)

    Bucket URL.
  - *`region`* : **string**, *REQUIRED*, length (**[1;+∞]**)

    AWS region, for Minio use us-east-1.
  - *`access_key`* : **string**, *REQUIRED*, length (**[1;+∞]**)

    The ‘public’ access key.
  - *`secret_key`* : **string**, *REQUIRED*, length (**[1;+∞]**)

    The ‘private’ secret key.

1.2.2 - Schema of an ∞Directory virtual host

Directory VHost

object

Directory Virtual Host definition.

- *`url`* : **string**, *REQUIRED*, length (**[0;1024]**), pattern (*`^https?:\/\/[^@\/A-Z]+?(:[1-9][0-9]{0,4})(\/.*)?\/directory$`*)

  Directory API URL for public usage. The port has to be explicited as this attribute is part of the license.

- *`endpoints`* : **object**, *REQUIRED*

  Allow to disable specified API endpoints.

  - *`enabledocstoreeditor`* : **boolean**, default (**false**)

    If set, docstoreeditor gui (/directory/docstoreeditor) will be available. This gui should not be used in production.

  - *`enablegetsuperadminregistrationcode`* : **boolean**, default (**false**)

    Unset this to disable /directory/api/directorysession/requestsuperadminregistrationcode and /directory/api/backend/registersuperadmin.

  - *`enablewebapidoc`* : **boolean**, default (**false**)

    If set, Web API documentation will be available under /directory/webapidoc/. This is not recommanded for production servers.

- *`securityflows`* : **oneOf**, *REQUIRED*

  Allows to restrict security flows accepted on this vhost.

  - **null**

    Accept all security schemes.

  - **array**, length (**[1;+∞]**)

    List of security flows accepted by the ∞Directory.

    - items : **string**, values (**app.admin**, **app.client**, **back.admin**, **back.connector**, **back.infinite**)

- *`securityschemes`* : **oneOf**, *REQUIRED*

  Allows to restrict security schemes accepted on this vhost.

  - **null**

Accept all security schemes.

- **array**, length (**[1;+∞]**)

  List of security schemes accepted by the ∞Directory.

  - items : **string**, values (**http.directory_key**, **http.m2m_bearer**, **http.session_bearer**, **infinitebearer.data_session**, **infinitebearer.data_session_download_token**, **infinitebearer.data_session_extended**, **infinitebearer.directory_session**, **infinitebearer.directory_session_download_token**, **infinitebearer.directory_session_extended**, **infiniteprivate**)

1.2.3 - Schema of HTTP Host search order

Host search order

array, distinct

Define which headers should be used to determine host and port used by the client.Unfortunately 2 sets of headers (‘Forwarded’ and ’X-Forwarded-*‘) exist for reverse proxy. So depending on your infrastructure you might need to change evaluation order. Evaluation will stop on the first header found. If no headers were found the ’Host’ header will be used. Sometimes exotic configuration (like AWS) may preserve the Host header, add X-forwarded-port and discard X-forwarded-Host, in this case, the policy host-with-x-forwarded-port may be used.

- items : **string**, values (**forwarded**, **x-forwarded-***, **host-with-x-forwarded-port**)

1.3 - ∞Proxy configuration

The ∞Proxy configuration file is located in `/etc/3djuump-infinite-proxy/conf_4_1.json` on Linux and in `%PROGRAMDATA%\3djuump-infinite-proxy\conf_4_1.json` on Windows.

1.3.1 - Schema of the configuration file

∞Proxy configuration file

object

- `$schema` : **string**, length (**[1;64]**), pattern (`^\./.*\.schema\.json$`)
- `directoryapi` : **object**, *REQUIRED*
  - `apikey` : **oneOf**, *REQUIRED*

    ∞Directory API key, if null m2m bearer will be used.

    - **null**

      If null HTTP.*_key authentication methods will be disabled.

    - **string**, length (**[1;+∞]**)

API key secret used to protect API endpoints, this secret will be used as HTTP Basic authentication with 'infinite' login. It will be preferred over OAuth2 method if both are specified.

- `url` : **string**, *REQUIRED*, length (**[0;1024]**), pattern (`^https?:\/\/[^@\/A-Z]+?(:[1-9][0-9]{0,4})(\/.*)?\/directory$`)

  URL to the directory, preferably a backend vhost.

- `elasticsearch` : **object**, *REQUIRED*

  Host and port of elasticsearch node HTTP interface. The node should be fully dedicated to this proxy. It is preferable to host the node on the same server as proxy service.

  - `url` : **string**, *REQUIRED*, length (**[0;1024]**), pattern (`^http:\/\/[^@\/]+?(:[1-9][0-9]{0,4})(\/.*)?$`)

    URL to the local elasticsearch cluster.

  - `login` : **string**, default (****), length (**[0;+∞]**)

    Elasticsearch connection login, could be empty if xpack.security is disabled.

  - `password` : **string**, default (****), length (**[0;+∞]**)

    Elasticsearch connection password, could be empty if xpack.security is disabled.

- `debugdocriver` : **boolean**, default (**false**)

  Enables rfDebug for doc river.

- `http_client_configuration` : HTTP configuration. see Http client global configuration
- `log` : see Log configuration
- `openidconnect` : **object**, *REQUIRED*
  - `common` : Common OpenId Connect settings. see OpenID connect common settings
  - `user_auth` : see OpenID Connect user identification ∞Proxy settings
  - `m2m_auth` : **oneOf**, *REQUIRED*
    - **null**

      No OAuth2 configuration for machine to machine communication, HTTP.m2m_bearer will be disabled and API key will be used.

