

# Searchable multi-dimensional Data Lakes supporting Cognitive Film Production & Distribution for the Promotion of the European Cultural Heritage

Grant Agreement No 101095303

## DELIVERABLE 3.1:

**Exploration Data Lakes & Ontologies**

Work Package: 3

## LEAD BENEFICIARY:

**UNIVERSITAT POLITECNICA DE VALENCIA (UPV)**

Delivery Date: 29.11.2024



## Document Sheet

<b>Project acronym</b>	SCENE
<b>Project full title</b>	Searchable multi-dimensional Data Lakes supporting Cognitive Film Production & Distribution for the Promotion of the European Cultural HeritagE
<b>Programme</b>	Horizon Europe
<b>Topic</b>	HORIZON-CL2-2022-HERITAGE-01-06
<b>Type of Action</b>	HORIZON-Research and Innovation Actions
<b>Grant Agreement</b>	101095303
<b>Start day</b>	1 February 2023
<b>Duration</b>	36 months

### LEGAL NOTICE

This project has received funding from the European Union Horizon Research and Innovation programme under grant agreement No 101095303. Neither the European Commission nor any person acting on behalf of the Commission is responsible for the use, which might be made, of the following information. The views expressed in this report are those of the authors and do not necessarily reflect those of the European Commission.

©SCENE Consortium, 2024

Reproduction is authorised provided the source is acknowledged.



## Document Information

<b>Deliverable number</b>	D3.1
<b>Deliverable name</b>	Exploration Data Lakes & Ontologies
<b>Lead beneficiary</b>	UPV
<b>WP</b>	3
<b>Related task(s)</b>	T3.1
<b>Type</b>	OTHER
<b>Reviewers (Organisation)</b>	FRAUNHOFER, DTT
<b>Delivery date</b>	29.11.2024
<b>Main author(s)</b>	UPV
<b>Contributor(s)</b>	CERTH

## Dissemination level

<b>PU</b>	Public	X
<b>SEN</b>	Sensitive, limited under the conditions of the Grant Agreement	
<b>Classified R-UE/EU-R</b>	EU RESTRICTED under the Commission Decision No2015/444	
<b>Classified C-UE/EU-C</b>	EU CONFIDENTIAL under the Commission Decision No2015/444	
<b>Classified S-UE/EU-S</b>	EU SECRET under the Commission Decision No2015/444	

## Document history

Version	Date	Changes	Reviewer/Contributor
v0.1	20/09/2023	Initial ToC	UPV
v0.15	26/09/2023	Introduction Chapter Initial contribution to data lakes	UPV
v0.20	13/10/2023	Initial contribution to semantics (Annex)	UPV
v0.21	18/10/2023	Small contribution to Section 3	UPV
v0.22	16/01/2024	Section 2 is completed, adding a new subsection (best practices) Added methodology in section 5.1.1	UPV
v0.23	05/02/2024	Small changes in the sections	UPV, CERTH
v0.25	28/02/2024	Sections 2.1.5.2, 2.1.5.3, 2.1.5.4 completed	UPV
v0.3	30/06/2024	Section 4 drafted	UPV
v0.4	13/09/2024	Review of all sections and general update aligned with SCENE architecture (some contributions shifted as Annex)	UPV
v0.5	08/11/2024	The implementation section for the data lake is completed.	UPV
V0.6	09/11/2024	Contribution from CERTH related to SCENE-O	CERTH
V0.7	22/11/2024	1 <sup>st</sup> internal review	FRAUNHOFER, DTT
V0.8	25/11/2024	Consolidated version addressing internal comments	UPV, CERTH
v1.0	29/10/2024	Final version for submission	CERTH



# Table of Contents

Publishable summary .....	12
1. Introduction.....	13
1.1. Context.....	14
1.1.1. Objectives .....	14
1.1.2. Relation with other Deliverables .....	15
1.2. Deliverable Structure .....	16
2. Exploration of Data Lakes and data-related concepts .....	17
2.1. Data terminology and data management evolution.....	17
2.2. General Architecture of a Data Lake.....	19
2.3. Available implementations .....	21
2.3.1. Open-source approaches.....	21
2.3.2. Cloud-based approaches .....	24
2.4. Best practices for data lakes.....	31
2.5. Data Lakes in the film-making industry.....	32
2.5.1. Content perspective (content-type).....	35
2.5.2. Structure perspective (metadata).....	36
2.5.3. Usability perspective (user perspective).....	37
2.6. Examples of film-related Data Lakes .....	38
3. Requirements Evaluation and Proposed Solution.....	43
3.1. General vision.....	43
3.1.1. Ingestion Layer: Incorporating New Data .....	44
3.1.2. Storage Layer: Managing Structured and Unstructured Data .....	46
3.1.3. Discovery Layer: Search and Ontology Interaction.....	46
3.1.4. Analysis Layer: Supporting AI/ML Tools .....	46
3.1.5. Conceptual end-to-end Workflow example: Integrating Data Lakes and Ontologies in SCENE	48
3.2. Component vision and interaction with other tasks.....	50
3.3. Requirements.....	52
3.3.1. Addressing End User Requirements.....	52
3.3.2. Internal requirements.....	52
3.4. Platform Technology Overview: Components and Justifications.....	54
3.4.1. Storage Layer.....	54
3.4.2. Analysis Layer .....	55
3.4.3. Discovery Layer.....	55
3.4.4. Ingestion Layer.....	55



- 3.4.5. *Operational Tools and AI/ML Integration* ..... 56
- 3.5. *Proposed API Overview and Rationale*..... 56
  - 3.5.1. *Data Storage API (MinIO’s S3-Compatible API)*..... 57
  - 3.5.2. *Data Ingestion API (NiFi Integration)* ..... 57
  - 3.5.3. *Operational Tools API (Custom REST API)* ..... 58
  - 3.5.4. *Semantic Mapping and Ontology API*..... 58
  - 3.5.5. *External System Integration API*..... 58
- 3.6. *GDPR Compliance and Data Privacy Considerations* ..... 58
- 3.7. *Potential Challenges and Mitigations*..... 59
- 4. *Exploration of film-related ontologies and Proposed Solution*..... 60
  - 4.1. *Film-related ontologies*..... 60
  - 4.2. *Proposed Ontology: SCENE-O* ..... 65
- 5. *Exploration of ontology design and related tools* ..... 67
  - 5.1. *Ontology and Related Tools Specification* ..... 68
  - 5.2. *Exploration of ontology conversion tools* ..... 70
  - 5.3. *Exploration of ontology alignment tools* ..... 71
  - 5.4. *Exploration of ontology design methodology*..... 73
- 6. *Data Lake Implementation* ..... 75
  - 6.1. *Architecture and building blocks*..... 76
    - 6.1.1. *Nginx Component overview (Proxy Layer- user interface)*..... 78
    - 6.1.2. *Keycloak Component Overview (Authentication and Access Management)* ..... 81
    - 6.1.3. *Apache NiFi Component Overview (Ingestion Layer)* ..... 82
    - 6.1.4. *Node-RED Component Overview (Ingestion Layer)*..... 84
    - 6.1.5. *Minio Component Overview (Storage Layer)* ..... 86
    - 6.1.6. *Mongo Component Overview (Storage Layer)* ..... 88
    - 6.1.7. *Trino Component Overview (Discovery Layer)*..... 89
    - 6.1.8. *WebVowl Component Overview (Discovery Layer)* ..... 91
    - 6.1.9. *Jupyter Notebook Component Overview (Visualisation and Interaction Layer)* ..... 93
  - 6.2. *APIs, code and online documentation* ..... 94
- 7. *Conclusions*..... 96
- 8. *References* ..... 98
- 9. *Annexes* ..... 101
  - A. *Data Management Tools and Methods*..... 101
    - i. *Tools*..... 101
    - ii. *Methods or Models*..... 102
    - iii. *Data Lakes and its evolution towards data mesh* ..... 102



- B. *Other data architectures*..... 103
  - i. *Data warehouse*..... 103
  - ii. *Data lakehouse (Delta Lake)*..... 104
  - iii. *Data mesh*..... 105
  - iv. *Data fabric* ..... 106
- C. *Introduction to semantic concepts* ..... 107
  - i. *Ontology Overview*..... 107
  - ii. *RDF as the basis for the semantic web*..... 108
  - iii. *Main Building Blocks in Semantic Applications* ..... 109
  - iv. *Linked Data*..... 110
  - v. *Simple Knowledge Organization System (SKOS)*..... 110
  - vi. *Web Ontology Language (OWL)*..... 110
  - vii. *Common Practices for Semantic Modelling*..... 110
  - viii. *Semantic benefits* ..... 111
- D. *Semantics and Ontologies in the Film-making Industry* ..... 111
  - i. *Schema.org* ..... 111
  - ii. *EBUCorePlus*..... 114
  - iii. *Dublin Core* ..... 118
  - iv. *Ontology for Media Creation (OMC) from MovieLabs* ..... 120
  - v. *VideoMetadataHub from IPTC* ..... 122
- E. *Requirements Questionnaire (first iteration)*..... 125
- F. *Requirements Questionnaire (second iteration)*..... 128
  - i. *Data Lakes*..... 129
  - ii. *SCENE-O*..... 131
  - iii. *Interactions between tools* ..... 133



# List of Figures

Figure 1. The general process for building data lakes and ontologies in task T3.1 ..... 13

Figure 2. Relation of D3.1 with other deliverables ..... 16

Figure 3. Data Lake simplified architecture ..... 19

Figure 4. Data Lake 3-layer architecture ..... 20

Figure 5. Data quality in data lakes ..... 21

Figure 6. Google-related products for data lakes. Source [4] ..... 25

Figure 7. Google tools for different types of input data. Source [5] ..... 26

Figure 8. IBM cloud data lake architecture example. Source [6] ..... 27

Figure 9. IBM data integration with other cloud systems. Source [7] ..... 27

Figure 10. AWS cloud data lake architecture example. Source [9] ..... 29

Figure 11. MS Azure cloud data lake architecture example with data management. Source [10] ..... 30

Figure 12. Cloud components offered by GCP, AWS and Azure. Source [12] ..... 31

Figure 13. SCENE High-Level architecture ..... 43

Figure 14. Data lake component diagram (as for D2.6) ..... 44

Figure 15. Interaction between T3.1, T3.2 and T3.3 ..... 50

Figure 16. SCENE ontology displayed in the ontology viewer ..... 65

Figure 17. Methodology for building a data lake ..... 76

Figure 18. Mapping of software tools and architecture components ..... 77

Figure 19. Data Lake web-based Control Panel ..... 79

Figure 20. General data flow for data lakes ..... 102

Figure 21. Evolution from centralised to decentralised data management ..... 103

Figure 22. Data warehouse architecture ..... 104

Figure 23. Delta lake ..... 105

Figure 24. Data mesh architecture ..... 106

Figure 25. Data fabric architecture ..... 107

Figure 26. Ontology spectrum. One view ..... 108

Figure 27. EBUCorePlus Overview ..... 115

Figure 28. EBUCorePlus displayed in the ontology viewer ..... 115

Figure 29. OMC building blocks ..... 120

Figure 30. OMC displayed in the ontology viewer ..... 121

# List of Tables

Table 1. Reference terminology in the data management domain ..... 17

Table 2. Data maturity phases for organisations ..... 19

Table 3. Open-source data lake-related solutions ..... 21

Table 4. Hadoop vs Spark ..... 22

Table 5. Open-source data-lake core solutions ..... 23

Table 6. Comparison between Minio and Ceph ..... 23

Table 7. Open-source data catalogues ..... 24

Table 8. Data lake core products (Google’s cloud approach) ..... 25

Table 9. Data lake core products (IBM’s cloud approach) ..... 26



Table 10. Data lake core products (AWS cloud approach) .....	28
Table 11. Data lake core products (Microsoft cloud approach) .....	29
Table 12. Best practices for data lakes .....	31
Table 13. Data Lake Features .....	34
Table 14. IMDb (Internet Movie Database) summary .....	38
Table 15. TMDb (The Movie Database).....	40
Table 16. EIDR (Entertainment Identifier Registry).....	41
Table 17. Other film-related data lakes .....	42
Table 18. Candidate technologies for data ingestion .....	45
Table 19. Candidate technologies for distributed querying .....	46
Table 20. AI/ML Processing Frameworks .....	47
Table 21. Internal feedback from partners (mostly data lake-related) .....	52
Table 22. Key Ontologies and Semantics in the film-making Industry.....	61
Table 23. Related ontologies linked with the FI industry.....	63
Table 24. Related thesauri and vocabulary from the cultural heritage perspective .....	63
Table 25. Important entities re-used from film-related ontologies and included in SCENE-O.....	66
Table 26. Reference terminology in the data management domain.....	68
Table 27. Summary of ontology conversion tools.....	70
Table 28. Summary of ontology alignment tools .....	71
Table 29. Summary table of the ontology design methodologies available.....	73
Table 30. Nginx's implementation details.....	79
Table 31. Keycloak's implementation details.....	81
Table 32. Apache NiFi's implementation details.....	82
Table 33. Node-RED implementations details .....	84
Table 34. Minio's implementation details .....	86
Table 35. MogoDB's implementation details .....	88
Table 36. Trino's implementation details.....	90
Table 37. WebVowl's implementation details .....	91
Table 38. Jupyter Notebook's implementation details .....	93
Table 39. Data lake vs data warehouse.....	101
Table 40. General use cases for data lakes .....	101
Table 41. Data mesh vs data fabric .....	102
Table 42. Triple examples using Compact URIs (CURIEs).....	108
Table 43. Common W3C namespaces.....	109
Table 44. Potential re-usable entities and properties from schema.org.....	112
Table 45. Potential re-usable entities and properties from EBUCorePlus.....	116
Table 46. Potential re-usable terms from Dublin Core .....	118
Table 47. Potential re-usable entities and properties from OMC.....	121
Table 48. Requirements summary for the data lake (feedback from 4 partners) .....	129
Table 49. User stories summary for the data lake (feedback from 4 partners) .....	130
Table 50. Requirements summary for the SCENE ontology (feedback from 4 partners) .....	131
Table 51. User stories summary for the SCENE-O (feedback from 4 partners) .....	133
Table 52. Mapping between tools and DL/semantics (feedback from 4 partners) .....	133



## Abbreviations

Abbreviations	Full name
ACID	Atomicity, Consistency, Isolation and Durability
AI	Artificial Intelligence
AML	AgreementMarketLight
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
BI	Business Intelligence
CNN	Convolutional Neural Network
CURI	Compact URI
EC	European Commission
EDW	Enterprise Data Warehouse
ETL	Extract Transform Load
FI	Film Industry
GA	Grant Agreement
GCN	Graph Convolutional Networks
GNN	Graph Neural Network
HDFS	Hadoop Distributed File System
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IAM	Identity and Access Management
IPTC	International Press Telecommunications Council
IRI	Internationalized Resource Identifier
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
KG	Knowledge Graph
LDP	Linked Data Platform
LOD	Linked Open Data
MDM	Master Data Management
ML	Machine Learning



MLOps	Machine Learning Operations
OAEI	Ontology Alignment Evaluation Initiative
OO	Object Oriented
ORC	Optimized Row Columnar
OS	Open Source
OWL	Web Ontology Language
PAML	Predictive Analytics and Machine Learning
PURL	Persistent URL
SaaS	Software as a Service
R2RML	RDB to RDF mapping Language (W3C)
RBAC	Role Based Access Control
RDF	Resource Description Framework
RDFa	Resource Description Framework in Attributes
RDFS	Resource Description Framework Schema
REST	Representational State Transfer
SaaS	Software as a Service
SCENE-O	SCENE Ontology
SKOS	Simple Knowledge Organization System
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
UML	Unified Modelling Language
UNSPSC	United Nations Standard Products and Services Code
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VOWL	Visual notation for OWL
W3C	World Wide Web Consortium
WP	Work Package
WSDL	Web Services Description Language
XML	eXtensible Markup Language



## Publishable summary

This deliverable, part of the Horizon Europe SCENE project under Grant Agreement No. 101095303, presents an in-depth analysis of data lakes and ontologies in the filmmaking industry to improve film production and distribution by integrating semantic concepts and tools and advanced data management technologies. Deliverable D3.1 focuses on Task 3.1 (Exploration of Data Lakes and Ontologies), which aims to establish the SCENE Ontology (SCENE-O) and the associated semantic annotation tools to enhance the project's data-driven capabilities supported by a data lake.

The overarching goal of the SCENE project is to create a modular and extendable architecture that supports the complete film production pipeline, from pre-production to distribution. This document addresses three of SCENE's key objectives (KO#1, KO#3, KO#6).

This deliverable is structured into five key sections, exploring data lakes and ontologies, proposed solutions, and expected implementations.

The **introduction** provides the context of the deliverable, detailing the project's goals and the relationship between Task 3.1 and the broader SCENE architecture. The section also discusses the relevance of ontologies and data lakes in the film industry, stressing the importance of SCENE-O as a scalable and agnostic ontology.

The **exploration of Data Lakes section** offers a comprehensive overview of data lakes, including architecture and best practices from the film industry. A comparison of open-source and cloud-based implementations (such as Hadoop, Spark, and MinIO) and their suitability for film data storage and retrieval is included. The chapter also highlights the challenges of managing large, diverse data types in film production and distribution.

The **exploration of Ontologies section** analyses current ontologies used in the film industry, emphasising the need for a semantic layer in the SCENE project to enhance data retrieval and interaction with knowledge graphs. The section provides examples of existing ontologies like schema.org, EBUCorePlus, and the Ontology for Media Creation (OMC), offering insights into how these models could be leveraged in SCENE. The SCENE ontology merges relevant concepts from those ontologies.

To **propose a valid solution**, there is a section evaluating both end-user and internal requirements for implementing a data lake and its integration into the SCENE platform. It outlines a general vision for incorporating data lakes into film production, proposing specific data ingestion, storage, discovery, and analysis solutions.

The **implementation section** details the methodology and tools used to implement the data lake. The section includes interfaces, APIs, and practical examples demonstrating how the SCENE platform will integrate the data lake and ontology systems. The implementation approach includes deploying open-source software and providing online documentation and a code repository (GitHub) that can be updated as the SCENE tools of the SCENE platform are developed and integrated with the data lake.

