yuuvis® Ultimate: a brief developer overview

Introduction

yuuvis® Ultimate lets you build cloud-based information and document management solutions, content-driven applications, and/or incorporate content management features into your applications. It is an API product which enables you to:

- store, tag, retrieve and manage any type of binary and/or text documents,
- build intelligent information management products by defining object types and attributes with custom schema
- search billions of documents using combined metadata and full text search with easy-to-use SQL-based query language
- set retention times and ensure that documents are maintained in the archive,
- search and retrieve the audit trail of a document
- convert over 60 file types, facilitating document visualization in your application.

In this document, a brief developer overview / reference will be presented. You can register and obtain a free API key at yuuvis.io. More comprehensive documentation with numerous examples is also available there.

yuuvis® Ultimate is certified as a cloud information management system, conforming to strict German and international norms and standards.

TABLE OF CONTENTS

Importing documents

To import and store documents, you must define their object type and metadata. Binary file is optional, unless specified as mandatory in the schema. A document can be imported in a single POST request with multipart body, or in two POST requests, first uploading metadata and afterwards the binary file.

If you would like to import and store multiple documents (objects) at the same time, you can use the same endpoint but provide a list of several metadata records instead of one as well as uniquely referenced content files.

It is also possible to work with compound documents. Compound documents are, in principle, combinations of the binary coding of several individual documents (e.g., a PDF stream), which can be translated back into the individual documents by means of the intervals (ranges) in which the binary content of individual documents can be found. Each interval will be treated as a separate logical object in the system, containing its own metadata. Effectively, you can store numerous documents in one binary stream and define offsets with individual metadata. This can be very useful when working with stream-based content (audio, video or large PDF archives). For each imported document, a content digest (SHA256) is generated and persisted automatically. You can validate the content of each stored document by providing its identification. That way you can ensure document integrity.

During import, content analysis will be performed synchronously and full text content will be automatically indexed and searchable. The list of supported formats for content analysis is available at yuuvis.io.

Retrieving and searching documents

You can easily retrieve any document from the repository by providing its unique identifier. You can thus retrieve document metadata or its content file. It is also possible to retrieve any specific version of any document by providing the version number (see also Updating documents, below).

If you do not know the id of the object you wish to retrieve, you need to perform an object search. yuuvis® Ultimate provides a search endpoint that can process search queries written in our query language. It is based on the Content Management Interoperability Services (CMIS) standard and consists of a subset of SQL-92 grammar augmented by a syntax for requesting full text. This query language effectively provides a relational view of the (otherwise unstructured) data. It is possible to perform simple SELECT queries on attributes, queries with aliases, use conditions, query full text, perform sorting and aggregations. The query language is under constant development, so please visit yuuvis.io for the current set of search features. The

search query is simply sent to the search endpoint in JSON format in the body of a POST request. The response of a search query is a list of documents matching the search criteria.

Retrieving document history

During document lifecycle, different actions may be performed on it, such as import, retrieve, update, and delete. For those actions there are the following history entries: creating metadata, creating binary content, deleting a document, updating metadata, updating content, updating both metadata and content, retrieving content and/or metadata, retrieving text, PDF or thumbnail rendition.

Each action will be recorded as single history entry for a stored document (object) in an audit log, which cannot be changed or otherwise manipulated. You can use the history of a stored document (object) to trace what actions happened from the creation of the document in the system. A history entry is structured like a map, i.e., the entry consists of key-value pairs, whereby a value can also be a key-value pair. Each history entry has additional properties such as timestamp, trace id or version number, which help in reconstructing document history more accurately.

Updating documents

To update the metadata of a stored document (object), you send a POST request with the new metadata values in its body and provide object id. The response returns the full, modified metadata with a new version number. Most of the system properties cannot be updated or are updated automatically. If the new metadata contains these properties, their values must match the values of the current version. Hence, you can retrieve the current metadata using the GET document metadata by ID endpoint first, to use the result. Edit some properties and post it again using this endpoint. A new endpoint has just been developed, which provides the possibility to update single properties without the need to do this, and it will be released soon.