    - Configure OAuth2 machine to machine identification using OpenId Connect client credentials flow. See HTTP.m2m_bearer authentication method. Those settings will be used to acquire a token and to validate received tokens. see OpenID Connect M2M settings

- `postgres` : see PostgreSQL configuration
- `proxyapi` : **object**, *REQUIRED*
  - `apikey` : **oneOf**, *REQUIRED*
    - **null**

      If null HTTP.*_key authentication methods will be disabled.
    - **string**, length (**[1;+∞]**)

      API key secret used to protect API endpoints, this secret will be used as HTTP Basic authentication with 'infinite' login. It will be preferred over OAuth2 method if both are specified.
  - `backendvhost` : **oneOf**, default (**null**)
    - **null**
    - Optional proxy API vhost for backend usage (used by proxy service and directory), if specified this vhost will not accept client security schemes. This vhost should be accessible by all directory instances. see Proxy VHost
  - `bindport` : **integer**, *REQUIRED*, range(**]1;65535]**)

    Bind port for proxy API HTTP implementation.
  - `enablehttpaccess` : **boolean**, default (**true**)

    Enables log of received HTTP requests.
  - `jwtcachelifetime` : **integer**, *REQUIRED*, range(**]0;500]**)

    How long (in secondes) a bearer is kept in cache.
  - `publicvhost` : Main proxy vhost always used by client. If a backendvhost security schemes is defined, implicit restrictions will be applied to this vhost. Should use HTTPS ! see Proxy VHost
  - `publicbind` : **boolean**, default (**false**)

    Specify if proxy API HTTP implementation should listen on any addresses, if false only loopback will be bound.
  - `httpaccesslog` : **boolean**, default (**true**)

    Enables log of received HTTP requests.
  - `hostsearchorder` : Define which headers should be used to determine host and port used by the client.Unfortunately 2 sets of headers ('Forwarded' and 'X-Forwarded-*') exist for reverse proxy. So depending on your infrastructure you might need to change evaluation order. Evaluation will stop on the first header found. If no headers

were found the 'Host' header will be used. Sometimes exotic configuration (like AWS) may preserve the Host header, add X-forwarded-port and discard X-forwarded-Host, in this case, the policy host-with-x-forwarded-port may be used. see Host search order

  - `disableauthforgetversion` : **boolean**, default (**false**)

    If true, authorization and vhost check will be disabled for /api/getversion endpoint. This is useful to run healthcheck of containers behind a load balancer were load balancer will use internal IP or hostname.

- `workingfolder` : **string**, *REQUIRED*

  Working folder were ∞Proxy data are stored. If relative, will be resolved relative to job file.

- `replication` : **object**, *REQUIRED*

  Defines replication restriction, allowing to retrieve only a subset of builds available on the Directory.

  - `oneoftags` : **array**, *REQUIRED*, length (**[0;+∞]**), distinct

    To be replicated a build should have at least one tag from this list (if the list is not empty).

    - items : **string**, length (**[1;64]**), pattern (`^[\x21\x23-\x39\x3c-\x5B\x5d-\x7e]+$`)

      Tag definition, because tags are also used as scopes limit them to scopes acceptable chars minus ':' and ';'. See https://www.rfc-editor.org/rfc/rfc6749#section-3.3.

  - `noneoftags` : **array**, *REQUIRED*, length (**[0;+∞]**), distinct

    To be replicated a build should not have a tag from this list (if the list is not empty).

    - items : **string**, length (**[1;64]**), pattern (`^[\x21\x23-\x39\x3c-\x5B\x5d-\x7e]+$`)

      Tag definition, because tags are also used as scopes limit them to scopes acceptable chars minus ':' and ';'. See https://www.rfc-editor.org/rfc/rfc6749#section-3.3.

- `asyncjobsolver` : Configuration of asynchronous job solver. see Asynchronous job solver

1.3.2 - Schema of an ∞Proxy virtual host

Proxy VHost

object

Proxy Virtual Host definition.

- *`url`* : **string**, *REQUIRED*, length (**[0;1024]**), pattern (*`^https?:\/\/[^@\/A-Z]+?(:[1-9][0-9]{0,4})(\/.*)?\/proxy$`*)
- *`securityflows`* : **oneOf**, *REQUIRED*

  Allows to restrict security flows accepted on this vhost.

  - **null**

    Accept all security schemes.

  - **array**, length (**[1;+∞]**)

    List of security flows accepted by the ∞Proxy.

    - items : **string**, values (**app.admin**, **app.client**, **back.admin**, **back.connector**, **back.infinite**)
- *`securityschemes`* : **oneOf**, *REQUIRED*

  Allows to restrict security schemes accepted on this vhost.

  - **null**

    Accept all security schemes.

  - **array**, length (**[1;+∞]**)

    List of security schemes accepted by the ∞Proxy.

    - items : **string**, values (**http.m2m_bearer**, **http.proxy_key**, **http.session_bearer**, **infinitebearer.data_session**, **infiniteprivate**)

1.3.3 - Schema ∞AsyncJobSolver configuration on an ∞Proxy

Asynchronous job solver

object

Configuration of asynchronous job solver.

- *`enable`* : **boolean**, *REQUIRED*

  Enable or disable asynchronous job handling.

- *`maxmemorymb`* : **integer**, *REQUIRED*, range(**]1024;65536]**)

  Maximum memory that the solver is allowed to dedicate for workers. See administration manual to check memory requirements per job types.

- *`maxcpu`* : **integer**, *REQUIRED*, range(**]1;64]**)

  Maximum cpu count that the solver is allowed to dedicate for workers.

- *`supportedjobtypes`* : **array**, default (**[]**), length (**[0;+∞]**), distinct

List of supported async job types. If the list is empty assums all job types are acceptable.

- items : **string**, values (**export2draster**, **export2dvecto**, **export3d**)

  `export2draster`: 2D raster image export.`export2dvecto`: 2D vectorial image export.`export3d`: 3D or BOM export.

- *openglprovider* : **string**, default (**system**), values (**none**, **system**, **mesa**)

  Allows to choose which OpenGL implementation should be used. *none* : will not load OpenGL support, making some export jobs unprocessable. *system* : will use OpenGL implementation provided by the OS. *mesa* : will use Mesa3D LLVM software renderer (only available on Windows).