The **outcomes of Task 3.1** feed directly into subsequent tasks, including T3.2 and T3.3, ensuring the semantic and technical frameworks for data lakes are aligned with the broader goals of the SCENE platform.

The deliverable identifies several **challenges**, such as handling complex data integration across diverse sources and ensuring GDPR compliance in data lakes. These challenges will be mitigated through advanced metadata management, robust APIs, and a focus on user needs and privacy.

# 1. Introduction

This document presents the analysis, conclusions, and results of the SCENE project's aims to **bring semantics and ontologies to the filmmaking industry**. The effort was quite **challenging** as the level of semantics introduced in the current sector is **minimal**, characterised by metadata instead of proper (complete) ontologies. Besides, the film industry covers **various and different phases** (pre-production, production, post-production and distribution). The project placed more effort on candidate approaches that could be more **useful** in the whole pipeline, especially for the data that is treated and stored in data lakes, for which a thorough state-of-the-art is performed to propose **approaches with a semantic layer based on ontologies**.

This deliverable presents the outcomes of task T3.1 'Exploration Data Lakes & Ontologies'. According to the GA [\[1\]](#), this task "will start with a detailed **identification** of the existing **film-related data lakes worldwide**, **aiming to both study them in terms of structure, content & usability** and also to explore the means how to **map** them into **knowledge graphs** (a.k.a. **ontologies**). The task's ultimate goal is to develop the ontology **SCENE-O** and the associated semantic annotation tools and converters for the SCENE **pilots** associated with the data lakes. This task will first analyse the current ontologies available in the filmmaking industry, especially based on the latest **JSONLD** standard issued. Then, different tags will be defined and developed, allowing **dynamic annotation** based on clear rules. Third, the different **syntactic** and **semantic characteristics** of the different film datasets and other data sources present in SCENE pilots will be analysed, exploring significant overlaps in terminology and establishing components of a **common ontology**. This will be extrapolated to achieve the project's as-agnostic and as-scalable-as-possible ontology. Afterwards, advanced semantic techniques will be used for the semantic representation by **clustering knowledge**, ensuring **linked data** compliance. Afterwards, SCENE-O will be developed following the guidelines established during the proposal stage. Then, the technical team will develop the needed bridges for the pilots. Finally, a set of guidelines and complete **documentation** on how to use the ontology will be created. This documentation will be made **public**, and the associated **API** will be created".



Figure 1. The general process for building data lakes and ontologies in task T3.1

The general process is depicted in Figure 1. The **first step** of the work relates to an **in-depth exploration** of current **data lakes, semantics and ontologies** in the film-related world (state of the art). Note that **semantics and ontologies** refer to different aspects; the first refers to including some sort of meaning in a certain system, whereas the other is more formal and powerful. In terms of data format, **JSON-LD** was initially envisioned for its extended usage, especially in the semantic web. Unfortunately, this format is not that common in the film-making industry.

The **second step** refers to the related **requirements** from two different perspectives. On the one hand, **end-user requirements and scenarios** should be considered to correctly understand what needs to be modelled in terms of ontologies, how they search for data (available in the last instance in the data lake) and how they can **benefit from the usage of semantics** (and data lakes). On the other hand, **technical requirements** should also be considered from other SCENE tools (e.g., the Media Asset Manager) to identify the different ways data should be treated and **tagged** (with **metadata**).



The **third step** refers to the **decision point** before implementing (ontologies) or deploying (data lakes). This phase is essential and critical as it analyses (i) **what is available** from the first phase and (ii) **what is required** from users and other tools of the SCENE overall architecture.

The **fourth step** relates to properly implementing data lakes and ontologies after deciding. The **data lake** will provide access to and manage data. In contrast, the **ontology** serves as the basis of a semantic layer on top of it, including needed **annotation tools** and **converters**.

The final step intends to provide helpful documentation with examples and an associated API that facilitates the search from a semantic context. The documentation provided in this deliverable is expected to be ported to an **online** repository and updated easily throughout the project's lifetime.

## 1.1. Context

### 1.1.1. Objectives

This deliverable is related to the following three SCENE objectives (out of eight) as explained below:

- **KO#1:** ...to provide an **AI-powered** solution based on a modular & extendable architecture to facilitate **end-to-end the complete film production pipeline**, spanning **from the pre-production to the distribution phase**, via a set of efficiently **interconnected & self-sustained cognitive systems**, that allow among others for the insightful selection of (i) multimedia material from existing **data lakes**, (ii) appropriate site for on-location filming, (iii) distribution channel, timing & way, (iv) target audience, etc.

Data lakes allow storing different types of objects, regardless of their size and format, even of their native structure, facilitating the ingestion of large amounts of data in this project, typically multimedia (binary) objects.

Using semantics and/or ontologies in SCENE can also play a role in the interconnected cognitive system, increasing its interoperability.

- **KO#3:** ... to **expand** the existing film-making related **data-lakes** into further **interdisciplinary but complementary dimensions** (e.g., location- & regional-aware, quality-related KPIs, etc.), and to **facilitate the semantic search** via the utilisation of AI-based technologies for **descriptive, self-emerging, self-updating, self-adapting & automatically integrated knowledge graphs/ontologies**, that respect both the existing standards of the industry and the interaction modus of Operandi of the relevant end-users

Data lakes are extended/expanded with a semantic layer supported by the SCENE Ontology, increasing how data can be searched and retrieved using knowledge graphs. Furthermore, the SCENE-O extends it to support additional features from task T3.2.

The SCENE ontology will be appropriately documented with examples, including access to code and online documentation, so that knowledge transfer expands internally to other tasks and externally to the overall community. SCENE-O is conceived to be re-used and expanded by the filmmaking industry in the future.

- **KO#6:** ... to **improve the accessibility** of the tangible & intangible Cultural Heritage of Europe and **eternally preserve** (the significance of) historical sites monuments, sites, customs, traditions & values for film-makers (while indirectly for the global audience), through a **unique & consolidating search engine** based on state-of-the-art extendable & easily-searchable knowledge graphs, that will efficiently allow for **the utilisation or even development of original SCENES with regional material even remotely** (i.e., not filming in-situ).



Provisioning a semantic search engine based on the SCENE-O allows one to perform better searches and retrieve relevant results; furthermore, it also allows an improved understanding of the available content if it has been adequately tagged at early stages. The tagging and inclusion of metadata are also supported by semantic tools developed within task T3.1 as part of the semantic layer.

The SCENE ontology will be developed after analysing the user needs from WP2 and in the different pilots, allowing them to co-create the relevant vocabulary that will be part of this ontology. This supports the preservation of Europe's cultural heritage.

### 1.1.2. Relation with other Deliverables

This report, Deliverable D3.1, is an integral part of Work Package 3 (WP3): 3D Reconstruction, Data Lakes & Media Asset Manager (MAM), & IPR Preservation (blockchain), corresponding precisely to task T3.1: Exploration of Data Lakes & Ontologies, spanning from M4 to M22.

#### *Interrelation with Other Tasks and Work Packages:*

The results presented in D3.1 are primarily utilised in subsequent tasks within WP3 to establish a fully operational cognitive search engine:

- T3.2: Ontology formulation mechanisms for raw data & ontology integration (M6-M24)
- T3.3: Media Asset Manager (M7-M24)

Furthermore, D3.1 plays a critical role in tasks related to other Work Packages, significantly contributing to the architecture specification, pilot deployment, and exploitation strategy:

- WP2: Particularly T2.4 and T2.5
- WP5: Task T5.1
- WP6: Task T6.2

#### *Concurrent and Contributing Deliverables:*

Deliverable D3.1 has been prepared concurrently with other deliverables within WP3, which share similar deadlines. This parallel development has enabled mutual input and enhancement:

- D3.2: Ontology formulation from raw data & ontology integration (M24)
- D3.3: Media Asset Manager (M24)
- D2.7: SCENE Reference Architecture.R2 (M24)

#### *Key input deliverables for D3.1 include:*

- D2.5: State-of-the-art analysis and specifications of SCENE solutions (M18)

The outcomes of D3.1 also serve as a relevant input for forthcoming deliverables in:

- WP5: D5.1 and D5.7, related to platform integration.
- WP6: D6.8, focusing on business models and exploitation plans.

The Figure below (Figure 2) shows the relation between D3.1 and other deliverables in a schematic view.

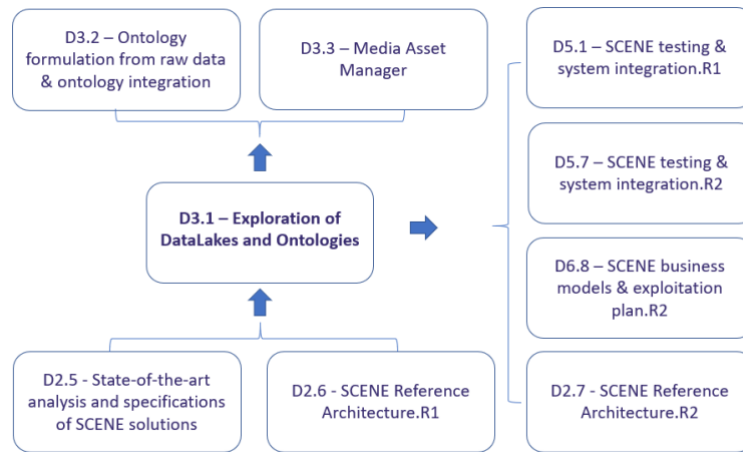


Figure 2. Relation of D3.1 with other deliverables

## 1.2. Deliverable Structure

The dissemination level of this report is public. It is intended explicitly for technical partners working on WP3 and WP4, as well as WP5 (pilot partners to tag their datasets properly). The structure of the document is as follows:

- **Section 1 (Introduction):** It provides an overview of the deliverable's objectives, its relevance to the SCENE project's goals, and its relation to other project deliverables. It establishes the context for exploring data lakes and semantic technologies in the film industry.
- **Section 2 (Exploration of Data Lakes and Data-Related Concepts):** It reviews the theoretical and practical aspects of data lakes, including their evolution, architecture, and the state of the art. This section compares open-source and cloud-based implementations and discusses best practices for designing and managing data lakes, particularly emphasising their application in the film industry.
- **Section 3 (Requirements Evaluation and Proposed Solution):** It summarises the end-user and technical requirements gathered from WP2, WP3, and WP4, aligning them with the SCENE architecture. Proposes an integrated approach for implementing SCENE-O (SCENE Ontology) and its corresponding data lake infrastructure.
- **Section 4 (Exploration of Film-Related Ontologies and Proposed Solution):** It examines existing ontologies within the film industry, identifies gaps, and proposes the SCENE-O ontology as a scalable and interoperable semantic layer to enhance data retrieval and interaction with knowledge graphs.
- **Section 5 (Ontology Design and Related Tools):** It details the methodology for designing the SCENE ontology, exploring tools for ontology conversion, alignment, and integration. It also highlights best practices for ensuring robust and efficient ontology development.
- **Section 6 (Data Lake Implementation):** It outlines the architecture and technical components for implementing the data lake, including tools, APIs, and practical guidelines for deployment. Examples and use cases are provided to demonstrate integration with SCENE-O and the broader SCENE platform.
- **Section 7 (Conclusions):** It summarises the deliverable's findings, emphasising their contribution to the SCENE project's objectives. This section also highlights the challenges encountered and strategies for future enhancements.
- **Section 8 (References):** It includes all cited sources and materials used throughout the deliverable.
- **Section 9 (Annexes):** It provides supplementary information, including detailed descriptions of data management tools, additional ontologies, and extended technical documentation relevant to data lakes and semantic technologies.



## 2. Exploration of Data Lakes and data-related concepts

### 2.1. Data terminology and data management evolution

A common challenge in the rapidly evolving data management landscape is the proliferation of diverse and sometimes **overlapping terminology**. As organisations strive to harness the **power of data** for informed decision-making, they encounter a large set of terms like **data lake**, **data warehouse**, **data mart**, **data fabric**, **data lake house**, and many more. These terms often represent distinct concepts, but their boundaries can blur, causing **confusion** and miscommunication among data professionals and stakeholders. Hence, a comprehensive reference terminology (from the beginning of this document) seems sensible for defining these concepts and establishing clear boundaries and relationships among them. A summary reference table is provided in Table 1.

Table 1. Reference terminology in the data management domain

Concept	Description
Data lake	<ul style="list-style-type: none"> <li>Central and reliable repository to store and process large amounts of data, which can be structured (relational databases), semi-structured (CSV, JSON), unstructured (documents, e-mails) or <b>binary</b> (images, audio, video). It can store data in its <b>native format</b> and process multiple data types.</li> <li>Can be deployed on-premises or in the cloud (e.g., Amazon, Microsoft, Google)</li> <li>Intended for <b>general analysis</b> and <b>reports</b> from raw/massive data (<b>data scientists</b> using AI techniques and languages such as SQL, Python, R)</li> <li>Simple architecture (James Dixon): Hadoop File System (<b>HDFS</b>) with lots of folders/files</li> <li>The media and entertainment sector typically uses data lakes to improve its recommendation system</li> <li><b>Pros:</b> High availability, fast access and processing, relatively cheap (cost/size)</li> <li><b>Cons:</b> Limited quality and data governance</li> </ul>
Data warehouse	<ul style="list-style-type: none"> <li>Data repository for <b>filtered</b> (processed) and <b>structured</b> data with a <b>specific purpose</b></li> <li>Typical target users are <b>business-oriented people (decision-making)</b>.</li> <li>Serves as a <b>single source of truth</b> for a company across multiple knowledge domains</li> <li>Source data (transactional systems, relational DBs) are converted from raw to high-quality data via <b>ETL</b> (Extract, Transform and Load) tools</li> <li>Changes are <b>less flexible</b> and <b>less typical</b> than for data lakes</li> <li><b>Pros:</b> data already processed, saving disk size, data useful for non-technical, high-performance for queries, governance model more robust (compared to data lake)</li> <li><b>Cons:</b> cost (compared to data lake), none to very limited support to unstructured data, delay until ‘fresh’ processed data arrives (limitation for RT)</li> </ul>
Data mart	<ul style="list-style-type: none"> <li><b>Subset</b> of Data warehouse more specific to a <b>particular domain</b> (e.g., finance)</li> </ul>
Data Lakehouse	<ul style="list-style-type: none"> <li>Combines the best from Data Lakes and Data Warehouses: (i) flexibility and cost-effectiveness from a Data Lake, and (ii) performance and structure of a DW</li> <li>Open data formats: Apache <b>Parquet</b>, Apache <b>Avro</b></li> <li>Open table format for analytic datasets (metadata layer): Apache <b>Iceberg</b>, providing SQL interface</li> <li>Support for different engines (<b>Spark, Trino, Flink, Presto, Hive</b>, etc.)</li> </ul>

<p><b>Data fabric</b></p>	<ul style="list-style-type: none"> <li>• <b>Unified</b> and <b>integrated</b> approach to data management and access across an organisation</li> <li>• It <b>unifies data from various sources</b>, such as databases, data lakes, cloud platforms, and streaming data, into a single, cohesive view</li> <li>• It emphasises <b>data integration</b>, transformation, and harmonisation to ensure data consistency and quality</li> <li>• It includes features for <b>data governance</b>, security, and compliance, ensuring that data is managed in a controlled and secure manner</li> <li>• It often involves the creation of an <b>abstraction layer</b> that hides the underlying complexity of data storage and retrieval, making it easier for users to access and work with data</li> </ul>
<p><b>Data mesh</b></p>	<ul style="list-style-type: none"> <li>• <b>Architectural</b> and <b>organisational</b> approach to data management</li> <li>• <b>Data is treated as a Product</b>, which is broken down into data silos. Six main features: discoverable, addressable, self-describing, trustworthy, secure, interoperable</li> <li>• The responsibility for data is <b>decentralised</b> and is given to different teams or domains, each one in charge of their own (data) products for the <b>end-to-end data lifecycle</b> (collection, storage, quality and access)</li> <li>• It typically includes a <b>Data Catalogue</b> that helps users discover and access data products from various domains</li> <li>• It <b>scales horizontally</b> by adding more domain-specific teams</li> <li>• Requires <b>skilled data professionals</b> and <b>significant investment</b></li> <li>• <b>Four pillars</b>: Domain Oriented decentralisation, Data as a Product, Self-serve data infrastructure as a platform, Federated Computational Governance</li> <li>• <b>No standard definition</b> of a data mesh</li> <li>• <b>It is difficult</b> to see the big picture for <b>combining data</b></li> <li>• Some <b>implementations</b> by companies (Intuit, Adevinta, HelloFresh, etc.)</li> </ul>
<p><b>Database</b></p>	<ul style="list-style-type: none"> <li>• <b>Structured collection</b> of data organised and stored in a computer system</li> <li>• Designed to manage and manipulate large volumes of data efficiently</li> <li>• Ensures <b>data integrity</b> and <b>consistency</b> through internal rules</li> <li>• Provide powerful <b>query languages</b> (e.g., SQL) to retrieve data by applying various operations on the data (filter, sort, join, etc.)</li> <li>• Support for <b>concurrent access</b> and <b>transactional</b> management</li> </ul>

*Note: In alignment with the Grant Agreement, our primary focus is establishing a robust data lake architecture. This approach allows us to efficiently handle large, diverse datasets, supporting SCENE's data management and analytical needs. While the current scope emphasises data lakes, this foundational setup is designed to be adaptable, potentially evolving into a data lakehouse architecture if future requirements appear. However, our immediate efforts remain centred on maximising the operational value of a data lake, as per the project's defined objectives*

It is important to split some terms from the previous table into two main domains:

- **Tools**, which are represented by **data warehouses**, **data lakes**, and **data lakehouses**. A tool is a **narrow** technological aspect, even if it can be broken down into several subcomponents.
- **Models/Methods**, represented by **data fabric** and **data mesh**. A model is a **broader general** concept covering various technological and non-technological aspects.

More detailed information about both (tools and models), features and differences can be found in the annexes (section 6.A).

Finally, besides describing the different technological approaches to managing data, it is also essential to not forget the **meaning of data** within an enterprise and the value it generates. The digital transformation accelerates how data impacts an organisation, as presented in **Table 2**.