The content of a document can be updated analogously, by sending a POST request with the new content file in its body and providing a document id.

Each time you update either the content file or the metadata of a stored document (object), a new version is created. The endpoints for fetching metadata or content files always return the current version. You can get metadata or content of any previous version by providing a version number in a GET request. Version numbering starts with one and is incremented with each update.

Deleting documents

A document can be deleted from the system by providing document id in a DELETE request to

the adequate endpoint. This request will delete the metadata record from the registry and content file from the repository (if present). All previous versions will be automatically deleted, too.

It is also possible to remove any version of a document, by providing version number in the DELETE request. The used version number must exist for the document (object). Note that if the version number indicates the current version of the document, the entire document - including all its versions - will be deleted.

It is possible to define link documents: they contain only metadata, but their content stream references an already existing object. if you delete such a link document (object) the corresponding content file is also deleted from the system and is also no longer available for the originally imported document, too.

Rendition and extraction services

yuuvis® Ultimate offers methods to generate renditions (previews) from uploaded documents. With the methods available, you can render native files (content files) into PDFs, PNG thumbnails, plain text, or receive a summary of an object's rendition information by performing a GET request and passing object id. You can then use these renditions to build previews or visualization in your client application or for any other purpose you require.

Furthermore, an extraction service is available, which automatically extracts metadata such as XMP or EXIF as well as other data from media and office files and maps them in the object attributes. The list of supported types and properties is available at yuuvis.io.

We are currently working on machine learning approaches for enriching metadata even further. Those will be available as endpoints in 2020.

Schema

yuuvis® Ultimate supports intelligent document management by providing a schema for all documents. Schema consists of object types and attributes. Attributes are assigned to object types and can be reused or inherited. An object type thus represents document class (such as invoice, person or contract) and attributes represent metadata (such as invoice date, birthday or contract expiration date).

The system supports a global schema and defines commonly used object types and attributes. Each API key holder can define local schema with custom attributes and object types, which will then be merged with global schema, but visible for the current API key (tenant) only.

You will use schema when creating and updating documents, as well as when searching for documents. You can retrieve local and global schemas with a simple GET request and you can export it to a file, too. A schema validating endpoints is also available.

Valid schema consists of properties (attributes), object types and system properties. An object must have one and only one object type. The object type classifies the object and defines the properties that the object must have or is allowed to have (properties may be optional). The schema defines a set of object types and a set of properties. There are furthermore some metadata like version or last modification date, whose values are provided by the system.

We support attributes of types string, integer, decimal, datetime, date, table, Boolean, enumeration. Attribute types are being constantly extended, please visit yuuvis.io for current information.

We support the following object types: documents, folders and secondary object types. Document object types are the elementary object types and you will use them most commonly to store your documents.

Folder objects cannot have content. Document objects can be assigned to a folder object. Then the folder object is the parent of the document objects and the document objects are the children of the folder object. This relation can be used when searching for documents within a folder.

Secondary object types are abstract. This means that secondary object types cannot be instantiated. They allow you to design a more complex schema. In a way the concept of secondary object types is similar to the concept of inheritance. Secondary object types can be used to group properties and then assign these property groups to other object types. Like other object types, a secondary object type can have references to properties. Document object types can in turn reference secondary object types, which gives them additional properties.

Retention management

yuuvis® Ultimate provides retention management. You can use system secondary type objects to inherit retention metadata for your object types. You can use those attributes to manage retention times. Those define the minimum amount of time an object cannot be deleted from the system using yuuvis® Ultimate API. You can use this capability to implement archiving solutions requiring long-term archiving.

Summary

yuuvis® Ultimate is a cloud native product, providing an API for efficient and intelligent document management. To get started, visit yuuvis.io and register to receive a free API key.

There are several product plans to choose from, from development to much more powerful production-grade plans. Comprehensive developer documentation is also available at yuuvis.io and additional coding examples are provided on github (https://github.com/yuuvis).

yuuvis® Ultimate is under constant development and is continuously delivered, so please follow online for the most up-to-date information.