1.4 - ∞CLI configuration file

1.4.1 - Search path

The ∞CLI will search for its configuration file in the following locations:

- `.` in the working directory of the executable
- *%APPDATA%\3djuump-infinite-proxy\conf_4_1.json* (Windows only)
- *%PROGRAMDATA%\3djuump-infinite-proxy\conf_4_1.json* (Windows only)
- */etc/3djuump-infinite-cli/conf_4_1.json* (Linux only)

1.4.2 - Schema of the configuration file

∞Cli configuration file

object

- *$schema* : **string**, length (**[1;64]**), pattern (`^\./.*\.schema\.json$`)
- *http_client_configuration* : HTTP configuration. see Http client global configuration
- *log* : see Log configuration
- *directory_collection* : **object**

  Connection settings per ∞Directory. Object key is the ∞Directory URL with explicit port, eg https://mydirectory:443/prefix/directory.

  - additional properties : **oneOf**
    - **object**
      - *url* : **string**, *REQUIRED*, length (**[0;1024]**), pattern (`^https?:\/\/[^@\/A-Z]+?(:[1-9][0-9]{0,4})(\/.*)?\/directory$`)

        Directory API URL for public usage. The port has to be explicited as this attribute is part of the license.

  - `http_client_configuration` : HTTP configuration to access the ∞Directory. see Http client global configuration
  - `apiKey` : **string**, *REQUIRED*, length (**[1;+∞]**)

    API key secret used to protect API endpoints, this secret will be used as HTTP Basic authentication with 'infinite' login. It will be preferred over OAuth2 method if both are specified.

- **object**
  - `url` : **string**, *REQUIRED*, length (**[0;1024]**), pattern (`^https?:\/\/[^@\/A-Z]+?(:[1-9][0-9]{0,4})(\/.*)?\/directory$`)

    Directory API URL for public usage. The port has to be explicited as this attribute is part of the license.

  - `http_client_configuration` : HTTP configuration to access the ∞Directory. see Http client global configuration
  - `openidconnect` : **object**, *REQUIRED*
    - `common` : Common OpenId Connect settings. see OpenID connect common settings
    - `m2m_auth` : **object**, *REQUIRED*
      - `additional_scopes` : **string**, default (****), length (**[0;1024]**), pattern (`^(()|([\x21\x23-\x5B\x5d-\x7e]+)( [\x21\x23-\x5B\x5d-\x7e]+)*)$`)

        Additional scope string that will be passed to the OpenID server on the token call to obtain and access_token. infinite.* scopes will be added automatically.

      - `client_id` : **string**, *REQUIRED*, length (**[1;+∞]**)

        OpenID application id.

      - `client_secret` : **string**, *REQUIRED*, length (**[1;+∞]**)

        OpenID application secret.

1.5 - Configuration of an HTTP client

1.5.1 - Standard schema

Http client global configuration

object

Main HTTP client configuration, could be override localy on sub configuration depending on the server to contact.

- *verify_ssl_peer* : **boolean**, default (**true**)

  Set this value to false to disable SSL peer verification.

- *client_certificate* : **oneOf**, default (**false**)
  - **object**
    - *crt* : **string**, *REQUIRED*, length (**[1;+∞]**)

      File path to client PEM certificate.

    - *key* : **string**, *REQUIRED*, length (**[1;+∞]**)

      File path to client PEM private key.

    - *password* : **string**

      Private key password if any.

  - **object**
    - *p12* : **string**, *REQUIRED*, length (**[1;+∞]**)

      File path to client P12 certificate.

    - *password* : **string**

      Private key password if any.

  - **boolean**, values (**false**)

    Disable use of certificate.

- *http_proxy* : **oneOf**, default (**false**)
  - **string**, length (**[0;1024]**), pattern (`^https?:\/\/.*$`)

    Enforce use of provided HTTP proxy for HTTP calls.

  - **boolean**, values (**false**)

    Disable use of any HTTP proxy for HTTP calls.

  - **boolean**, values (**true**)

    Enforce use of the automatic HTTP proxy configuration from the system for HTTP calls.

1.5.2 - Schema when overriding

Http client override configuration

object

Allow to override HTTP global configuration.

- *verify_ssl_peer* : **oneOf**, default (**null**)
    - **null**

      Use global configuration.

    - **boolean**

      Set this value to false to disable SSL peer verification.

- *client_certificate* : **oneOf**, default (**null**)
    - **null**

      Use global configuration.

    - **object**
        - *crt* : **string**, *REQUIRED*, length (**[1;+∞]**)

          File path to client PEM certificate.

        - *key* : **string**, *REQUIRED*, length (**[1;+∞]**)

          File path to client PEM private key.

        - *password* : **string**

          Private key password if any.

    - **object**
        - *p12* : **string**, *REQUIRED*, length (**[1;+∞]**)

          File path to client P12 certificate.

        - *password* : **string**

          Private key password if any.

    - **boolean**, values (**false**)

      Disable use of certificate.

- *http_proxy* : **oneOf**, default (**null**)
    - **null**

      Use global configuration.

    - **string**, length (**[0;1024]**), pattern (`^https?:\/\/.*$`)

      Enforce use of provided HTTP proxy for HTTP calls.

- **boolean**, values (**false**)

  Disable use of any HTTP proxy for HTTP calls.

- **boolean**, values (**true**)

  Enforce use of the automatic HTTP proxy configuration from the system for HTTP calls.

1.6 - Logging configuration for services and tools

1.6.1 - Main schema

Log configuration

object

- `debugtypeblacklist` : **array**, default (**[]**), distinct

  RfDebug output that should be omitted.

  - items : **string**, length (**[1;+∞]**)
- `enablediag` : **boolean**, default (**false**)

  DEPRECATED `loglevel` should be used instead. Enables DEBUG log level. This SHOULD NOT BE MAINTAINED IN PRODUCTION as it will log sensitive data and have a negative impact on overall performances.