Table 2. Data maturity phases for organisations

Phase	Description
1. Reactive	Structured data is mainly managed locally and used reactively
2. Informative	Structured data is managed and analysed centrally to inform the business department
3. Predictive	Captured data comes in significant volumes and allows advanced analytics to make predictions leading to business decisions
4. Transformative	Data helps the company efficiently transform towards its business purpose by handling data anytime, anywhere

## 2.2. General Architecture of a Data Lake

A simplified overview of a data lake is depicted in Figure 3. The **ingestion layer** is responsible for **processing** the different input data sources. This process should be understood in a simple way without any relevant data transformation. The input data are then stored in the **storage layer**, preserving its nature (structured, semi-structured, unstructured or raw data). Two additional layers go together with the storage layer:

- **Governance layer:** It is responsible for implementing and enforcing data policies on the system. It is a sort of control layer able to manage pipelines of data. For data lakes, **governance** and **data quality** are a big challenge: poorly managed data lakes are called **data swamps** when there is a lot of duplicated, inaccurate or incomplete data. Avoiding data swamps (during operation) implies being aware of the relevance of data and metadata for a given organisation.
- **Analysis layer:** This layer provides supporting tools to prepare the data, extract features or train AI models with the available data here.

The analysis layer's output is expected to feed the **application layer**, which can be implemented as a simple dashboard or an app if users intend to consume that data. The application layer can also take part of the output data and feed a data warehouse.

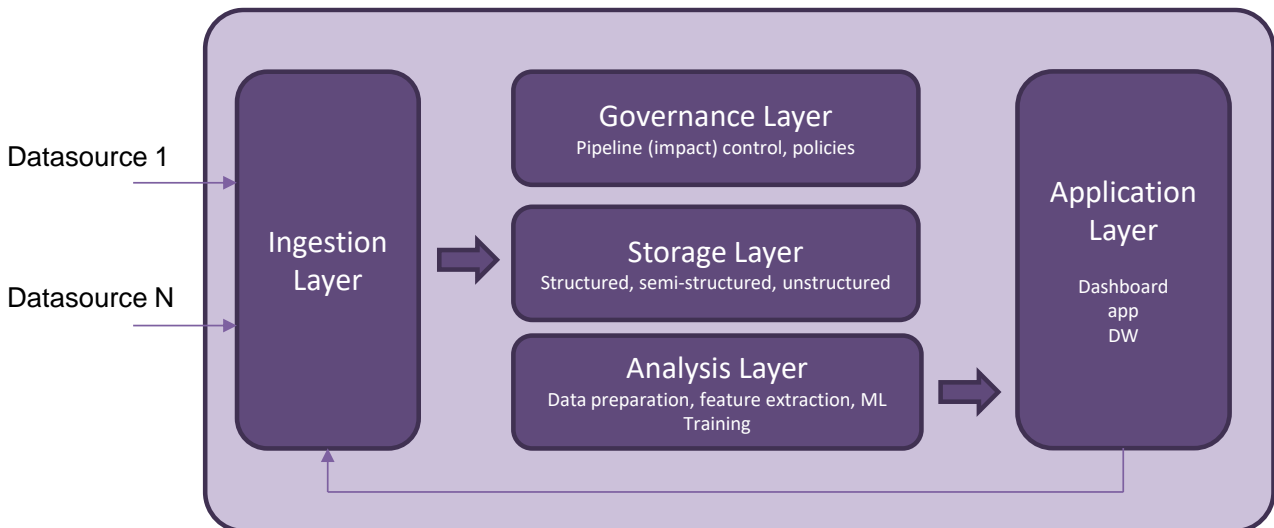


Figure 3. Data Lake simplified architecture

From an **AI perspective**, the four AI models typical phases from [IBM's AI Ladder](#) map to the following layers:

- **Collect:** ingestion layer and storage layer
- **Organize:** analysis layer and storage layer
- **Analyse:** analysis layer

- **Infuse:** application layer

From a **logical perspective**, considering [Hadoop architecture](#), there are **three** (high-level) layers, as depicted in **Figure 4**. Hadoop is a typical reference technology used in data lakes because it supports large volumes of data (Big Data).

The **first layer (data sources)** considers all **input data sources** that should be handled and stored. They are typically divided into (i) traditional data that has typically been managed from the internal operational systems (ERP, CRM and billing) and (ii) emerging data sources that have been added in the last years to enhance the operational picture and better study a particular situation. These data can be either internal or external.

The **second layer (processing & storage)** includes various tools and sublayers. First, the **ingestion sublayer** provides a set of converters for the different input sources without altering the original structure; it will allow further search and discovery once the data is stored in the layer, typically in the form of a **distributed and scalable file system**.

The **third layer (visualisation)** refers to the **visualisation of data** and the **user profiles** accessing such visualisation. In most cases, common users are data scientists, analysts, developers or administrators; however, sometimes also, some specific (business) users might have access to this data, such as a CEO or CTO who needs to have some information as soon as possible.

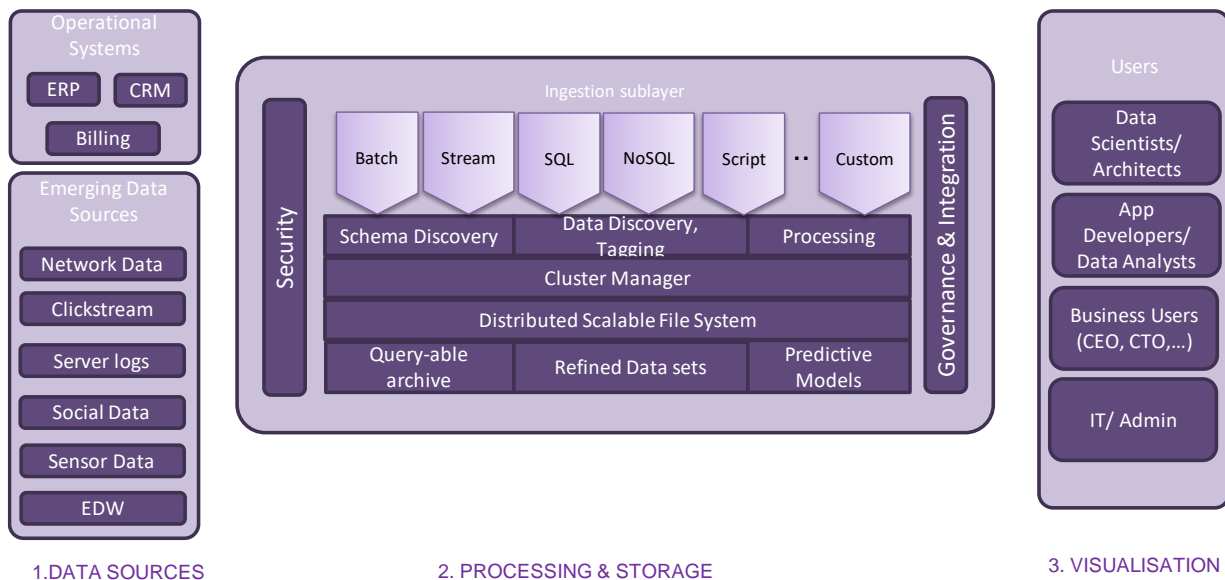


Figure 4. Data Lake 3-layer architecture

From an operational point of view of the quality of the data within the data lake, there are typically three areas or phases depicted in **Figure 5** that have various names in the data engineering terminology:

- The **first** phase refers to the **bronze, landing or raw zone**, where data is stored without modification to preserve the original format. Automated and manual data tagging is also included in this zone. Sometimes, there is a previous supporting transient zone that facilitates ingestion.
- The **second** phase is the **silver, refinery, stage or trusted zone**, where data is filtered and cleaned according to the company's needs (defined by data scientists). The result is data ready for processing. Here, the schemas of the data are defined as needed.
- The **third** phase is the **gold, refined, curated or production zone** with business-level data (e.g., loaded into Hive) that is continuously updated and can be delivered to users and apps. Sometimes, this phase is 'externalised' to a data warehouse.

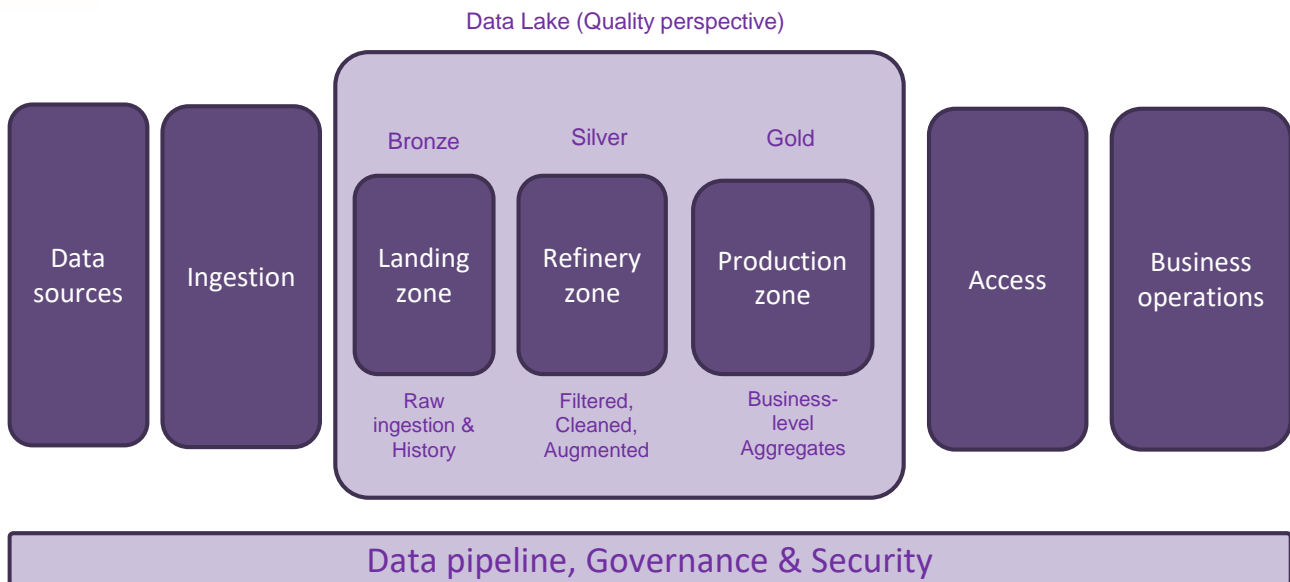


Figure 5. Data quality in data lakes

Other data architectures (data warehouse, delta lake, data mesh and data fabric) can be found in the annexes (section 6.B).

## 2.3. Available implementations

This section explores both open-source and cloud-based implementations of data lakes.

### 2.3.1. Open-source approaches

For open-source approaches, the low-level layer of **data lakes** was typically implemented via Hadoop Distributed File System (**HDFS**) for the early data lakes; modern ones use **Spark** for distributed computing, which is faster than (Hadoop) MapReduce with in-memory processing. On top of the local approaches, some additional (open) layers and/or projects are summarised in the Table below.

Table 3. Open-source data lake-related solutions

Solution	Description
Apache Hadoop	Apache Hadoop is an open-source framework for distributed storage and processing large data sets using a cluster of commodity hardware. It includes HDFS <a href="https://hadoop.apache.org/">https://hadoop.apache.org/</a>
Apache Spark	Apache Spark is a newer open-source framework than Hadoop, and it is used for managing and processing large volumes of data for analytics. It uses in-memory caching and optimised query execution for fast analytic queries against data of any size. <a href="https://spark.apache.org/">https://spark.apache.org/</a>
Apache Hive	Apache Hive is a <b>data warehouse</b> infrastructure built on top of Hadoop. It provides a high-level language (HiveQL) for querying and managing large datasets stored in Hadoop files. <a href="https://hive.apache.org/">https://hive.apache.org/</a> <i>Note: technically speaking, Apache Hive is a data warehouse (not a data lake), but its appearance is sometimes common with data lakes and data lakehouses.</i>
Apache Hudi	Apache Hudi (Hadoop Upserts Deletes and Incrementals) is a data lake management framework for stream processing on top of Apache Hadoop



		compatible storage systems and cloud-based approaches. It supports record-level insert, update, and delete <a href="https://hudi.apache.org/">https://hudi.apache.org/</a>
<b>Apache (Optimized Columnar)</b>	<b>ORC Row</b>	ORC is a self-describing type-aware columnar file format designed for Hadoop workload (commonly used in Hive). RC is significantly faster than RC File or Parquet, and is being used by Facebook and Yahoo. <a href="https://orc.apache.org/">https://orc.apache.org/</a>
<b>Presto/Trino</b>		Presto was created at Facebook as a project and was later divided into PrestoDB and PrestoSQL. The latter one was rebranded as Trino in 2021. PrestSQL is a distributed SQL query engine optimised for ad-hoc analysis. It can query data from various sources, including Hadoop, making it a good choice for querying data in a data lake. <a href="https://prestodb.io/">https://prestodb.io/</a> <a href="https://trino.io/">https://trino.io/</a>
<b>Apache Iceberg</b>		Apache Iceberg is a table format for large, slow-moving tabular data that aims for best practices in systems like Spark, Trino, Flink, Presto, Hive and Impala <a href="https://iceberg.apache.org/">https://iceberg.apache.org/</a>
<b>Alluxio (Community)</b>		Alluxio, formerly known as Tachyon, is an open-source data orchestration layer that sits between storage systems and computational frameworks to manage data efficiently. <a href="https://www.alluxio.io/community/">https://www.alluxio.io/community/</a>
<b>Delta Lake</b>		Delta Lake is an open-source storage layer that brings ACID transactions to Apache Spark and big data workloads <a href="https://delta.io/">https://delta.io/</a>

As for the first two items of the previous table, both Hadoop and Spark allow the processing of big data in different ways; Hadoop is simpler but slower, whereas Spark is faster, more expensive and more complex. Both approaches are compared in **Table 4**. Note, however, that both technologies are not exclusive, and an organisation might use Spark and Hadoop together to meet their data analytics goals.

*Table 4. Hadoop vs Spark*

<b>Apache Hadoop</b>	<b>Aspect</b>	<b>Apache Spark</b>
Data is processed and stored on external storage	<b>Architecture</b>	Data is processed and stored in internal memory
Batch processing (slower)	<b>Performance</b>	Real-time processing (faster)
Cheap	<b>Cost</b>	Expensive
Accessible by adding more nodes	<b>Scalability</b>	Complex
Use of external libraries	<b>Machine Learning</b>	Internal (built-in) libraries
Strong (encryption, access control)	<b>Security</b>	Basic to moderate

Moving beyond the foundational technologies like Hadoop and Spark, several core high-level, open-source data lake implementations cater to more specific needs and provide advanced functionalities. These implementations, including **Minio**, **Ceph**, and **Dremio**, offer enhanced data storage, management, and analytics capabilities, making them vital for modern data infrastructures (see **Table 5**).

Table 5. Open-source data-lake core solutions

Solution	Description
Minio	Open-source object storage solution that is <b>compatible with Amazon S3</b> . It is tailored to store vast amounts of unstructured data such as multimedia files, backups, and container images. Minio stands out due to its <b>straightforward setup, high performance</b> , and suitability for <b>private cloud</b> environments. <a href="https://min.io/">https://min.io/</a>
Ceph	Versatile open-source storage platform that delivers unified <b>object, block, and file storage</b> . Known for its exceptional <b>scalability</b> , Ceph can efficiently manage petabytes of data. Its architecture ensures high performance and <b>reliability</b> , making it an excellent choice for environments requiring robust <b>data redundancy</b> and seamless integration with cloud services. <a href="https://ceph.io/en/">https://ceph.io/en/</a>
Dremio	Open-source data lake engine designed to streamline and <b>accelerate data analytics</b> . It provides a <b>self-service platform</b> for business intelligence and data science, facilitating <b>fast query performance</b> and <b>compatibility</b> with a wide range of <b>data sources</b> , including Hadoop and cloud storage. <a href="https://www.dremio.com/">https://www.dremio.com/</a>

When comparing Minio and Ceph (see Table 6), it is essential to highlight their distinct strengths and suitable use cases. Minio is renowned for its ease of deployment and management, making it an excellent choice for organisations seeking a straightforward, reliable solution for testing and trial purposes. Its S3 compatibility also makes it an attractive option for those looking to integrate with existing S3-compatible applications seamlessly.

On the other hand, Ceph offers a more comprehensive storage solution with support for object, block, and file storage. It excels in environments that require high scalability, robust performance, and advanced data redundancy features. Ceph’s architecture is designed to handle large-scale deployments and provide seamless integration with various cloud platforms and data management tools.

Dremio is not as widely adopted as Minio or Ceph and generally takes a lower priority for organisations focused on core data lake implementations.

Table 6. Comparison between Minio and Ceph

Minio	Aspect	Ceph
Object storage designed to be S3-compatible	<b>Architecture</b>	Unified storage supporting object, block, and file storage
Easily scales horizontally	<b>Scalability</b>	Designed for large-scale deployments, can handle petabytes of data
High performance for object storage operations	<b>Performance</b>	High performance with strong data redundancy and self-healing capabilities
Ideal for storing unstructured data in private cloud setups	<b>Primary Use Case</b>	Suited for extensive storage needs across various data types and cloud environments
Basic redundancy options	<b>Redundancy</b>	Advanced redundancy with self-healing features
Smooth integration with S3-compatible applications.	<b>Integration</b>	Excellent integration with cloud platforms and diverse data management tools.



By focusing on these advanced, high-level solutions, organisations can optimise their data lake architectures to handle complex, large-scale data storage and processing needs more effectively. These tools provide the flexibility, scalability, and performance necessary to support modern data analytics and management strategies.

### 2.3.1.1. Data Governance

The **data governance** layer in data lakes is probably one of the most challenging aspects regarding available open-source implementations. Though there are open-source **data catalogues** for effective data management, they are not natively integrated with any data lakes and are offered as independent modules. We found only one data governance module specifically developed for a data lake called Tombolo [2]; however, it is not generic but created by the company HPCC systems for their own data lake.

A data catalogue (in a general context and not specifically related to data lakes) offers **metadata management, data discovery and governance capabilities**, allowing users to search for data assets and their context and lineage. Some of the most common data catalogues are briefly described in Table 7.