- `loglevel` : **string**, default (**INFO**), values (**INFO**, **DEBUG**, **TRACE**)

  Specifies log level. `INFO` > `DEBUG` > `TRACE`. A log level lower than `INFO` SHOULD NOT BE MAINTAINED IN PRODUCTION as it will log sensitive data and have a negative impact on overall performances.

- `folder` : **oneOf**, default (****)

  - **string**, length (**[1;+∞]**)

    Change default log location. If relative, will be resolved relative to configuration or job file.

  - **string**, length (**[0;0]**)

    Use default log location.

  - **null**

    Disable file logging.

- `rotatecount` : **integer**, default (**64**), range(**]-1;512]**)

  Number of backup log to keep, if -1 all logs will be kept.

- `maxfilesizemb` : **integer**, default (**64**), range(**]16;1024]**)

Maximum log file size.

- *timerotate* : **string**, default (**weekly**), values (**disabled**, **daily**, **weekly**, **monthly**)

  Enable time base log rotation.

- *loki* : see Loki http log handler configuration
- *log2console* : **boolean**, default (**true**)

  Enable log output to console.

1.6.2 - Schema of the Loki configuration

Loki http log handler configuration

oneOf

- **null**
- **object**

  Configuration for Grafana Loki HTTP push log handler.

  - *posturl* : **string**, *REQUIRED*, length (**[0;1024]**), pattern (*^https?:\/\/[^@\/]+?(\/.*)$*)

    An URL that should point to an endpoint compatible with POST /loki/api/v1/push, body will be gziped JSON, this endpoint is expected to return 200 or 204 on success. URL should not contains credentials.

  - *login* : **['string', 'null']**, default (**null**)

    Loki connection login.

  - *password* : **['string', 'null']**, default (**null**)

    Loki connection password.

  - *http_client_configuration* : HTTP configuration for calls to calls to loki endpoint. see Http client override configuration
  - *label* : **object**, additional properties allowed

    Optional labels that will be added to loki streams.

    - pattern (*^(?!log$).*$*) : **string**, length (**[1;64]**)
  - *max_entry_length* : **integer**, default (**4096**), range(**]0;+∞]**)

    Maximum size in bytes of log message send to loki, if log entry is longer it will be truncated. If zero full message will not be truncated.

1.7 - PostgreSQL configuration

PostgreSQL configuration

object

- *`login`* : **string**, *REQUIRED*, length (**[0;+∞]**)

  PostgreSQL database connection login, could be empty if using SSPI or GSS authentication.

- *`password`* : **string**, *REQUIRED*, length (**[0;+∞]**)

  PostgreSQL database connection password, could be empty if using SSPI or GSS authentication.

- *`database`* : **string**, *REQUIRED*, length (**[1;+∞]**)

  Target database name.

- *`connect_timeout`* : **integer**, default (**15**), range(**]2;+∞]**)

  Maximum wait duration per host while trying to establish a connection. Value is in secondes.

- *`ssl`* : **object**, *REQUIRED*

  - *`enable`* : **boolean**, *REQUIRED*

    Should we use SSL connection.

  - *`verify_ssl_peer`* : **boolean**, default (**true**)

    If disabled, server certificat will not be validated.

  - *`rootCA`* : **string**, default (****)

    File path to rootCA.crt that will be used to verify server certificat, if empty default libpq cert location will be used.

  - *`client_certificate`* : **oneOf**, default (**false**)

    - **object**

      - *`crt`* : **string**, *REQUIRED*, length (**[1;+∞]**)

        File path to client PEM certificate.

      - *`key`* : **string**, *REQUIRED*, length (**[1;+∞]**)

        File path to client PEM private key.

      - *`password`* : **string**

        Private key password if any.

    - **object**

      - *`p12`* : **string**, *REQUIRED*, length (**[1;+∞]**)

File path to client P12 certificate.

- `password` : **string**

  Private key password if any.

- **boolean**, values (**false**)

  Disable use of certificate.

- `hosts` : **array**, *REQUIRED*, length (**[1;8]**)

  List of host, allowing to specify primary and replicat servers. Connection attempt will respect list order, to distribute read-only load on hot standby servers, put them first in the list.

  - items : **object**
    - `host` : **string**, *REQUIRED*, length (**[1;+∞]**)

      Hostname or ip.

    - `port` : **integer**, *REQUIRED*, range(**]1;65535]**)

      Tcp port.

1.8 - OpenID Connect configuration

1.8.1 - Common settings

OpenID connect common settings

object

Common OpenId Connect settings.

- `configuration_endpoint` : **string**, *REQUIRED*, length (**[0;1024]**), pattern (`^https:\/\/([^\/]*?)\/.*$`)

  OpenID Provider configuration URL (https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderConfigurationRequest).

- `http_client_configuration` : HTTP configuration for calls to calls to the OpenID server. see Http client override configuration

1.8.2 - User identification for the ∞Directory

OpenID Connect user identification ∞Directory settings

object

Configure user identification and session access token using OpenId Connect code flow.

- `additional_query_parameters` : **object**

Specifies additional query parameters that should be added to OpenId Connect endpoint calls.

- `authorization_endpoint` : **object**

    Additional query parameters for authorization_endpoint.

    - additional properties : **string**
- `revocation_endpoint` : **object**

    Additional query parameters for revocation_endpoint.

    - additional properties : **string**
- `token_endpoint` : **object**

    Additional query parameters for token_endpoint.

    - pattern (`^(?!scope$).*$`) : **string**

- `additional_scopes` : **object**, *REQUIRED*
    - `primo_token` : **string**, *REQUIRED*, length (**[0;1024]**), pattern (`^(()|([\x21\x23-\x5B\x5d-\x7e]+)( [\x21\x23-\x5B\x5d-\x7e]+)*)$`)

        Additional scope string that will be passed to the OpenID server on the authorize call to obtain first id_token and access_token that will be passed to authentication_webhook.

    - `client_token` : **string**, *REQUIRED*, length (**[0;1024]**), pattern (`^(()|([\x21\x23-\x5B\x5d-\x7e]+)( [\x21\x23-\x5B\x5d-\x7e]+)*)$`)

        Additional scope string that will be passed to the OpenID server to obtain access_token that will be passed to the client.