Table 7. Open-source data catalogues

Data Catalogue	Description
Apache Atlas	A scalable and extensible set of core foundational governance services and metadata framework. It supports HDFS, Hive, HBase, and Spark. <i>Pros:</i> robust system with security features, large community <i>Cons:</i> complex set-up, significant resources for scaling <a href="https://atlas.apache.org/#/">https://atlas.apache.org/#/</a>
Datahub	Metadata platform supporting metadata management, data discovery, and data lineage capabilities. <i>Pros:</i> Supports many data sources, databases, file systems, and APIs; friendly UI for data discovery; growing community <i>Cons:</i> limited governance and security features, significant resources for scaling <a href="https://datahub.io/">https://datahub.io/</a>
Metacat	Metacat is a unified metadata exploration API service focused on federated views of metadata systems, allowing arbitrary metadata storage about data sets and metadata discovery <i>Pros:</i> flexible and extensible, growing community <i>Cons:</i> complex set-up, limited support for non-Hadoop platforms <a href="https://github.com/Netflix/metacat">https://github.com/Netflix/metacat</a>
Amundsen	Data discovery and metadata engine intended for analysts and data scientists as well as data and software engineers <i>Pros:</i> robust data lineage and discovery, user-friendly UI for data discovery <i>Cons:</i> limited features, relatively new (not mature enough) <a href="https://www.amundsen.io/">https://www.amundsen.io/</a>
CKAN	world's leading open-source data management system <i>Pros:</i> customisable (plugins), easy to use, robust API, large community <i>Cons:</i> complex to set up, limited functionalities (e.g., analytics) <a href="https://ckan.org/">https://ckan.org/</a>

### 2.3.2. Cloud-based approaches

This section briefly summarises the approaches available to the most common cloud providers related to data lakes (Google, IBM, Amazon and Microsoft).

### 2.3.1.1. Google Cloud

Data lakes in **Google Cloud** are typically offered as a suite with other services (summary table in Table 8).

Table 8. Data lake core products (Google’s cloud approach)

Concept	Description
DataFlow	Unified stream and batch data processing that's serverless, fast, and cost-effective <a href="https://cloud.google.com/dataflow/">https://cloud.google.com/dataflow/</a>
Cloud Data Fusion	Fully managed, cloud-native data integration at any scale to build ETL/ELT pipelines <a href="https://cloud.google.com/data-fusion">https://cloud.google.com/data-fusion</a>
BigQuery	Serverless, highly scalable and cost-effective enterprise data warehouse designed for business agility (built-in ML/AI and BI for insights) <a href="https://cloud.google.com/bigquery/">https://cloud.google.com/bigquery/</a>
Dataproc	Fully managed and highly scalable service for running Apache Hadoop, Apache Spark, Apache Flink, Presto, and 30+ open-source tools and frameworks. <a href="https://cloud.google.com/dataproc/?hl=en">https://cloud.google.com/dataproc/?hl=en</a>
Cloud storage	Globally unified, scalable and durable object reference service for storing unstructured data. <a href="https://cloud.google.com/storage">https://cloud.google.com/storage</a>

It is important to note that cloud providers offer a wide variety of services and tools that may relate to each other, and listing all of them is beyond the scope of this document. For Google, one can find a helpful description through its [Google Cloud documentation](#). From the point of view of data lakes and related tools, the most common services and tools are depicted in Figure 6. The reader can check some data lake design patterns on Google (full stack, unified batch/streaming model, lambda streaming) [3].

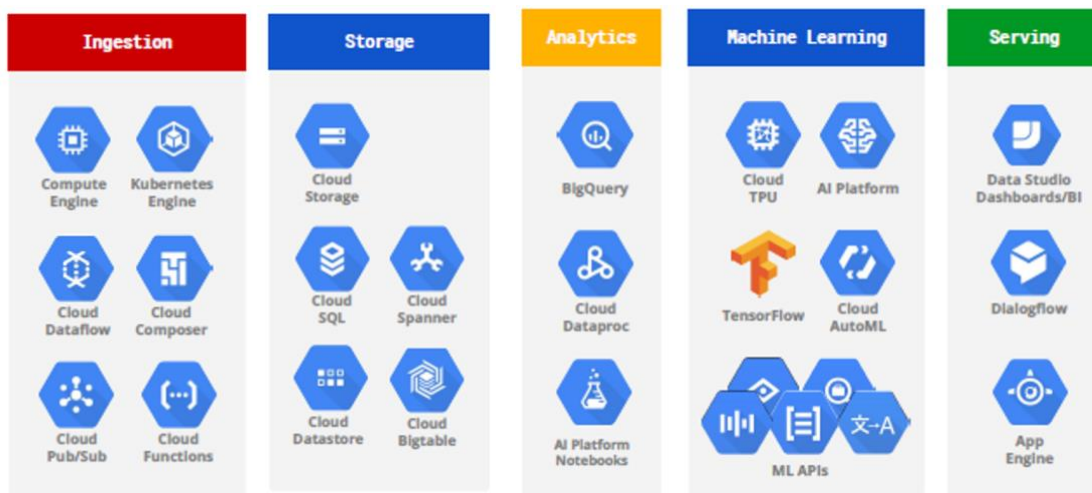


Figure 6. Google-related products for data lakes. Source [4]

Cloud Storage typically supports data storage as a catch-all solution; however, *CloudSQL* and *Cloud spanner* can be used for **relational data**, and *Cloud Firestone* and *Cloud BigTable* for **NoSQL data**. A decision tree matching data type and Google tool is depicted in Figure 7.

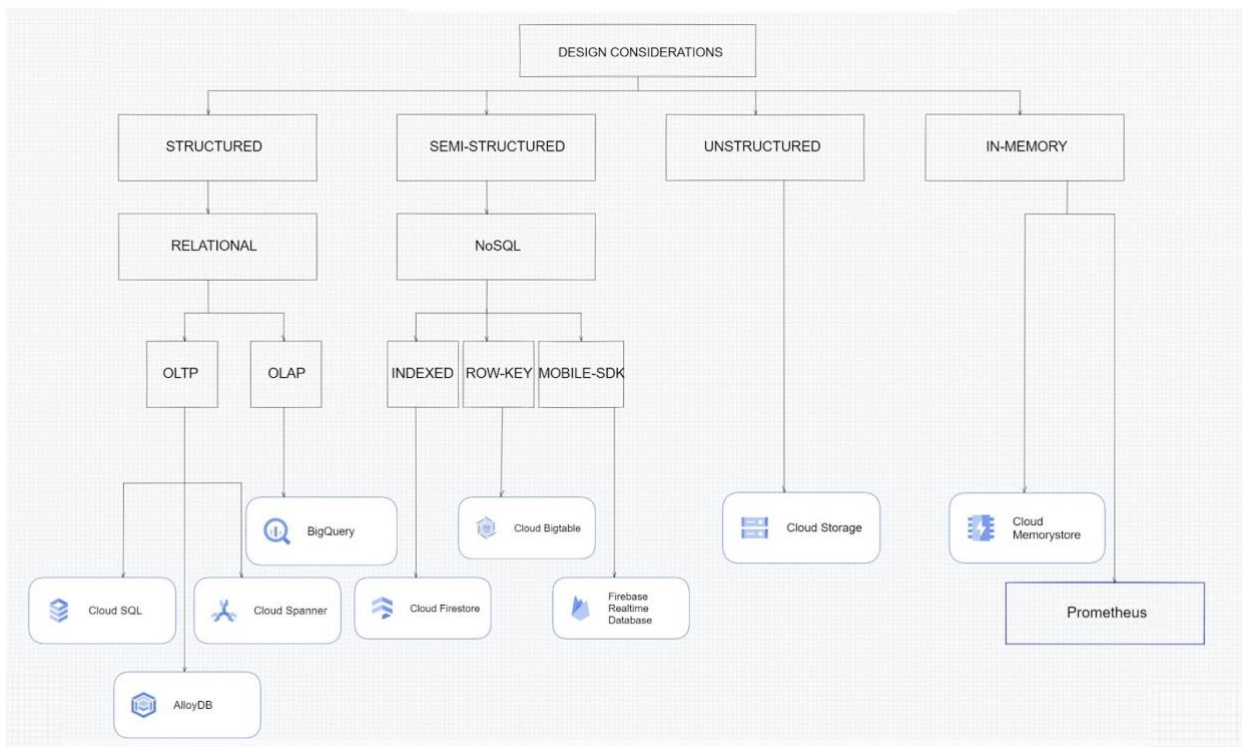


Figure 7. Google tools for different types of input data. Source [5]

### 2.3.1.2. IBM’s Cloud

Data lakes by IBM are offered within the watsonx family of products (AI and data platform, see Table 9)

Table 9. Data lake core products (IBM’s cloud approach)

Concept	Description
<b>Watsonx.ai studio</b>	New foundational models, generative AI and machine learning <a href="https://www.ibm.com/products/watsonx-ai">https://www.ibm.com/products/watsonx-ai</a>
<b>Watsonx.data</b>	Fit-for-purpose data store, built on an open lakehouse architecture <a href="https://www.ibm.com/products/watsonx-data">https://www.ibm.com/products/watsonx-data</a>
<b>Watsonx.governance toolkit</b>	Accelerate AI workflows built with responsibility, transparency and explainability <a href="https://www.ibm.com/products/watsonx-governance">https://www.ibm.com/products/watsonx-governance</a>

The general process for a base solution is depicted in Figure 7, but depending on the application scope (Big Data, Logging, IoT), it may differ slightly. For example, in a Big Data application, all data is ingested into IBM Cloud Object Storage (step 1- grey circle in the middle of Figure 8), a service able to store vast amounts of data. Afterwards, the IBM Analytics Engine deploys Hadoop and Spark to analyse the data (step 2). Finally, the Watson Data Science Workbench is used to analyse the data, build AI models, and provide insights and dashboards (step 3).

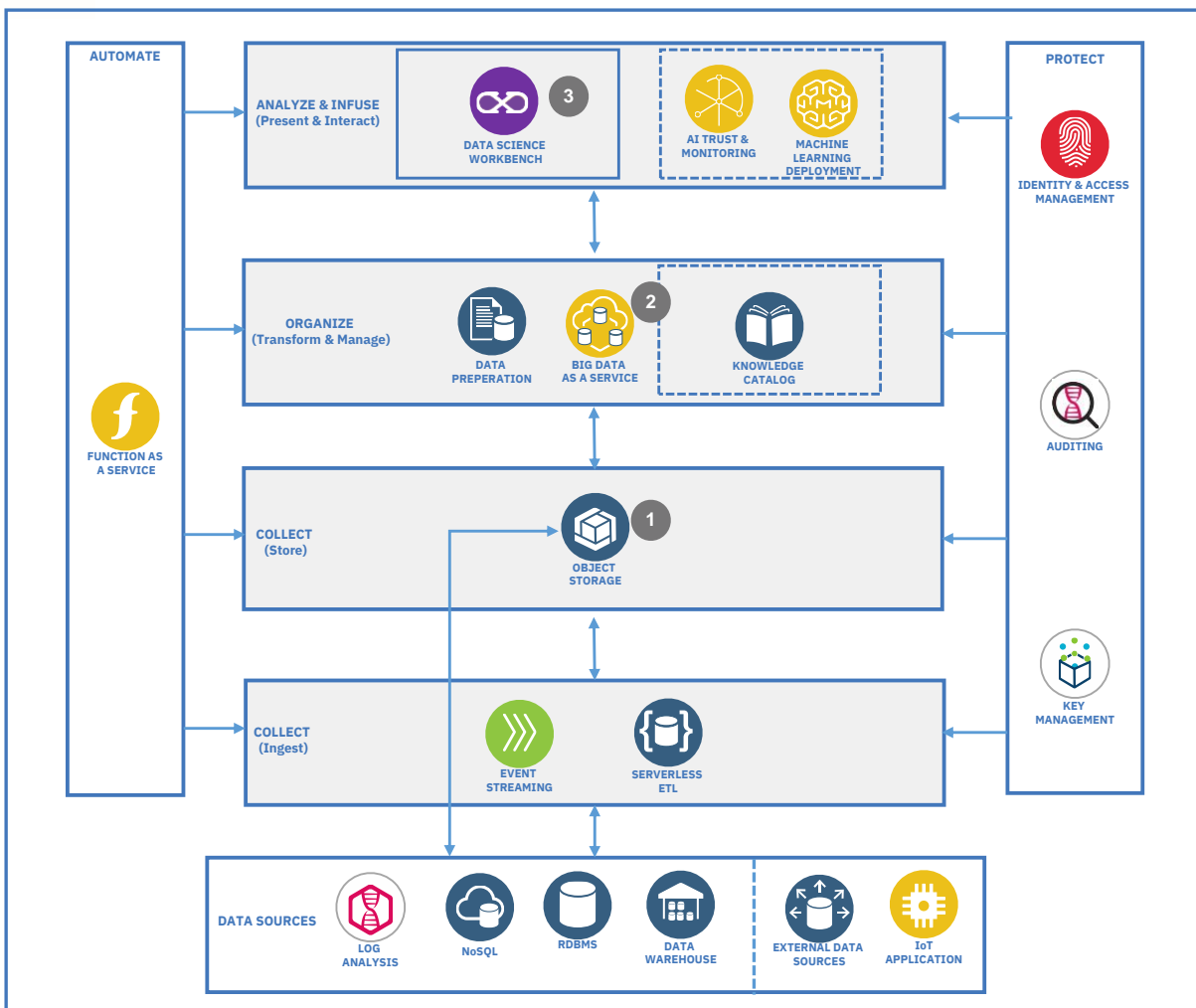


Figure 8. IBM cloud data lake architecture example. Source [6]

The vision of IBM is quite flexible and can coexist and integrate with other platforms, such as AWS, as depicted in Figure 9. It attempts to align and support hybrid environments (public, private and edge environments). IBM promotes using data lakehouses as a combination of advantages from data warehouses and data lakes, even in hybrid-cloud scenarios.

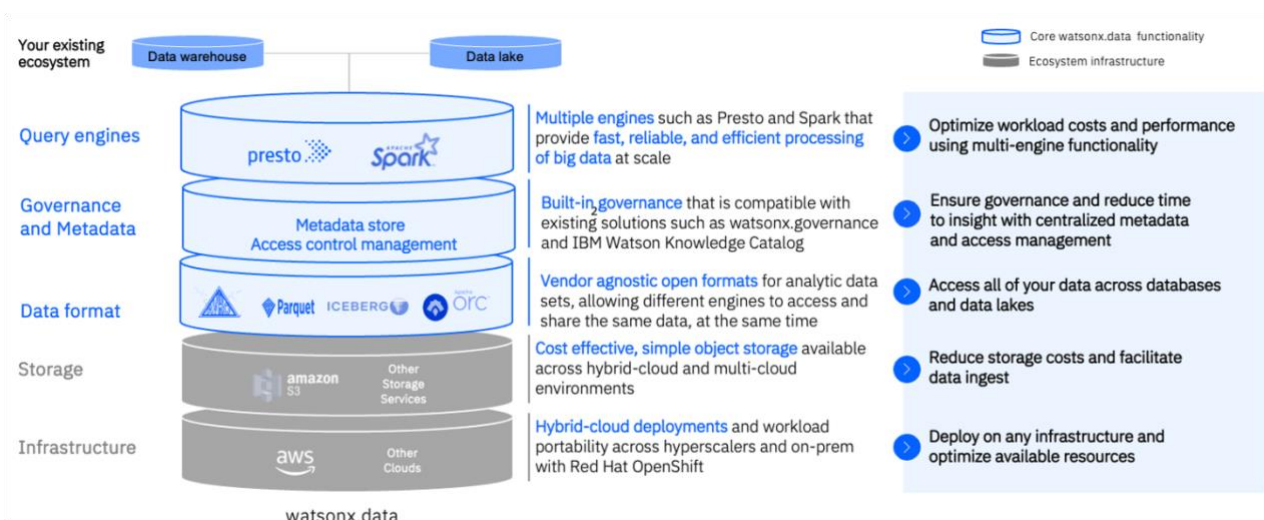


Figure 9. IBM data integration with other cloud systems. Source [7]



### 2.3.1.3. Amazon's Cloud

AWS's data lakes are offered as services that combine data storage and analytics solutions (see Table 10).

Table 10. Data lake core products (AWS cloud approach)

Concept	Description
<b>AWS Lambda</b>	Compute service to run code in an automated self-managed environment (servers, clusters) for serverless applications <a href="https://aws.amazon.com/lambda/">https://aws.amazon.com/lambda/</a>
<b>Amazon OpenSearch Service</b>	Provides robust search (and analytics) capabilities (log files, messages, metrics, config info, documents), including built-in OpenSearch dashboards and Kibana <a href="https://aws.amazon.com/opensearch-service/">https://aws.amazon.com/opensearch-service/</a>
<b>Amazon Cognito</b>	User authentication and access control <a href="https://aws.amazon.com/cognito/">https://aws.amazon.com/cognito/</a>
<b>AWS Glue</b>	Serverless service for data transformation (discovery, preparation, integration) that allows building and monitoring ETL pipelines for data lakes <a href="https://aws.amazon.com/glue/">https://aws.amazon.com/glue/</a>
<b>Amazon Athena</b>	Serverless analytics service able to analyse huge amounts of data (built on open-source Trino and Apache Spark) <a href="https://aws.amazon.com/athena/">https://aws.amazon.com/athena/</a>
<b>Amazon S3</b>	Simple Storage Service to store and protect data for various use cases (data lakes, back-up, low-cost archiving, etc.) <a href="https://aws.amazon.com/s3/">https://aws.amazon.com/s3/</a>
<b>Amazon DynamoDB</b>	Serverless, NoSQL, fully managed database <a href="https://aws.amazon.com/dynamodb/">https://aws.amazon.com/dynamodb/</a>

Although the basic and nuclear core of the data lake is Amazon S3, AWS users usually use additional services to facilitate their work in tagging, searching, sharing, transforming and analysing the data. Siemens uses AWS services to build what they call [Data Lake 2 Go](#). Toyota is another AWS client through its Connected Data Lake [\[8\]](#)

A simple example of data lakes on AWS is provided in Figure 10, which also includes an [example code in GitHub](#)

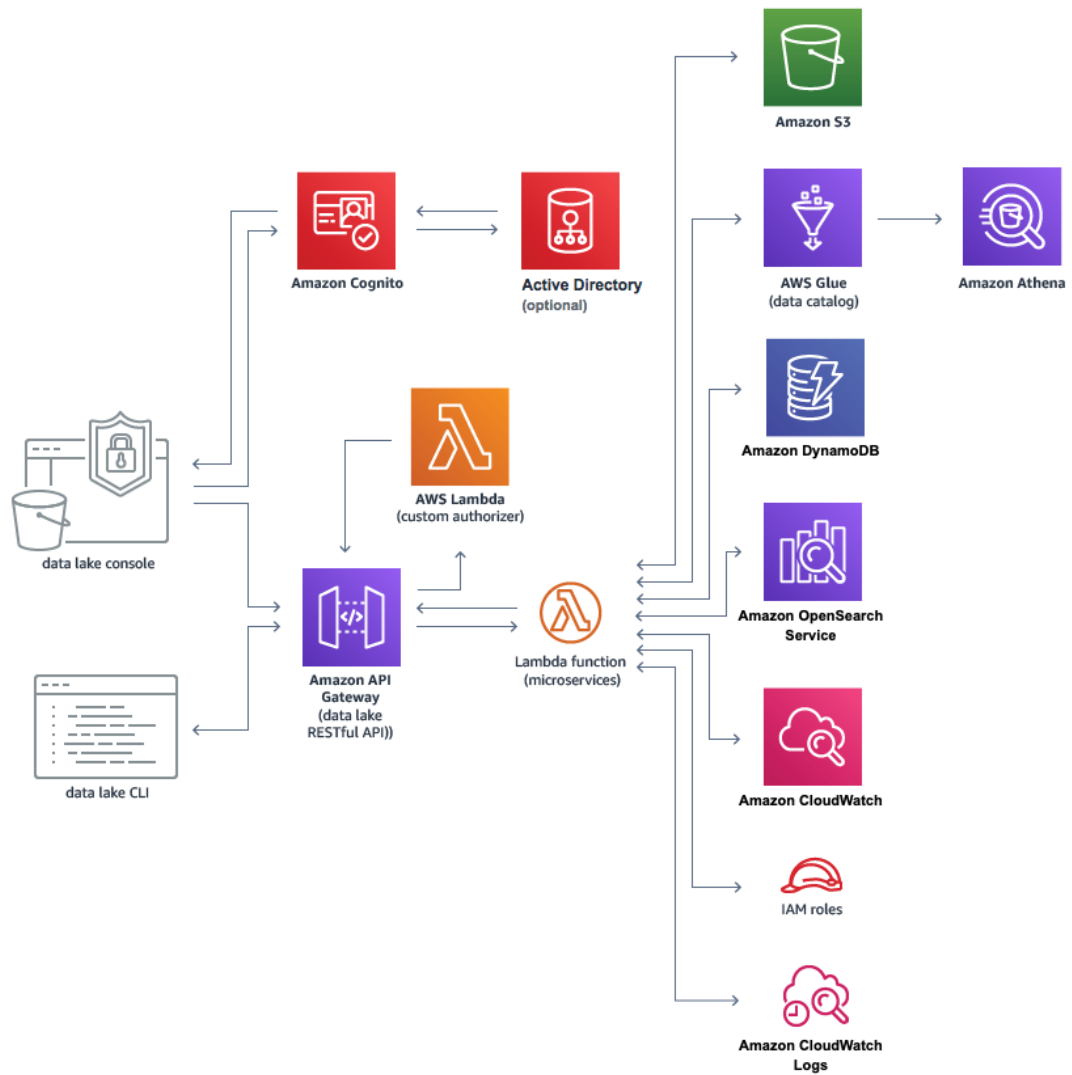


Figure 10. AWS cloud data lake architecture example. Source [9]

### 2.3.1.4. Microsoft’s Cloud

Data lakes by **Microsoft** are also offered as a set of tools that not only target storage services (see Table 11).

Table 11. Data lake core products (Microsoft cloud approach)

Concept	Description
<b>Azure Data Lake Storage (Gen 2)</b>	Single storage platform to integrate all data silos. Gen2 refers to the current implementation; the previous one (Gen1) will be retired in March 2024. Gen2 converges the capabilities of e Gen1 with Azure Blob Storage. <a href="https://azure.microsoft.com/en-us/products/storage/data-lake-storage/">https://azure.microsoft.com/en-us/products/storage/data-lake-storage/</a>
<b>Azure Synapse</b>	Distributed system for storing and analysing large datasets. It typically uses PolyBase (data virtualisation feature for SQL server) to load data from Azure Data Lake Storage <a href="https://azure.microsoft.com/en-us/products/synapse-analytics/">https://azure.microsoft.com/en-us/products/synapse-analytics/</a>
<b>Data Factory</b>	Fully managed, serverless data integration service to construct ETL or ELT process before loading data in Azure Synapse <a href="https://azure.microsoft.com/en-us/products/data-factory/">https://azure.microsoft.com/en-us/products/data-factory/</a>
<b>Microsoft Entra ID</b>	Formerly known as Active Directory, it is a cloud identity and access management solution

	<a href="https://www.microsoft.com/en-us/security/business/identity-access/microsoft-entra-id">https://www.microsoft.com/en-us/security/business/identity-access/microsoft-entra-id</a>
<b>Power BI</b>	A suite of business analytics tools to analyse data and share insights. Power BI can either interact with Analysis Services by querying a semantic model stored or with Azure Synapse <a href="https://www.microsoft.com/en-us/power-platform/products/power-bi/">https://www.microsoft.com/en-us/power-platform/products/power-bi/</a>
<b>Azure Purview</b>	This data governance service maintains a unified data catalogue as well as data landscape maps. Features include automated data discovery, sensitive data classification, and data lineage <a href="https://azure.microsoft.com/en-us/products/purview/">https://azure.microsoft.com/en-us/products/purview/</a>
<b>Azure Monitor</b>	Collects and analyses data (e.g., performance metrics and activity logs) on environments and Azure resources. <a href="https://azure.microsoft.com/en-us/products/monitor/">https://azure.microsoft.com/en-us/products/monitor/</a>

A simple example of a data lake, including data management, is depicted in Figure 11. Here, it is important to use the data governance portal (MS Purview), as sometimes data lakes lack this component or are deployed with basic features. Another example of an Azure architecture as a data lakehouse can be found in [11].

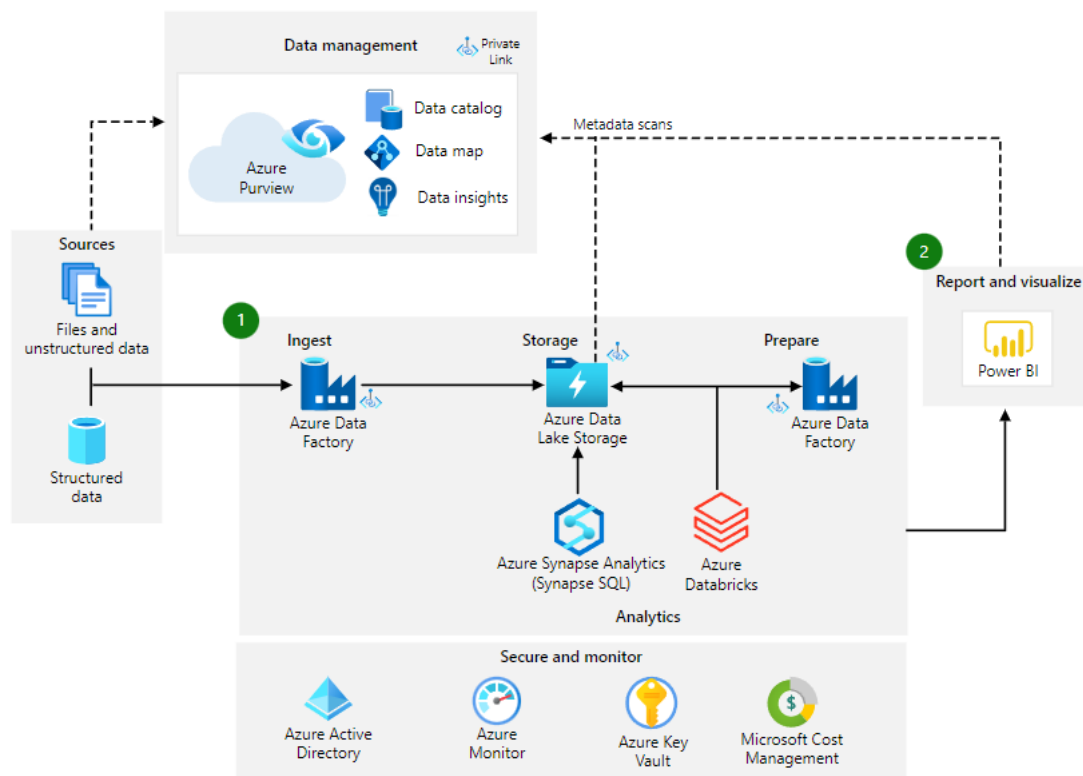


Figure 11. MS Azure cloud data lake architecture example with data management. Source [10]

### 2.3.1.5. Short Analysis and Comparison of the cloud-approaches

If one tries to compare all previous cloud data platforms, it appears **challenging** because they are composed of a set of different components that are typically designed and deployed for specific use cases; there is no single architecture scenario for a cloud provider, and some meetings and discussions are needed between organisations and cloud provider consultants to understand the data requirements, the business goals, and develop afterwards the most suitable architecture for their needs. Anyway, just to align cloud components in the different data stages, a summary picture is provided below (*note: some components might have been updated or renewed by cloud providers*).



Figure 12. Cloud components offered by GCP, AWS and Azure. Source [12]

## 2.4. Best practices for data lakes

Considering the diversity of possible configurations for data lakes, it seems relevant to provide recommendations to follow whenever a data lake is built within an organisation. These best practices intend to generate awareness of the data lake's different steps or building blocks, summarised in the table below. They have been extracted from various suggestions and recommendations in online resources from the documentation sites of AWS, Google Cloud, Microsoft Azure and IBM Data Lake architectures.

Table 12. Best practices for data lakes

Best practice item	Description
Check closer Data ingestion	Do not propose a data lake architecture before deeply analysing the input data requirements: size, performance, format, etc., as it can have a significant impact on the performance later. Additionally, knowing the input data is crucial to understanding how an organisation works and what types of data are relevant. A common statement about data lakes is: "Store now, analyse later." That should not be followed unless you want to design completely blindly.
Keep the raw data	Try to preserve one logical area of the data lake for the raw data without any transformation (except, maybe, personal information for GDPR reasons). Keeping the original data format allows for a clear, unified starting point, which can be later processed on different cleansing or filtering data pipelines for different users and purposes



<p><b>Favour Computing over storage</b></p>	<p>Data storage has become relatively cheap in recent years and easy to scale. Computing performance might be tricky as the data grows. Therefore, storing multiple copies of the same data is practical if the overall performance increases. This means that the same data might be transformed into different formats suitable (optimised) for the final application which is consuming it (e.g., two different ML models)</p>
<p><b>Keep Retention policy</b></p>	<p>Most of the data might not be stored forever in the data lake. After a certain (configurable) period, some pieces of data might not be relevant (no application is making use of them) and should be deleted/discarded. There are two main reasons: (i) cost and (ii) compliance. On the one hand, data storage is relatively cheap compared to data warehouses but not free. On the other hand, some regulatory requirements (e.g., GDPR) might dictate the deletion of personal data after a certain period or based on the user's request</p>
<p><b>Partition the data</b></p>	<p>By partitioning the data, the query costs are reduced, similar to for databases where you create multiple tables. A partition is a logical entity (e.g., a Hive meta store item mapping to a folder on a S3 bucket). A common approach is partitioning by timestamp (minute, hour, day) depending on the needed granularity of the target applications. This time partitioning is also helpful for enforcing automated data retention policies, as data might become useless after a period of time</p>
<p><b>Readable file format</b></p>	<p>It is highly recommended to use open-source formats, such as Apache Parquet or ORC (commonly used in Hadoop ecosystems), rather than proprietary ones. Parquet is typically better for analytical workloads and large and complex data structures. ORC is intended for write-intensive tasks and data modifications. Additionally, columnar storage makes data typically easier to read (for analytics applications). Apache Parquet and ORC are column-based standards, whereas Apache Avro is file-based (more suitable for non-regular queries and ETL pipelines) For compression mechanisms applied to data for cost reasons, using a 'weak' compression standard is typically better to speed up the decompression and use compute power in real analytics.</p>
<p><b>Merge small files</b></p>	<p>Data produced by data streams and logs might produce many small events daily. For performance reasons, it is wiser to merge or compact these files into one single one</p>
<p><b>Ensure access control and governance</b></p>	<p>Data is probably one of the most valuable assets of an organisation, and therefore, it should be protected by some sort of access control list feature. It is critical to ensure who can access (and modify) the data, and significant time should be dedicated to this task. Data governance features might also help here, as good management of data with a proper data catalogue improves the availability (discoverability) of the data so that it can be properly used and audited.</p>
<p><b>Be adaptive and flexible</b></p>	<p>Besides data requirements, it is important to consider the user's skills when designing the data lake. A complex design architecture might appear unapproachable to a customer. You might adapt the initial (complex) design into a simpler one so that the user becomes familiar with it while remaining open and modular to additional enhancements when there is a need to scale further or add new features.</p>

## 2.5. Data Lakes in the film-making industry

Data lakes are increasingly being used in the filmmaking industry to improve the **efficiency** and **creativity** of the filmmaking process. The data centralisation through data lakes brings efficiency by simplifying data access



and retrieval for filmmakers, editors, and production teams who need quick access to varied data types across different filmmaking phases. Direct efficiency benefits include streamlined data access, automated data ingestion, enhanced collaboration, and cost savings. Besides, data lakes enhance creativity in filmmaking by providing insights into audience preferences and enabling easy access to reusable assets, which inspire new ideas and allow agile adjustments to content.

Data lakes can **store** various **data types**, including structured, semi-structured, and unstructured. This makes them ideal for storing the diverse types of data generated in filmmaking, such as:

- **Video and audio footage**
- **Production costs**
- **Social media data**
- **Box office data**
- **Audience demographics**

By storing all of this data in a **single location**, data lakes make it easier for filmmakers to **access** and **analyse** the data they need to make **informed decisions** throughout the filmmaking process.

Here are some **specific examples** of how data lakes are being used in the film industry:

- **Pre-production:** Filmmakers can use data lakes to **analyse audience and box office data to identify trends and predict which types of films are likely to succeed**. They can also use data lakes to budget their films and create production schedules.
- **Production:** Filmmakers can use data lakes to **track production costs** and progress and to identify potential problems early on. They can also use data lakes to **manage their creative assets**, such as **video** and **audio** footage.
- **Post-production:** Filmmakers can use data lakes to **analyse audience feedback** and **social media data** to identify which parts of their films resonate with audiences. They can also use data lakes to create and **test different versions** of their films before releasing them to the public.

In addition to these specific examples, data lakes can also be used to **improve** the overall **efficiency** and **creativity** of the filmmaking process by:

- Making it **easier to share data** between different departments: Data lakes can make it easier for filmmakers to share data between departments, such as production, marketing, and distribution. This can help to improve **communication and collaboration** throughout the filmmaking process.
- Enabling **new forms of data analysis**: Data lakes can enable new forms of data analysis, such as **machine learning** and **artificial intelligence**. This can help filmmakers to gain new insights from their data and to make better decisions throughout the filmmaking process.

Overall, data lakes are a **powerful tool** that can be used to improve the efficiency and creativity of the filmmaking process. As the film industry continues adopting new technologies, data lakes will likely play an increasingly important role in this sector.

Here are some examples of film studios and production companies that are using data lakes:

- **Disney:** Disney uses a data lake to store and analyse data from its various businesses, including films, television, and theme parks. Disney uses this data to improve its marketing campaigns, create new content, and make better business decisions.



- **Netflix:** Netflix utilises a data lake to store and analyse data from its millions of subscribers. Netflix employs this data to recommend new content to subscribers, improve its streaming service, and decide which new shows and movies to produce.
- **Warner Bros.:** Warner Bros uses a data lake to store and analyse data from its films, television shows, and video games. Warner Bros employs this data for marketing campaigns, new content creation, and better decision-making.

If we restrict the scope to the European market, we can also list some relevant companies, such as Studio Babelsberg (Germany), Canal+(France), BBC (United Kingdom), Mediaset (Italy) and Beta Film (Germany).

In summary, data lakes play a crucial role in the film-making industry in many ways, as depicted in the Table below.

Table 13. Data Lake Features

Data Lake Feature	Description
<b>Content Storage</b>	Data lakes can store massive amounts of raw and processed video footage, audio files, images, and other media assets. This storage capacity is essential for preserving and managing the vast content generated during production.
<b>Data Ingestion</b>	Film productions generate data from various sources, including cameras, sound equipment, post-production software, etc. Data lakes allow for data ingestion from these diverse sources, making it easier to centralise and access all relevant information in one location.
<b>Metadata Management</b>	Metadata is critical in filmmaking for tracking and organising assets. Data lakes can store and manage metadata associated with each media asset, including details like scene descriptions, cast and crew information, shooting locations, and timestamps. This metadata helps streamline the production and post-production processes.
<b>Data Analysis</b>	Data lakes can store audience preferences, market trends, and box office performance data. Film studios and production companies can leverage this data to make data-driven decisions, such as selecting film projects, marketing strategies, and distribution plans
<b>Collaboration</b>	Film-making often involves collaboration between various teams and individuals, including directors, producers, editors, and visual effects artists. Data lakes provide a centralised platform where these teams can access and work on the same data simultaneously, improving collaboration and efficiency.
<b>Cost Management</b>	Film production is expensive, and managing costs is crucial. Data lakes can store financial data, allowing studios to track expenses, budgets, and cost projections in real time. This helps control and optimise production costs.
<b>Archiving and Preservation</b>	Film studios must often preserve their content for future use or historical purposes. Data lakes provide a reliable long-term storage solution, ensuring valuable media assets remain accessible and secure.
<b>Personalisation</b>	Data lakes can store viewer data and preferences, enabling film studios to offer personalised content recommendations and marketing campaigns. This can enhance audience engagement and revenue generation
<b>Security and Compliance</b>	The film industry deals with sensitive content, so data lakes must have robust security measures to protect against data breaches and unauthorised access. Additionally, compliance with copyright and content distribution regulations is essential, and data lakes can help manage compliance data
<b>Streaming Services</b>	Many film studios are expanding into streaming platforms. Data lakes can store content libraries, user data, and streaming analytics, helping studios optimise their streaming offerings and enhance the user experience

### 2.5.1. Content perspective (content-type)

In the film-making industry, the diversity and richness of data play a critical role in every production phase, from pre-production planning to post-production and distribution. The SCENE's data lake embodies this diversity by encompassing various content types crucial for filmmakers, researchers, and enthusiasts alike. This subchapter explores the content perspective of these data lakes, focusing on the variety of data types, content acquisition and integration strategies, and management practices that facilitate the effective utilisation of film-related data.