- `allowed_jwt_alg` : **array**, *REQUIRED*, distinct

    List of algorithm that will be allowed for JWT (id_token and access_token) delivered by the OpenID server.

    - items : **string**, values (**RS256**, **RS384**, **RS512**)
- `authentication_webhook` : **oneOf**, *REQUIRED*
    - **null**

        No authentication webhook.

    - **object**

        Define authentication webhook that will be called on each user identification.

        - `url` : **string**, *REQUIRED*, length (**[0;1024]**), pattern (`^https:\/\/.*$`)

        - `http_client_configuration` : HTTP configuration for calls to calls to the authentication webhook. see Http client override configuration

- `client_id` : **string**, *REQUIRED*, length (**[1;+∞]**)

  OpenID application id.

- `client_secret` : **string**, *REQUIRED*, length (**[1;+∞]**)

  OpenID application secret.

- `hmac_secret` : **oneOf**, *REQUIRED*

    - **null**

      Set to null if HS* sign algorithm are not allowed.

    - **string**, length (**[0;+∞]**)

      OpenID secret for HS* sign algorithm, only supported of id_token. If not null HS256, HS384 and HS512 alg will be accepted.

- `id_token_alias` : **object**, *REQUIRED*

  Allows to copy and optionally remap id_token extra fields (except some sensitive ones) to standard fields to customize user information display. Object keys are extra field name to copy.

    - pattern (`^(?!client_id$|nonce$|aud$|azp$|exp$|iat$|nbf$|acr$|iss$).*$`) : **oneOf**

        - **string**, values (**address**, **email_verified**, **email**, **family_name**, **given_name**, **locale**, **middle_name**, **name**, **nickname**, **phone_number_verified**, **phone_number**, **picture**, **preferred_username**, **profile**, **updated_at**, **zoneinfo**)

          Remap target field name.

        - **null**

          Only copy.

- `token_aud` : **oneOf**, default (**null**)

    - **null**

      Audience (aud) value is assumed to contain client_id.

    - **boolean**, const (**false**)

      Disable aud field validation. Not recommanded but could be usefull when dealing with a weird OpenId Connect server.

    - **string**, length (**[1;+∞]**)

      Value that should be contained in access tokens aud field.

        - **array**, length (**[1;16]**)

          List of potential aud field values. At least one should be equal to access tokens aud field.

            - items : **string**, length (**[1;+∞]**)

              Value that should be contained in access tokens aud field.

- *token_iss* : **oneOf**, default (**null**)
    - **null**

      Issuer (iss) value will be retrieved from configuration endpoint.

    - **boolean**, const (**false**)

      Disable iss field validation. Not recommanded but could be usefull when dealing with a weird OpenId Connect server.

    - **string**, length (**[1;+∞]**)

      Value that should be contained in access tokens iss field.

    - **array**, length (**[1;16]**)

      List of potential iss field values. At least one should be equal to access tokens iss field.

        - items : **string**, length (**[1;+∞]**)

          Value that should be contained in access tokens iss field.

- *token_azp* : **oneOf**, default (**null**)
    - **null**

      Authorized party (azp) value is assumed to contain client_id.

    - **boolean**, const (**false**)

      Disable azp field validation. Not recommanded but could be usefull when dealing with a weird OpenId Connect server.

    - **array**, length (**[1;32]**), distinct

      List of accepted azp values, at least one should be contained in access tokens azp field.

        - items : **string**, length (**[1;+∞]**)
- *use_oidc_access_token* : **boolean**, *REQUIRED*

  Enable use of access_token (OpendID server should also return a refresh_token) delivered by OpenID server to protect ∞Directory and ∞Proxy API calls from client applications (HTTP.session_bearer security scheme). If disabled, tokens delivered by the Directory will be used.

- `use_PKCE` : **boolean**, *REQUIRED*

  Enable use of Proof Key for Code Exchange (rfc7636) (https://tools.ietf.org/html/rfc7636).

- `user_unique_id` : **string**, *REQUIRED*, values (**oidc**, **email**, **azureoid**)

  Define which field of id token will be used as user unique id.
  OpenId Connect : sub of OpenId id
  email : user email /! email should not be reused later for an other user
  azureoid : Azure AD user object id.

1.8.3 - User identification for the ∞Proxy

OpenID Connect user identification ∞Proxy settings

object

- `allowed_jwt_alg` : **array**, *REQUIRED*, distinct

  List of algorithm that will be allowed for JWT (id_token and access_token) delivered by the OpenID server.

  - items : **string**, values (**RS256**, **RS384**, **RS512**)
- `token_aud` : **oneOf**, default (**null**)
  - **null**

    Audience (aud) value is assumed to contain client_id.

  - **boolean**, const (**false**)

    Disable aud field validation. Not recommanded but could be usefull when dealing with a weird OpenId Connect server.

  - **string**, length (**[1;+∞]**)

    Value that should be contained in access tokens aud field.

  - **array**, length (**[1;16]**)

    List of potential aud field values. At least one should be equal to access tokens aud field.

    - items : **string**, length (**[1;+∞]**)

      Value that should be contained in access tokens aud field.
- `token_iss` : **oneOf**, default (**null**)
  - **null**

Issuer (iss) value will be retrieved from configuration endpoint.

- **boolean**, const (**false**)

  Disable iss field validation. Not recommanded but could be usefull when dealing with a weird OpenId Connect server.

- **string**, length (**[1;+∞]**)

  Value that should be contained in access tokens iss field.

- **array**, length (**[1;16]**)

  List of potential iss field values. At least one should be equal to access tokens iss field.

  - items : **string**, length (**[1;+∞]**)

    Value that should be contained in access tokens iss field.

- `token_azp` : **oneOf**, default (**null**)
  - **null**

    Authorized party (azp) value is assumed to contain client_id.

  - **boolean**, const (**false**)

    Disable azp field validation. Not recommanded but could be usefull when dealing with a weird OpenId Connect server.

  - **array**, length (**[1;32]**), distinct

    List of accepted azp values, at least one should be contained in access tokens azp field.

    - items : **string**, length (**[1;+∞]**)
- `use_oidc_access_token` : **boolean**, *REQUIRED*

  Enable use of access_token (OpendID server should also return a refresh_token) delivered by OpenID server to protect ∞Directory and ∞Proxy API calls from client applications (HTTP.session_bearer security scheme). If disabled, tokens delivered by the Directory will be used.

1.8.4 - Machine to Machine identification

OpenID Connect M2M settings

object

Configure OAuth2 machine to machine identification using OpenId Connect client credentials flow. See HTTP.m2m_bearer authentication method. Those settings will be used to acquire a token and to validate received tokens.

- *additional_query_parameters* : **object**

  Specifies additional query parameters that should be added to OpenId Connect endpoint calls.

  - *token_endpoint* : **object**

    Additional query parameters for token_endpoint.

    - pattern (`^(?!scope$).*$`) : **string**
- *additional_scopes* : **string**, default (****), length (**[0;1024]**), pattern (`^(()|([\x21\x23-\x5B\x5d-\x7e]+)( [\x21\x23-\x5B\x5d-\x7e]+)*)$`)

  Additional scope string that will be passed to the OpenID server on the token call to obtain and access_token. infinite.* scopes will be added automatically.

- *allowed_jwt_alg* : **array**, *REQUIRED*, distinct

  List of algorithm that will be allowed for JWT (id_token and access_token) delivered by the OpenID server.

  - items : **string**, values (**RS256**, **RS384**, **RS512**)
- *client_id* : **string**, *REQUIRED*, length (**[1;+∞]**)

  OpenID application id.

- *client_secret* : **string**, *REQUIRED*, length (**[1;+∞]**)

  OpenID application secret.

- *token_aud* : **oneOf**, default (**null**)

  - **null**

    Audience (aud) value is assumed to contain client_id.

  - **boolean**, const (**false**)

    Disable aud field validation. Not recommanded but could be usefull when dealing with a weird OpenId Connect server.

  - **string**, length (**[1;+∞]**)

    Value that should be contained in access tokens aud field.

  - **array**, length (**[1;16]**)

    List of potential aud field values. At least one should be equal to access tokens aud field.

    - items : **string**, length (**[1;+∞]**)

      Value that should be contained in access tokens aud field.

- `token_iss` : **oneOf**, default (**null**)
  - **null**

    Issuer (iss) value will be retrieved from configuration endpoint.
  - **boolean**, const (**false**)

    Disable iss field validation. Not recommanded but could be usefull when dealing with a weird OpenId Connect server.
  - **string**, length (**[1;+∞]**)

    Value that should be contained in access tokens iss field.
  - **array**, length (**[1;16]**)

    List of potential iss field values. At least one should be equal to access tokens iss field.
    - items : **string**, length (**[1;+∞]**)

      Value that should be contained in access tokens iss field.
- `token_azp` : **oneOf**, default (**null**)
  - **null**

    Authorized party (azp) value is assumed to contain client_id.
  - **boolean**, const (**false**)

    Disable azp field validation. Not recommanded but could be usefull when dealing with a weird OpenId Connect server.
  - **array**, length (**[1;32]**), distinct

    List of accepted azp values, at least one should be contained in access tokens azp field.
    - items : **string**, length (**[1;+∞]**)

2 - Security guidelines

3D Juump Infinite relies on several third-party server-side software components that, like any software, are subject to security vulnerabilities. Make sure you apply the latest security patches on the software components used.

Existing CVE (*Common Vulnerabilities and Exposures*) affecting software components used and/or bundled are listed at release time. This listing and impact analysis is available in *manual/cve/3D Juump Infinite known CVE [version].html*. Addionally this listing could be regenerated using provited python script in *manual/cve/resources/extractCVE.py*.

3 - Third-party software licenses

The details of licenses is available on the 3D Juump Infinite Third-party License manual provided with the software.

4 - Range of use

This annex summarizes the range of use of the software.

4.1 - Minimum requirements

4.1.1 - Server requirements

4.1.1.1 - ∞Directory, ∞Proxy and ∞AsyncJobSolver

The ∞Directory and ∞Proxy softwares run on any of the following operating systems with an **IPv6 stack** and x86-64 CPU architecture :

Operating system	System Service	Docker (Debian 13 (trixie) base images)
Microsoft Windows 11	Yes	Partial using Docker Desktop (performance issues with mounted volumes). Not recommanded due to Docker Desktop instabilities
Linux Debian 12 (bookworm)	Yes (deprecated)	Yes using docker-compose
Linux Debian 13 (trixie)	Yes	Yes using docker-compose
Linux Ubuntu 22.04 LTS (jammy)	No	Yes using docker-compose
Linux RedHat 9.4 (plow)	No	Yes using docker-compose

Other operating system might be supported using docker images, however deployment is not supported out of the box and has to be fully handled by the customer.

Minimum hardware requirements are:

- Quad-core processor (support of POPCNT x86 instruction is mandatory, support of CRC32 x86 instruction is recommended)
- For the ∞Directory and ∞Proxy: 8GB of RAM
- For the ∞AsyncJobSolver: 4GB of RAM
- For the ∞AsyncJobSolver: 8GB of disk space per worker for temporary cache
- 2GB disk space for binaries + sufficient disk space for data (depending on your data sources)
- High-speed hard drive disk or solid-state drive highly recommended
- 1Gbit/s network connection between the ∞Directory and ∞Proxy
- For the ∞AsyncJobSolver: on Linux EGL 1.5 is required with at least a surfaceless platform, on Windows an hardware able to run Mesa3D
- File system supporting files of 4GB+ (NTFS, Ext4, ...)