#### Variety of Data Types

The SCENE data lakes can store a comprehensive range of content types to support the multifaceted needs of the film-making industry, including:

- **Video:** Raw footage, edited clips, trailers, and full-length features.
- **Audio:** Soundtracks, sound effects, dialogue tracks, and ambient sounds.
- **Images:** Posters, production stills, and behind-the-scenes photographs.
- **Text:** Scripts, storyboards, production notes, and press releases.
- **Metadata:** Information about the content, including cast, crew, production details, and content descriptors.

This diversity ensures that users can find almost any content needed for film production, analysis, or educational purposes within a unified repository.

#### Content Acquisition and Integration

Acquiring and integrating content into the SCENE data lake involves a multi-faceted approach:

- **User-Generated Content:** Leveraging contributions from filmmakers, scriptwriters, and the film-making community to enrich the data lake with unique and diverse content.
- **Licensed Databases:** Partnering with film databases and archives to incorporate comprehensive datasets, ensuring a wide coverage of historical and contemporary film content.
- **Content Providers:** Collaborating with production companies, distributors, and other industry stakeholders to secure current and relevant content flow.

These strategies ensure the SCENE data lake is continually updated with fresh, diverse, and comprehensive content, making them a valuable resource for the film industry.

#### Content Management Strategies

Effective content management is key to maximising the utility of the data lake. The SCENE project intends to explore several strategies to manage the vast amounts of data:

- **Categorisation and Tagging:** Implementing a detailed categorisation and tagging system based on the SCENE ontology allows for precise content classification, facilitating easy retrieval and analysis.
- **Metadata Usage:** Enhancing content with rich metadata, including descriptors for genre, themes, technical specifications, and contextual information, to improve searchability and discovery.



- **Dynamic Annotation:** Allowing users to contribute annotations and tags further enriches the content's metadata, providing deeper insights and perspectives on the data.

In summary, the content perspective of the SCENE data lake highlights the project's commitment to serving the diverse needs of the film-making industry. By managing various content types through strategic acquisition, integration, and management practices, SCENE ensures its data lakes remain an invaluable resource for filmmakers, researchers, and enthusiasts. Incorporating advanced categorisation, tagging, and metadata enriches the user experience, making exploring and utilising film-related data more intuitive and effective.

### 2.5.2. Structure perspective (metadata)

The structure perspective in data lakes, particularly within the film-making industry, is pivotal for managing, accessing, and analysing the voluminous and varied data these repositories hold. This perspective emphasises the role of metadata in transforming raw data into a **structured, intelligible, and valuable resource**. This subchapter delves into the importance of metadata in enhancing the structure of data lakes, the methodologies employed in metadata management, and the impact of these practices on the accessibility and utility of data.

Metadata is the foundational layer covering data lakes with **meaning and structure**. In the context of the film-making industry, metadata can encompass a wide range of information, from technical details about video and audio files (such as format, duration, and bitrate) to descriptive information about content (such as themes, genres, cast, and crew).

Key areas of focus when dealing with metadata are:

- **Metadata Standards and Schemas:** Adopting standardised metadata schemas ensures consistency and interoperability across different systems and platforms. Common standards used in the film industry include Dublin Core, MPEG-7, and custom schemas developed to cater to specific film production and distribution needs.
- **Metadata Generation and Enrichment:** Automated tools and manual processes are crucial in generating and enriching metadata. Machine learning algorithms can analyse content to tag genres, identify objects and characters in scenes, and even assess sentiment and themes. Manual tagging, albeit more labour-intensive, adds invaluable depth and accuracy to metadata, especially in capturing nuanced content attributes.
- **Metadata Storage and Management:** Efficiently storing and managing metadata involves leveraging database technologies that can handle the complexity and scale of data lakes. NoSQL databases, graph databases, and dedicated metadata management systems are commonly employed for flexibility, scalability, and robust querying capabilities.

The strategic organisation and management of metadata directly influence the **discoverability** and **accessibility** of content within data lakes. Well-structured metadata allows for advanced search functionalities, enabling users to perform complex queries based on various content attributes. This capability is especially crucial in the film-making industry, where the ability to locate and access specific content quickly can significantly impact creative processes and decision-making.

Some implementation **examples** could be the following ones:

- **Automated Tagging for Raw Footage:** Utilizing AI-driven tools to analyse and tag raw footage with descriptive metadata facilitates efficient post-production sorting and retrieval.



- **Semantic Search Engines:** Developing search engines that leverage metadata to understand the context and semantics behind user queries, offering more accurate and relevant search results.
- **Content Recommendation Systems:** Using metadata to power recommendation algorithms that suggest content based on user preferences, viewing history, and content similarity.

While metadata significantly enhances the structure and utility of data lakes, managing it presents several **challenges**, including ensuring **data quality**, maintaining **consistency** across large datasets, and protecting **sensitive information**. Adopting best practices such as continuous data quality assessments, employing metadata standards, and implementing robust data governance policies are critical to addressing these challenges.

The structure perspective centred around the **comprehensive management** and **utilisation of metadata** is essential for transforming data lakes from mere storage repositories into dynamic, accessible, and valuable resources. Effective metadata management streamlines production workflows in the film-making industry and opens up new avenues for content exploration, analysis, and monetisation. As technologies evolve, so will the metadata management strategies, continually enhancing the structure and capabilities of data lakes in the film industry.

### 2.5.3. Usability perspective (user perspective)

The usability perspective, particularly within data lakes and ontologies for the film-making industry, focuses on the end-user experience. This encompasses how filmmakers, researchers, and industry professionals interact with and benefit from the complex ecosystem of data and metadata. This subchapter explores key aspects such as user interface design, personalisation features, and feedback mechanisms that enhance usability and user engagement.

#### User Interface and Experience Design

Designing user interfaces (UI) for accessing and interacting with data lakes requires a deep understanding of the end users' needs and workflows. The UI should be intuitive, allowing users to navigate vast amounts of data and metadata easily. Key considerations include:

- **Simplicity:** A clean, uncluttered interface prioritising essential functionalities and information.
- **Search and Discovery:** Advanced search features, including faceted search and semantic querying, to efficiently locate relevant data.
- **Visualization Tools:** Dynamic visualisation tools representing complex data relationships and patterns, aiding in data analysis and insights generation.

Currently, in the SCENE project, the user interface is expected to be provided from the different modules and tools that build the SCENE platform. For the SCENE data lake, the UI will be mainly administrative. Therefore, these tools should exploit the data lake APIs to enrich the user experience.

#### Personalisation and Recommendations

Personalisation enhances the usability of data lakes by tailoring the user experience to individual preferences and needs. This can be achieved through:

- **User Profiles:** Users can create profiles that store preferences, search history, and frequently accessed data.
- **Recommendation Systems:** Leveraging machine learning algorithms to recommend relevant data, films, or research based on the user's past interactions and preferences.



- **Adaptive Interfaces:** Interfaces that adapt to the user's skill level and preferences, offering a customised experience that evolves over time.

Currently, in SCENE, this feature is expected to be mainly supported by the modules and tools; however, the data lake might provide supporting backend help if required.

### Feedback Mechanisms and Continuous Improvement

Feedback from users is invaluable for the iterative improvement of data lakes and ontologies. Implementing effective feedback mechanisms enables the identification of usability issues and the discovery of new requirements:

- **User Feedback Tools:** Integrated tools for collecting user feedback directly within the platform, including surveys, feedback forms, and usability testing sessions.
- **Analytics and Usage Tracking:** Analysing user interaction data to identify patterns and potential areas for improvement.
- **Community Engagement:** Creating forums or user groups to encourage sharing tips, best practices, and user-generated content.

The SCENE platform already envisions tools to handle the previous mechanisms; however, the handled data can be stored in the data lake to facilitate interoperability among the tools, if needed.

### Challenges and Strategies

Ensuring usability in complex data ecosystems involves overcoming several challenges, such as managing the diversity of user needs, ensuring data privacy and security, and maintaining system performance. Strategies to address these challenges (currently supported by data lakes) include:

- **User-Centric Design:** Engaging with users throughout the design and development process to ensure the system meets their needs.
- **Scalability and Performance Optimization:** Implementing scalable architectures and optimising performance to handle large datasets without compromising user experience.
- **Privacy and Security Measures:** Ensuring that personalisation and user data are handled securely, with clear policies and user controls.

## 2.6. Examples of film-related Data Lakes

Below are summary tables (Table 14, Table 15, Table 16) of data lakes related to the film-making industry. In these cases, the data lake is perceived as a front-end for searching for films, TV shows, or actors. We could not find any covering the four phases (pre-production, production, post-production and distribution).

*Table 14. IMDb (Internet Movie Database) summary*

IMDB	Description
<b>Content &amp; General info</b>	Extensive information about films, TV shows, actors, production crew, and other related data. The official IMDb website can be accessed at <a href="https://www.imdb.com">IMDb.com</a>  <b>API:</b> <ul style="list-style-type: none"> <li>• IMDb provides access to its data through an API available via the AWS Data Exchange. There is an <a href="#">official API online documentation</a></li> </ul>



- The API uses GraphQL for querying data sets, making it efficient and customisable for various data needs. Sample queries and detailed instructions on using the API with different programming languages are available in the [API documentation](#)
- IMDb also offers bulk data access in JSON Lines file format. This allows users to download large datasets for offline analysis. The bulk data is updated regularly and includes extensive metadata about movies, TV shows, and more. Documentation for bulk data, including schemas and data dictionaries, is available [here](#)

**Data size:** The bulk data sets are pretty large, and the size can vary depending on the specific data product and the amount of data being downloaded

**Ontology and Schema:** The IMDb data is structured using a well-defined schema, which ensures consistency and ease of use. Detailed schemas for both API and bulk data are provided, documenting the format and relationships between different data entities. This helps in understanding the ontology used by IMDb, which includes entities such as titles, names, ratings, and box office data. No explicit ontology is available.

<p><b>Structure</b></p>	<p>Organised data with rich metadata, user reviews, ratings, and extensive tagging. IMDb organises its data into several main categories, including:</p> <ul style="list-style-type: none"> <li>• <b>Movies:</b> Titles, release dates, genres, runtime, plot summaries, cast and crew, user ratings, and reviews.</li> <li>• <b>TV Shows:</b> Episode lists, seasons, air dates, and episode summaries.</li> <li>• <b>Actors:</b> Biographical data, filmography, photos, and awards.</li> </ul> <p>Each entity (movie, actor, etc.) has a unique identifier (IMDb ID)</p>
<p><b>Usability</b></p>	<p><b>Accessibility:</b> Accessible via a comprehensive API, widely used in the industry for various applications.</p> <p><b>Integration:</b> Suitable for integration into various applications for detailed film-related data retrieval</p> <p><b>Ease of Use:</b></p> <ul style="list-style-type: none"> <li>• <b>Documentation:</b> Detailed documentation is provided, which helps developers understand the API's capabilities and how to use it effectively.</li> <li>• <b>Complexity:</b> Due to the data's richness and the API's complexity, there can be a learning curve, especially for those unfamiliar with AWS services.</li> </ul> <p><b>Support:</b> Extensive support through AWS, including forums, official support, and community contributions.</p> <p><b>Community Involvement:</b> Active user base that contributes reviews, ratings, and additional data, enhancing the richness and usability of the database.</p>
<p><b>Other info (strengths)</b></p>	<p><b>Comprehensive Database:</b> Extensive movies, TV shows, and personality coverage.</p> <p><b>User-Generated Content:</b> Rich with user reviews and ratings, providing insights into public opinion.</p> <p><b>Detailed Metadata:</b> Extensive information on each entry, including trivia and detailed biographies.</p>



Table 15. TMDb (The Movie Database)

TMDb	Description
<p><b>Content &amp; General information</b></p>	<p>User-contributed data on films and TV shows, including detailed cast, crew, release dates, and genre information. The official website for TMDb (The Movie Database) is <a href="#">TMDb</a></p> <p><b>API:</b></p> <ul style="list-style-type: none"> <li>• TMDb offers a robust RESTful API that provides access to a wide range of data on movies, TV shows, and actors. The API is available in two versions: <a href="#">v3</a> and <a href="#">v4</a>, each with specific capabilities and endpoints</li> <li>• Examples for <a href="#">Search &amp; Query</a> are available</li> <li>• TMDb also provides bulk data export options. Users can access comprehensive datasets for offline analysis, which include extensive metadata about movies, TV shows, and other entities.</li> </ul> <p><b>Data Size and Format:</b> The bulk data is available in various formats, including JSON. The size of the datasets can vary significantly.</p> <p><b>Ontology and Schema:</b> TMDb's data is organised using well-defined schemas, ensuring consistency and ease of use. The API documentation includes details about the data structure, relationships between entities, and the specific fields available for each type of data (e.g., movies, TV shows, actors). No explicit ontology is available.</p>
<p><b>Structure</b></p>	<p>Community-driven with extensive tagging and metadata. TMDb (similar to IMDb) categorises its data into:</p> <ul style="list-style-type: none"> <li>• <b>Movies:</b> Titles, release dates, genres, overviews, cast and crew, user ratings, trailers, and posters.</li> <li>• <b>TV Shows:</b> Episode guides, seasons, air dates, and episode summaries.</li> <li>• <b>Actors:</b> Biographies, filmography, photos, and social media links.</li> </ul> <p>Each entity has a unique identifier (TMDb ID) TMDb uses a JSON-based schema (no JSON-LD) to represent its data</p>
<p><b>Usability</b></p>	<p>It provides a robust API for data access widely integrated into various applications and services, supporting real-time updates</p> <p><b>Integration:</b></p> <ul style="list-style-type: none"> <li>• <b>Flexible API:</b> The API supports a wide range of queries for movies, TV shows, etc., making it versatile for different applications.</li> <li>• <b>Image Integration:</b> High-quality images and posters can be integrated into applications, enhancing visual appeal.</li> </ul> <p><b>Ease of Use:</b></p> <ul style="list-style-type: none"> <li>• <b>Documentation:</b> Provides clear and concise documentation with examples and guidelines, making it easy for developers to get started.</li> <li>• <b>Community Contributions:</b> Data is user-contributed and regularly updated, ensuring a comprehensive and current database.</li> </ul> <p><b>Support:</b> Good support structure with forums and active community engagement.</p>



	<p><b>Community Engagement:</b> Strong community involvement ensures data is continuously updated and enriched with user-generated content.</p>
Other info (strengths)	<p><b>Rich Visual Content:</b> High-quality images and trailers available for integration.</p> <p><b>Community Engagement:</b> An active user community contributes to and updates data.</p> <p><b>Detailed Search and Query Options:</b> Extensive filtering and search options are available via API.</p>

Table 16. EIDR (Entertainment Identifier Registry)

Perspective	Description
Content & General information	<p>Unique identifiers for movie and television assets to ensure data consistency and integration across platforms. The official website for EIDR (Entertainment Identifier Registry) is <a href="https://www.eidr.org">EIDR.org</a>.</p> <p><b>API:</b> EIDR provides a comprehensive identification system for audio-visual content. The API and related technical documentation are available on the EIDR website, allowing integration into various business systems. Key documents include:</p> <ul style="list-style-type: none"> <li>• <b>Introduction to the EIDR Data Model:</b> Overview of how data is structured within EIDR.</li> <li>• <b>Registry Technical Overview:</b> Detailed technical specifications of the registry.</li> <li>• <b>EIDR Data Fields Reference:</b> Explanation of the data fields used.</li> <li>• <b>Programmers Guide and REST API Reference:</b> Guidelines and reference material for developers.</li> </ul> <p>For detailed technical documentation, you can visit the <a href="#">EIDR Technical Documentation page</a>.</p> <p><b>Data Access and Size:</b> EIDR IDs and metadata can be accessed via API or the EIDR website. The size and scope of the data depend on the number of assets registered and the depth of metadata associated with each ID.</p> <p><b>Ontology and Schema:</b> EIDR uses a well-defined schema to manage the relationships and metadata for audio-visual content. This schema supports various product types and ensures consistency and interoperability across different systems. Detailed schemas and data models are provided in the technical documentation to help users integrate EIDR IDs into their workflows.</p>
Structure	<p>Focused on providing a standardised framework for identifying media content. EIDR is structured around unique identifiers for:</p> <ul style="list-style-type: none"> <li>• <b>Content:</b> Titles, alternate titles, release dates, associated metadata (e.g., cast, crew, production details), and versions/edits of the content.</li> <li>• <b>Relationships:</b> Relationships between different versions, edits, and related content items</li> </ul>
Usability	<p>Utilised by significant studios and data aggregators for managing digital media assets. API access and real-time processing support</p> <p><b>Integration:</b></p>



- **Interoperability:** Designed to integrate seamlessly with various business systems, enhancing the ability to track and manage audio-visual content across different platforms.
- **Universal Identifiers:** EIDR IDs are widely accepted and used in the industry, facilitating interoperability and reducing the risk of data duplication.

**Ease of Use:**

- **Documentation:** Comprehensive technical documentation is available, detailing the data model, API usage, and best practices.
- **Learning Curve:** This may have a steeper learning curve for those unfamiliar with content identification and tracking systems.