- An OpenGL implementation supporting at least OpenGL 3.0 and following OpenGL extensions (GL_ARB_vertex_array_object, GL_ARB_draw_instanced, GL_ARB_instanced_arrays)

Recommended hardware requirements are:

- Octo-core processor (support of POPCNT x86 instruction is mandatory, support of CRC32 x86 instruction is recommended)
- For the ∞Directory: 12GB of RAM
- For the ∞Proxy: 16GB of RAM
- For the ∞AsyncJobSolver: 2GB of RAM per CPU core
- For the ∞AsyncJobSolver: a dedicated GPU with 4GB of RAM

4.1.1.2 - ∞CLI: PSConverter

To obtain the best performances with the *PSConverter*, it is advised to batch as many jobs as possible per call to the *PSConverter*.

The worker count parameter has to be adjusted depending on the available CPU resources, memory resources and the complexity of the data.

Minimum hardware requirements are:

- Quad-core processor (support of POPCNT x86 instruction is mandatory, support of CRC32 x86 instruction is recommended)
- 1.5GB of available RAM per worker
- High-speed hard drive disk or solid-state drive highly recommended
- 1Gbit/s network connection to the ∞Directory

Recommended hardware requirements are:

- Octo-core processor (support of POPCNT x86 instruction is mandatory, support of CRC32 x86 instruction is recommended)
- 3GB of available RAM per worker
- High-speed hard drive disk or solid-state drive highly recommended

4.1.1.3 - ∞CLI: Generator

Depending on the complexity and volume of the data, the recommended hardware requirements might need to be increased.

The duration of the generation process will be decreased by adding more CPU cores or other resources.

Minimum hardware requirements are:

- Quad-core processor (support of POPCNT x86 instruction is mandatory, support of CRC32 x86 instruction is recommended)
- 8GB of available RAM
- High-speed hard drive disk or solid-state drive highly recommended
- 1Gbit/s network connection to the ∞Directory

Recommended hardware requirements are:

- Octo-core processor (support of POPCNT x86 instruction is mandatory, support of CRC32 x86 instruction is recommended)
- 16GB of available RAM per worker
- High-speed hard drive disk or solid-state drive highly recommended

4.1.2 - Native application requirements

The client application can run on a broad range of CPUs, from a mobile to a high-end x86-64 central processing unit (CPU). The amounts of CPU power and required RAM depend on the complexity of your DMU and the availability of a dedicated graphic processing unit (GPU).

A dedicated graphic card (GPU) is not required but when it is available, it should be installed with an OpenGL 3.1 and above driver. Any NVIDIA GeForce 8 and above, AMD Radeon 4xxx and above or Intel HD4000 and above may fit. Without a dedicated GPU, the 3D Juump client application is able to run with the integrated GPU.

Eventually, you should also reserve enough hard disk space to store your digital mock-up data and workspaces. The disk space required for the installation of the ∞Client application and the Local DMU Manager is only a negligible fraction of the occupied size, respectively 200MB and 800MB.

Minimal requirements of the ∞Client:

- **Windows 11 64-bit version**
- Dual Core CPU x86-64
- 4GB RAM
- 4GB disk space for binaries and caches
- GPU with OpenGL 3.1 and GLSL 140 support with 3GB RAM
- 1280 x 800 pixel display

Recommended requirements of the ∞Client:

- Windows 11 64-bit version
- Quad Core CPU x86-64
- 6GB RAM
- 8GB disk space for binaries and caches (SSD)
- GPU with OpenGL 3.1 and GLSL 140 support with 4GB dedicated RAM
- 1920 x 1080 pixel display

Minimal requirements for Web Applications / Web Browser:

- Dual Core CPU x86-64
- 8GB RAM
- WebGL 2.0 support
- `deflate` and `br` decompression support
- 64 bit browser, Chromium based, Firefox or Safari (tested on latest Chromium)

Recommended requirements for Web Applications / Web Browser:

- Quad Core CPU x86-64
- Dedicated GPU
- 64 bit Chromium based browser

A keyboard and a mouse are mandatory for fully using 3D Juump Infinite. Touchscreen support is limited to navigation.

When using the Local DMU Manager, it is recommended to add **+2 cores** to the CPU and add **+2GB** of RAM, for the minimal and recommended requirements above.

⚠ In order to install the Local DMU Manager on the user host, you must have an **administrator** account. Note the *administrator* account is only required for the installation. Once the application is deployed, the user can have access to their data with their regular account.

4.2 - Supported input formats

3D Juump Infinite is able to process the following formats:

Format	Ext	Version
3D Experience	.3Dxml	All - R4172, 2014x - 2022x
3DM OpenNurbs – Rhino *(Windows only)*	.3dm	All - 6
3DS	.3ds	
ACIS	.sat .sab	All - R27
ASC Medusa 3D	.asc	
CADDS (explicit parts) & CAMU	._pd ._ps	4, 5
CATIA V4	.model .dlv .exp .session	All - 4.xx
CATIA V5	.CATPART .cgr .CATProduct	R10 - R34
CATIA V6	.3Dxml	R210 - R213, 2011x - 2013
CATIA V6 3D EXPERIENCE	.3Dxml	All - R417, 2014x - 2022x
FBX	.fbx	
glTF v2	.gltf .glb	
I-DEAS	.arc .unv .asc	All - NX5
IGES	.igs .iges	5.2, 5.3
Inventor	.ipt .iam	All - 2024
JT-Format (JtOpen)	.jt	7.0 - 10.5
Nastran	.nas	
NX Unigraphics	.prt	11.1 - CR 2312
OBJ	.obj .mtl	
Parasolid XT-Format	.x_t .xmt_txt .x_b	All - 34

PlmXml *(experimental)*	.plmxml	schema v4
CREO ProEngineer	.prt .xpr .asm .xas	13 - Creo 10
CREO ProEngineer Neutral	.neu	13 - Creo 10
ROBCAD	.rf	
Solidworks	.sldprt .sldasm .prt .asm	1999 - 2023
STEP	.stp .step .stpx .stpz .stpxz	AP203, AP214, AP242
Straessle EUKLID	.edx	
STL	.stl	
VDA	.vda	FS 2.0
VRML	.wrl .wrz .vrml	97

4.2.1 - Geometry

3D Juump Infinite only accepts surface information. All other geometric information is ignored (in particular, vector and point data are not supported).