**Support:** Offers training sessions, events, and extensive documentation to help users get the most out of the system.

**Industry Adoption:** Broad adoption across the industry ensures users can benefit from a widely recognised and standardised system.

<b>Other info (strengths)</b>	<p><b>Universal Identifier System:</b> Ensures unique identification and tracking of content across various platforms.</p> <p><b>Interoperability:</b> Facilitates interoperability between different systems and platforms.</p> <p><b>Comprehensive Metadata:</b> Detailed metadata and relationship mapping between content items.</p>
-------------------------------	--

While IMDb, TMDb, and EIDR are among the most comprehensive and widely used film-related data lakes, several other platforms could also be valuable for different aspects of film data. Below are a few additional options (see Table 17). These additional data lakes might complement IMDb, TMDb, and EIDR by providing specific data types such as reviews, box office performance, or region-specific information.

*Table 17. Other film-related data lakes*

Tool	Description
<a href="#"><u>Rotten Tomatoes</u></a>	<p><b>Content:</b> Movie and TV show reviews, critic scores, audience scores, and aggregated ratings.</p> <p><b>API Access:</b> Provides access to movie ratings and review data.</p> <p><b>Usability:</b> Useful for integrating review scores and audience sentiment into applications.</p>
<a href="#"><u>Box Office Mojo</u></a>	<p><b>Content:</b> Detailed box office revenue data for movies.</p> <p><b>API Access:</b> Through IMDb's API (since it is part of IMDb).</p> <p><b>Usability:</b> Essential for applications requiring detailed financial performance data of films.</p>
<a href="#"><u>OMDb API (The Open Movie Database)</u></a>	<p><b>Content:</b> Comprehensive movie and TV show information, including ratings from IMDb, Rotten Tomatoes, and Metacritic.</p> <p><b>API Access:</b> Simple and easy-to-use API with no extensive setup required.</p> <p><b>Usability:</b> Ideal for developers looking for a quick and straightforward way to access movie data.</p>
<a href="#"><u>The British Film Institute (BFI)</u></a>	<p><b>Content:</b> Database of British films and television programs, including production data, credits, and archival material.</p>

	<p><b>API Access:</b> Limited public API, but data can be accessed for research purposes.</p> <p><b>Usability:</b> Valuable for research and applications focusing on British cinema and television.</p>
<a href="#">Letterboxd</a>	<p><b>Content:</b> User-generated movie ratings, reviews, watchlists, and film diaries.</p> <p><b>API Access:</b> Limited API access is required to retrieve user-generated content and film data.</p> <p><b>Usability:</b> Useful for social and community-driven film applications.</p>

### 3. Requirements Evaluation and Proposed Solution

#### 3.1. General vision

The SCENE architecture is thoroughly described in **deliverable D2.6**<sup>1</sup>; in this section, we might just summarise it from the perspective of the data lake. The data lake and its potential management through ontologies comprise the **data layer** in the overall architecture (see Figure 13), together with the blockchain module.

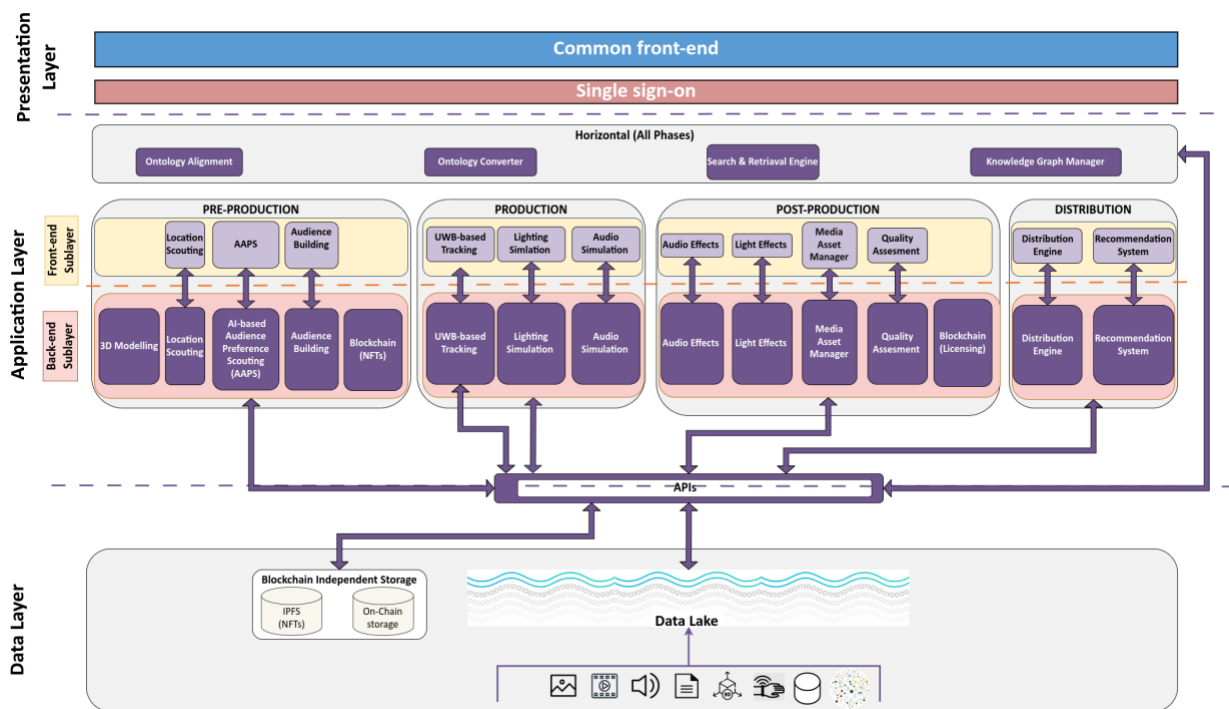


Figure 13. SCENE High-Level architecture

Three main considerations appear relevant:

- The **blockchain module** is used for a **decentralised approach** while interacting with external (final end-users). In contrast, the **data lake module** is a centralised approach that manages and processes common data for the different SCENE tools. They complement each other and are used for different purposes.
- The **end-user** interacting through the front end seems quite distant from the data lake and might not **directly perceive its benefits**. From this perspective, using data lakes and ontologies seems to support the different SCENE tools by providing enhanced storage systems and **potential semantic management** through ontologies. This aspect will be better explained in section 4.3 about gathering requirements for data lakes and ontologies.

<sup>1</sup> D2.6 SCENE Reference Architecture.R1

- The communication between the data lake (even the data layer) and the SCENE tools is performed through **APIs**. From this perspective, we should provide APIs for the **four main blocks of the data lake** as described in the **functional viewpoint** of the architecture in D2.6:
  - o Storage layer
  - o Analysis layer
  - o Discovery layer
  - o Ingestion layer

These four blocks are briefly described below with some examples and technological options.

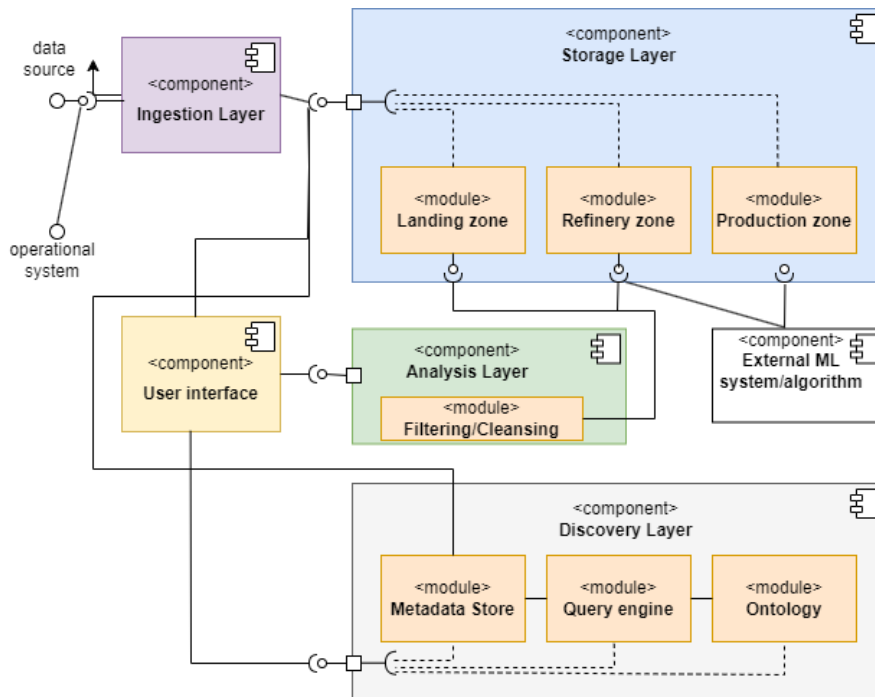


Figure 14. Data lake component diagram (as for D2.6)

### 3.1.1. Ingestion Layer: Incorporating New Data

The **Ingestion Layer** integrates external and internal data sources into the platform using ETL (Extract, Transform, Load) pipelines.

- **Example Process:** A film production company uploads multimedia files, such as video footage and related metadata (title, director, filming location).
- **ETL Pipeline:** The data is extracted from its original source (local server or cloud storage), transformed into a standardised format (e.g., JSON, CSV), and loaded into the **Storage Layer**. If required, metadata is enriched with additional information (e.g., location data or technical details about the footage).

Candidate technologies for data ingestion are provided below. The first four are open-source, whereas the last three are associated with cloud-based approaches.



Table 18. Candidate technologies for data ingestion

Technology	Pros	Cons
<a href="#">Apache NiFi</a>	<ul style="list-style-type: none"> <li>- Easy-to-use GUI interface for flow-based programming</li> <li>- Supports a wide range of data formats and sources</li> <li>- Strong integration with the Apache ecosystem</li> <li>- Full ETL support</li> </ul>	<ul style="list-style-type: none"> <li>- Can become resource-intensive at scale</li> <li>- Complex flows can be harder to manage and debug</li> <li>- Limited performance tuning options</li> </ul>
<a href="#">Apache Kafka</a>	<ul style="list-style-type: none"> <li>- High-throughput and low-latency real-time data streaming</li> <li>- Distributed and fault-tolerant</li> <li>- Good integration with various analytics platforms</li> </ul>	<ul style="list-style-type: none"> <li>- Requires significant setup and maintenance overhead</li> <li>- Not ideal for small datasets</li> <li>- Complexity increases with scaling</li> <li>- Partial ETL support (requires tools like Kafka Streams, and Kafka Connect for transformations)</li> </ul>
<a href="#">Apache Flume</a>	<ul style="list-style-type: none"> <li>- Well-suited for ingesting massive quantities of log data into Hadoop ecosystems</li> <li>- Fault-tolerant and highly reliable for moving large volumes of data</li> <li>- Supports distributed architecture</li> </ul>	<ul style="list-style-type: none"> <li>- Limited to mostly log data and lacks flexibility for other use cases</li> <li>- May require custom integrations for non-Hadoop ecosystems</li> <li>- Outdated in comparison to modern ingestion tools</li> <li>- No ETL support (ingestion only)</li> </ul>
<a href="#">Node-RED</a>	<ul style="list-style-type: none"> <li>- User-friendly, visual flow-based interface for rapid prototyping and integrations</li> <li>- Lightweight and ideal for event-driven and API-based workflows</li> <li>- Flexible and suitable for quick integration tasks</li> </ul>	<ul style="list-style-type: none"> <li>- Less suited for high-volume, complex data ingestion processes;</li> <li>- Limited in advanced data lineage tracking;</li> <li>- May lack performance tuning and scalability for enterprise-level ingestion</li> </ul>
<a href="#">AWS Kinesis</a>	<ul style="list-style-type: none"> <li>- Fully managed service with seamless AWS integration</li> <li>- Scalable real-time streaming data processing</li> <li>- Simplifies handling real-time data ingestion for cloud-based workloads</li> </ul>	<ul style="list-style-type: none"> <li>- Tied to the AWS ecosystem, reducing flexibility with other platforms</li> <li>- Expensive at large scale</li> <li>- Requires AWS expertise for optimisation</li> <li>- Partial ETL support (Requires integration with other AWS services for transformations)</li> </ul>
<a href="#">Azure Data Factory</a>	<ul style="list-style-type: none"> <li>- Integrates well with Microsoft Azure cloud services and tools</li> <li>- Easy-to-use drag-and-drop interface for orchestrating data flows</li> <li>- Supports a wide variety of sources and connectors</li> <li>- Full ETL support</li> </ul>	<ul style="list-style-type: none"> <li>- Limited support outside of the Azure ecosystem</li> <li>- Complexity in handling hybrid environments (on-prem and cloud)</li> <li>- Pricing can increase rapidly with large data volumes</li> </ul>
<a href="#">Google Cloud Dataflow</a>	<ul style="list-style-type: none"> <li>- Serverless, no-ops streaming data processing with built-in scaling</li> <li>- Strong support for both batch and stream processing</li> </ul>	<ul style="list-style-type: none"> <li>- Limited flexibility outside Google Cloud</li> <li>- Can get expensive for continuous large-scale data flows</li> </ul>



- |   |  |
|---|--|
| <ul style="list-style-type: none"> <li>- Integration with Google Cloud Storage and BigQuery for seamless ingestion</li> <li>- Full ETL support</li> </ul> | <ul style="list-style-type: none"> <li>- Requires specific expertise in Google Cloud Dataflow SDK</li> </ul> |
|---|--|

### 3.1.2. Storage Layer: Managing Structured and Unstructured Data

Once ingested, the **Storage Layer** ensures efficient handling of structured data (metadata) and unstructured data (video files, audio, images, text).

- **Storage Format:** The video files are stored as large binary objects in MinIO (S3-compatible object storage), while the metadata is stored in a NoSQL database like MongoDB for flexibility.
- **Semantic Mapping:** To facilitate ontology-driven processes, certain metadata fields (e.g., tags, locations, timestamps) are (automatically) mapped to predefined classes and properties in the **SCENE ontology**. This ensures that all multimedia assets are semantically annotated for later retrieval.

High-level candidate technologies for the storage layer have been described and compared in Section 2.3.1 (Minio, Ceph and Dreamio).

### 3.1.3. Discovery Layer: Search and Ontology Interaction

The **Discovery Layer** makes the ingested and stored data easily searchable. It leverages the ontology to enable **semantic search** capabilities.

- **Example Process:** A user (e.g., film producer) searches for footage tagged with "New York City," "sunset," and "action sequence." The system queries SCENE-O to identify multimedia files with these semantic tags.
- **Ontology Query:** Through an ontology-driven API, the discovery layer identifies relationships and properties within the metadata that match the query. For instance, the term "action sequence" may be mapped to several ontological concepts such as "fight scene" or "chase."

### 3.1.4. Analysis Layer: Supporting AI/ML Tools

The **Analysis Layer** integrates AI/ML tools to analyse the ingested multimedia data. This includes automatic tagging, scene classification, and content suggestions.

- **Example Process:** AI models analyse raw video footage to automatically classify scenes based on action, location, and mood. These classifications are then mapped to predefined SCENE-O classes (e.g., "urban area," "night-time," "intense emotion").
- **Enhancing Metadata:** Once classified, the metadata associated with each multimedia file is updated to include the AI-generated tags. These tags become searchable entities within the ontology, improving future retrieval.

Below are candidate technologies for distributed querying (including open-source and cloud-based approaches), shown in Table 19, and AI/ML Processing (e.g., automatic tagging), shown in Table 20. Some of the distributed querying solutions are also commercially presented as data warehouses.