4.2.2 - Metadata

3D Juump Infinite is able to extract product structure, including external references. It also extracts textual, numeric and datum metadata, plus any combination of lists and maps of the above.

4.2.3 - Annotations

3D Juump Infinite is able to retrieve annotation from source files. Textual information should be supported in all cases. The rendering of NOA and PMI is supported to our best effort, there may be limitations in corner cases or small divergences in rendering compared to their original source file.

4.3 - Supported output formats

4.3.1 - Geometry

Supported output formats for geometry export are:

- FBX
- GLTF2
- JT
- OBJ
- STEP[12]+JT
- STEP+WRL
- VRML

[12] STEP AP242 Part 21

- WRL+WRL
- WRZ+WRZ

Exported geometries are tessellated. Annotations are not exported.

4.3.2 - Image

Supported output formats for image export are:

- JPEG
- PNG
- TIFF

Transparency support depends on the output format.

4.3.3 - Metadata

Supported output formats for metadata export are:

- CSV (comma separated)
- CSV (semi-colon separated)
- JSON
- XML

4.3.4 - Presentation

Supported output formats for presentation export are:

- HTML
- HTML with DZSlide embedded viewer
- JSON
- Markdown
- ODP

Note: Markdown and ODP do not support rich-text, thus when slide comments containing rich-text are found, they are exported as raw-text instead.

4.4 - Limits

Data structure limitations are documented inside JSON schema definitions.

4.4.1 - JSON limits

3D Juump Infinite enforce restriction on JSON document content.

- JSON data should be UTF-8 or ASCII encoded.
- ASCII non printable characters (range [0x00-0x1F] except ‘\r’ ‘\n’ ‘\t’) are forbidden.

4.4.2 - Metadata limits

- Maximum length of utf-8 document ids (in byte) : 255
- Maximum number of `nested` sub objects per metadata document: 10000
- Maximum number of `configured` field in index mapping: 8

- Maximum number of bytes for `text` and `keyword` metadata fields: 32767
- Maximum number of clauses per search/filter : 10000
- Maximum number of terms per attribute filter : 65536
- Maximum number of attribute values to allow enumeration : 1024
- Maximum size of attribute value to allow enumeration (in byte) : 100

4.4.3 - Per build limits :

- Minimum number of assemblies[13]: 1
- Minimum number of single parts[14]: 1
- Maximum number of parts[15]: 268435455
- Maximum number of links[16]: 268435455
- Maximum number of instance metadata[17]: 536870911
- Maximum number of ids[18]: 1073741823
- Maximum number of instances (nodes + leaves): 134217727
- Maximum number of geometry instances[19]: 16777215
- Maximum number of distinct materials: 32767
- Maximum number of annotation views: 1073741823
- Maximum number of annotation types: 1024
- Maximum number of annotations visible at the same time: 262143
- Maximum number of configuration per build : 9999
- Maximum scene size : [-2e6;+2e6] around origin along each axis
- Maximum number of distinct tag groups combination : 2048
- Maximum number of tag groups on which a resource (metadata document, instance, geometry, …) could appear : 64
- Maximum number of distinct combination of tag groups in restriction table : 32767
- Maximum number of distinct group of configuration : 1000000
- Maximum number of disting extra tags : 32
- Maximum number of metadata documents per anchor point (part, link, instance) : 4
- Maximum number of attached documents per part : 32
- Maximum number of annotation documents per part : 128

[13] Structure document with children

[14] Structure document with geometric representation

[15] PartMetadata documents

[16] LinkMetadata documents

[17] InstanceMetadata documents

[18] Distinct `id` fields

[19] One source model instantiated at one world position

- Maximum product structure depth : 100
- Maximum number of vertex and index per source model : 16777215

4.5 - Local Web Application limits

Local Web application (backed by the Local DMU Manager) have limitation compared to standard web applications (backed by an ∞Directory server).

- AssetManager is not available (No store/share features).
- Access rights limitation is not available (No data level access control or feature limitations).
- Only one session can be opened (Only one browser tab).
- ∞AsyncJobSolver are not available (No 2D or 3D export services, etc ...)

4.6 - Async Job solver limits

Async Job solvers shoult respect a CPU, RAM and time enveloppe. Those restrictions are set to ensure that a faulty job will not consum too many resources and will harm the backend. Note that solvers are allowed to exceed temporarily the RAM enveloppe.

- Export 2D jobs should end in less than 600 seconds using less than 2560 MB of RAM. They will use between 1 and 2 cpu core
- Export Vectorial jobs should end in less than 600 seconds using less than 2560 MB of RAM. They will use between 1 and 2 cpu core
- Export 3D jobs should end in less than 600 seconds using less than 2048 MB of RAM. They will use between 1 and 1 cpu core
- Export result file size should be smaller than ${SV_LIMITS_ASYNCJOB_RESULT_SIZE_LIMIT_MB} MB.

5 - Export control classification

The Software, which integrates dual use information security items of American origin (ECCN 5D992.c <10%), is subject to the US Export Administrative Regulations (EAR) 15 C.F.R. part 730 et seq. for the country Group E:1 and E:2 which are, at the date of the License Terms and Conditions: Iran, North Korea, Sudan, Syria and Cuba. In particular, the User shall not use, export or re-export the Software in those countries and with end users or for end uses in breach of the US export control regulations.

The Software has been the subject of a declaration of operations relating to a means of cryptology to the ANSSI (Declaration N ° 17070363). However, the Software does not come under Regulation (EC) N ° 428/2009 of May 5, 2009, setting up a Community regime for the control of exports, transfer, brokering and transit of dual-use items, as confirmed by the Direction Générales des Entreprises / Services des Biens à Double-Usage Goods in his mail N ° FR 80404.