*Table 19. Candidate technologies for distributed querying*

Technology	Pros	Cons
<a href="#">Trino</a>	<ul style="list-style-type: none"> <li>- Fast distributed querying of large datasets</li> </ul>	<ul style="list-style-type: none"> <li>- Requires significant setup for large clusters</li> </ul>



	<ul style="list-style-type: none"> <li>- Supports ANSI SQL</li> <li>- Open-source and widely adopted</li> <li>- Scalable with many connectors (Hadoop, S3, Kafka)</li> </ul>	<ul style="list-style-type: none"> <li>- May need external storage for large-scale data</li> <li>- Limited to SQL querying</li> </ul>
<a href="#"><u>ClickHouse</u></a>	<ul style="list-style-type: none"> <li>- Extremely fast for analytical queries</li> <li>- High-performance columnar storage engine</li> <li>- Suitable for real-time analytics</li> <li>- Open-source</li> </ul>	<ul style="list-style-type: none"> <li>- Limited support for unstructured data</li> <li>- Less flexible for ad-hoc querying</li> <li>- Requires optimisation for performance at scale</li> </ul>
<a href="#"><u>Apache Doris</u></a>	<ul style="list-style-type: none"> <li>- High concurrency and low-latency queries</li> <li>- Optimized for real-time analytics and data warehousing</li> <li>- Easy to use with SQL syntax</li> <li>- Highly scalable with MPP architecture</li> </ul>	<ul style="list-style-type: none"> <li>- Newer and less mature compared to ClickHouse/Trino</li> <li>- May require more manual tuning for complex workloads</li> <li>- Smaller ecosystem</li> </ul>
<a href="#"><u>Apache Druid</u></a>	<ul style="list-style-type: none"> <li>- Designed for real-time OLAP workloads</li> <li>- Fast ingestion and query speeds</li> <li>- Integrates well with streaming data</li> <li>- Highly scalable for time-series and log data</li> </ul>	<ul style="list-style-type: none"> <li>- Limited SQL support compared to traditional query engines</li> <li>- Complex configuration for certain workloads</li> <li>- Storage costs may increase with time-series data retention</li> </ul>
<a href="#"><u>BigQuery (Google Cloud)</u></a>	<ul style="list-style-type: none"> <li>- Fully managed, serverless, and scalable</li> <li>- Seamless integration with Google Cloud services</li> <li>- Automatic optimisations for performance</li> <li>- Great for ad-hoc queries and large datasets</li> </ul>	<ul style="list-style-type: none"> <li>- Expensive for very large queries</li> <li>- Limited flexibility for hybrid and multi-cloud setups</li> <li>- Vendor lock-in with Google Cloud</li> </ul>
<a href="#"><u>Amazon Athena</u></a>	<ul style="list-style-type: none"> <li>- Serverless, pay-per-query model</li> <li>- Integrates well with AWS data lakes (S3)</li> <li>- No infrastructure to manage</li> <li>- Works out-of-the-box with standard SQL</li> </ul>	<ul style="list-style-type: none"> <li>- Limited to querying data on AWS (S3)</li> <li>- Performance can be inconsistent on very large datasets</li> <li>- Costs can rise quickly depending on query complexity</li> </ul>

Table 20. AI/ML Processing Frameworks

Technology	Pros	Cons
<a href="#"><u>TensorFlow</u></a>	<ul style="list-style-type: none"> <li>- Extensive support for deep learning</li> <li>- Supports distributed training on large datasets</li> <li>- Strong for both research and production</li> <li>- Large ecosystem and community support</li> </ul>	<ul style="list-style-type: none"> <li>- Steep learning curve for beginners</li> <li>- More complex debugging compared to some alternatives</li> <li>- Heavier deployment requirements</li> </ul>
<a href="#"><u>PyTorch</u></a>	<ul style="list-style-type: none"> <li>- Easier to learn and use</li> <li>- Dynamic computation graph (intuitive for research)</li> <li>- Strong support for academic research</li> <li>- Integration with popular ML libraries</li> </ul>	<ul style="list-style-type: none"> <li>- Less mature deployment tooling than TensorFlow</li> <li>- Ecosystem is smaller than TensorFlow</li> <li>- Distributed training requires more manual work</li> </ul>



<p><a href="#"><u>Apache Spark (MLlib)</u></a></p>	<ul style="list-style-type: none"> <li>- Scalable across distributed clusters</li> <li>- Integrated with big data tools and ecosystems</li> <li>- Handles massive datasets efficiently</li> <li>- Well suited for large-scale data processing</li> </ul>	<ul style="list-style-type: none"> <li>- Higher latency compared to deep learning-specific tools</li> <li>- Limited deep learning capabilities</li> <li>- Requires configuration for high performance</li> </ul>
<p><a href="#"><u>Dask</u></a></p>	<ul style="list-style-type: none"> <li>- Scales well from single machines to clusters</li> <li>- Easy integration with pandas, NumPy, and scikit-learn</li> <li>- Great for parallel computing</li> <li>- Suitable for both data processing and ML tasks</li> </ul>	<ul style="list-style-type: none"> <li>- Less mature than Spark for large clusters</li> <li>- Requires manual optimisation for complex workflows</li> <li>- Fewer built-in ML libraries than other options</li> </ul>
<p><a href="#"><u>Scikit-learn</u></a></p>	<ul style="list-style-type: none"> <li>- Great for traditional ML algorithms (regression, classification, clustering)</li> <li>- Simple and intuitive API</li> <li>- Strong integration with NumPy, pandas, and other libraries</li> <li>- Good for small to medium datasets</li> </ul>	<ul style="list-style-type: none"> <li>- Limited support for deep learning</li> <li>- Doesn't scale as well for large datasets</li> <li>- Lacks built-in distributed computing features</li> </ul>

### 3.1.5. Conceptual end-to-end Workflow example: Integrating Data Lakes and Ontologies in SCENE

This section describes a tentative end-to-end workflow demonstrating how various SCENE architecture components, particularly the data lake and SCENE-O, work together to manage multimedia content. The workflow outlines the interaction between ingestion, storage, semantic annotation, and retrieval processes, showcasing the added value of ontology integration.

#### Example Scenario: Uploading and Managing Film Production Footage

Imagine a film production company that has completed shooting several scenes for an upcoming movie. The director uploads raw footage and associated metadata (e.g., location, time of day) into the SCENE platform. This scenario illustrates how the different layers of the SCENE architecture work together, utilising data lakes and ontologies to store, analyse, and semantically manage multimedia content.

#### Step 1: Ingestion of Raw Footage and Metadata

The **ingestion layer** begins by integrating new data into the system. The director uploads a batch of video files, along with metadata such as filming location, time of day, and the type of scenes (e.g., chase scene, dialogue).

- **ETL Process:** This raw data is extracted from the production company's local server, transformed into a format suitable for the data lake (e.g., MP4 for video files, JSON for metadata), and loaded into the system using the ETL pipeline managed by, e.g., *Apache NiFi*.
- **Metadata Enrichment:** During ingestion, additional metadata, such as GPS coordinates for locations or specific scene-related keywords, is automatically generated. This enriched metadata is prepared for semantic mapping.

#### Step 2: Storing and Structuring the Data

Once ingested, the multimedia data is passed to the **storage layer**, where it is categorised and stored for future use.

- **Multimedia Files:** Video files are stored as large objects in the S3-compatible object storage system (e.g., *MinIO*), efficiently storing and retrieving large unstructured files.



- **Metadata Management:** The metadata, including automatically generated tags like location and time, is stored in a NoSQL database such as *MongoDB* to ensure flexibility and scalability. This metadata is structured to comply with SCENE-O by mapping relevant fields (e.g., "Athens City" mapped to the ontology's geographic class) to ensure consistency across the platform.

### **Step 3: Semantic Mapping and Annotation**

As soon as the data is stored, it goes through an automatic semantic mapping process powered by the SCENE-O ontology. This step aligns metadata with the ontology, making the content searchable by its semantic attributes.

- **Ontology Mapping:** The metadata (e.g., location, type of scene) is aligned SCENE-O properties, such as mapping "chase scene" to the ontology class for "action sequences" and linking "Athens City" to the relevant geographic node.
- **AI-Assisted Annotation:** An AI tool integrated into the system analyses the raw footage to generate additional semantic tags automatically. For example, AI might identify objects in the scene (e.g., cars, buildings) or infer the mood (e.g., "high intensity"). These tags are added to the metadata and stored alongside the original files.
- **Manual Enrichment:** A production assistant can manually refine the metadata using a semantic annotation tool, adding more granular details like character names or specific emotions depicted in the scene. This process helps improve the quality and accuracy of the annotations, ensuring that the footage is well-categorized for future searches.

### **Step 4: Discovery and Semantic Search**

Once annotated, the data is now ready to be discovered by users through the **discovery layer**. This is where the value of semantic management becomes evident, as users can now search for specific types of content based on its semantic tags.

- **User Query:** A producer, for instance, searches for footage related to "night-time chase scenes" filmed in "urban settings." The system's search engine queries SCENE-O for these specific properties and retrieves all matching video files and related metadata.
- **Ontology-Driven Search:** The system can infer related terms thanks to the ontology's structure. For example, "night-time chase" might also retrieve footage tagged as "action scene at dusk," broadening the scope of search results. This flexibility makes the discovery layer more powerful than traditional keyword-based search engines.

### **Step 5: Analysis and Advanced AI Tools**

The **analysis layer** incorporates AI and machine learning tools to enhance content management further. In this scenario, the producer uses AI tools for automatic content classification and personalised recommendations based on previous search behaviour.

- **AI-Generated Insights:** The AI tools analyse the retrieved footage and suggest other relevant content, such as scenes with similar emotional tones (e.g., "tense" or "dramatic") or comparable action sequences. These recommendations are based on AI models that continuously learn from user preferences and behaviours.
- **Ontology Enrichment:** As the AI models classify more content, SCENE-O is dynamically updated to include new relationships and properties discovered through the analysis, making the system more robust and adaptive.

Throughout this workflow example, SCENE-O plays a central role in organising and semantically enhancing the multimedia content:

- **Metadata Alignment:** By mapping metadata to predefined SCENE-O classes, the system ensures that all multimedia assets are semantically structured, allowing for more accurate and flexible searches.

- **Semantic Search:** The discovery layer relies heavily on the ontology to provide rich, context-aware search results, which is impossible with traditional search engines.
- **Ontology Learning:** AI-driven processes continuously enrich the ontology by adding new concepts or refining existing relationships based on user interactions and AI-generated classifications.

This workflow can be expanded and adapted based on specific project needs:

1. **Handling Various Data Formats:** The workflow can easily be adjusted to handle different multimedia file formats such as 3D models, audio files, or scripts, each with its own metadata schema that can be mapped to SCENE-O.
2. **Collaborative Annotation Tools:** A manual annotation tool allows teams to work collaboratively on enriching metadata, ensuring more accurate and personalised tagging. These tools can provide taxonomies or predefined lists from the ontology, simplifying the process for end-users.
3. **Scalability:** The platform can scale to manage vast data. As new media files and metadata are ingested, the SCENE-O ontology will evolve to accommodate new categories and relationships, ensuring the data lake remains organised and semantically rich.

### 3.2. Component vision and interaction with other tasks

During the initial phases of the project, discussions among the partners of WP3 took place about the **interaction among data lakes, ontologies and tools implemented within WP3 and WP4**. A clear relationship was established with **T3.2** (ontology formulation mechanisms) and **T3.3** (Media Assets Manager). A diagram is depicted in Figure 15. and integrates the three different perspectives.

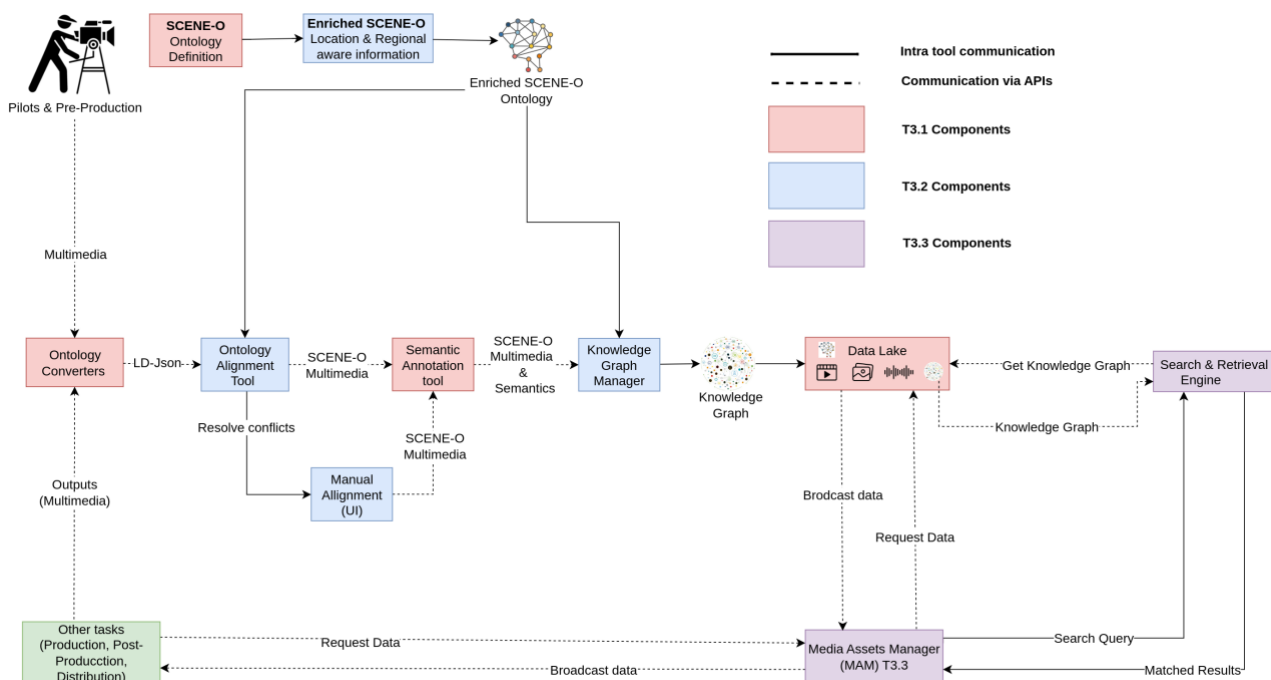


Figure 15. Interaction between T3.1, T3.2 and T3.3

#### T3.1 Perspective (red boxes)

To expand the SCENE-O ontology, ontologies available and described in section 5 were used. These ontologies and schemas were not always in the same format and aligned with the SCENE-O. Therefore, the tools implemented within T3.2 (D3.2) were used to convert the additional schemas and align the semantic entities



of each to compare them with the entities included in the SCENE-O ontology and extend it with the new entities included.

Depending on the format of multimedia files and considering the large variety of formats and standards, mappings are likely to be required among them. The input (multimedia) files do not follow a specific ontology. Still, the metadata (or part of it) can be mapped to SCENE-O properties of a given class through **ontology converters**.

The conversion intends to be automatic, but some intermediate (manual) processes might be needed. The end-user should be allowed to enrich the multimedia files with semantic tags through a **semantic annotation tool**. This manual process should present the user with a list (e.g., taxonomy of concepts from the ontology) to tag a given multimedia object.

The metadata information from the multimedia files is stored in the **data lake**, if possible, graphically (graph format). Otherwise, it should be able to generate a graph under request.

### T3.2 perspective (blue boxes)

From the T3.2 perspective, the ontology formulation mechanisms interact heavily with both the existing SCENE ontology (from T3.1) and the media content managed within the Media Asset Manager (T3.3). The primary focus of this task is to **enhance and expand** the initial SCENE-O developed in T3.1 by incorporating more contextual dimensions, particularly those related to location- and region-specific metadata. This expansion allows the ontology to support a broader range of data and use cases within the project.

The interactions in T3.2 primarily focus on taking the base SCENE ontology and extending it with **additional layers of semantic meaning**, enabling the system to understand regional and **location-aware information**. The ontology alignment also facilitates smooth integration between the metadata from T3.1 and the media content that will be organised and managed by T3.3. The Neuro-Symbolic AI allows for real-time adjustments and handling inconsistencies, ensuring the project's evolving knowledge graph remains coherent.

Regarding interaction with other tasks, T3.2 acts as a bridge between the ontology generated by T3.1 and the media management functionalities required by T3.3, ensuring that the knowledge graph integrates smoothly into practical applications.

### T3.3 perspective (purple boxes)

From the T3.3 perspective, the Media Asset Manager (MAM) is the interface for managing and utilising the videos generated or processed during the project. It uses SCENE-O and the expanded ontologies from T3.2 to facilitate efficient **content management, exploration, and utilisation**. This includes support for format-agnostic content ingestion, classification, and creation of collections or bundles of media assets.

The MAM incorporates **search and retrieval mechanisms** that leverage the aligned ontologies from T3.2, allowing for more accurate **content discovery based on semantic tags**, metadata, and ontology-driven relationships. The integration with the ontology developed in T3.1 (and enhanced in T3.2) enables advanced features such as non-linear navigation through content or navigating based on specific metadata parameters like location or audience preferences.

Furthermore, the MAM provides additional functionalities for **managing content licensing**, ensuring compliance with legal frameworks, and enabling monetisation options. The ability to trigger actions based on audience behaviours or specific metadata tags ties back into the ontology alignment efforts from T3.2. The MAM can offer users more personalised and contextually aware media experiences by connecting these different media assets to the underlying knowledge graph.

Regarding task interaction, T3.3 leverages the ontology infrastructure established in T3.1 and T3.2 to deliver a functional and innovative media asset management system that is flexible and scalable, ensuring a cohesive workflow across the project's content management phases.



### 3.3. Requirements

#### 3.3.1. Addressing End User Requirements

The feedback extracted from WP2, which involved the end-users, had a general scope and was irrelevant **to data lakes and ontologies**. In fact, during the first iteration of the end-user questionnaire in WP2, the SCENE consortium realised that technical aspects and questions were hard to understand from many end-users, and the second iteration tried to simplify the questionnaire, focussing on high-level aspects. Therefore, the details of using data lakes and ontologies were brought to a second level, as the end-users (producers, location scouts, distributors, etc.) could not distinguish among the different storage services and the added potential benefits of using combined versions of data lakes and ontologies. This is not a failure but a confirmation that, nowadays, technical tools should reach technical end-users dealing with related technical tools. To this respect, we mainly **changed the target end-users** to internal consortium partners responsible for all tasks in WP3 and WP4 (those who were developing technical tools for end-users); therefore, they should reflect through (technical) requirements how a data lake and/or an ontology can better help their tools to provide added value to the end-users. This process is reflected in the next subsection.

#### 3.3.2. Internal requirements

Feedback was gathered internally to check how both data lakes and ontologies (semantics) can benefit **the different tasks** within WP3 and WP4. Two **iterations** were made:

- The **first iteration** tried to gather **initial feedback** from the different tasks to check for different general functionalities that could be mapped to data lakes or ontologies. For the latter aspect, it was more practical to start using semantics, which is better understood even in technical realms. The questionnaire used is included in Annex 8.4.
- The **second iteration** tried to focus more **specifically** on the different functionalities offered by data lakes or ontologies and link them to the different tasks in terms of relevance. This allowed us to identify which **functionalities had more priority** and which were less important to the different tasks in WP3 and WP4. The questionnaire used was elaborated in the form of different tables to be filled by technical partners and is included in Annex 8.5.

Additionally, a round of **internal telcos** was performed individually with each technical partner to understand their needs better and explain the **potential usage** of data lakes and ontologies. A summary table for some tasks is given below.

*Table 21. Internal feedback from partners (mostly data lake-related)*

Task	Notes
<b>T3.3 Media Assets Manager (MAM)</b>	<ul style="list-style-type: none"> <li>- The needed inputs are video files and associated metadata. This can be either uploaded from a local folder or the data lake</li> <li>- MOG already has experience storing video files with S3 buckets (MinIO). It might be interesting to explore the idea of exposing a video URL for visualisation</li> <li>- The metadata can be either stored in SQL or no-SQL databases</li> <li>- The distribution engine (T4.6) does not include CDN intelligence</li> </ul>
<b>T3.3 UWB tool</b>	<ul style="list-style-type: none"> <li>- It is still unclear if the UWB will be used in the pilots currently; it seems rather unlikely</li> <li>- The tool can track the position of artists in a scene. It uses its own tracking format: position of actors, ID of wearable device, and timestamp information.</li> </ul>



	<ul style="list-style-type: none"> <li>- It is likely that a JSON format might be developed to associate video information with tracking (position) of the artist appearing in the different scenes. Some JSON examples and videos (if pilots are finally not going to use the tool) might be included in the data lake for testing purposes. Maybe the MAM might help in editing purposes</li> <li>- The tool might be used either in pre-production or post-production (e.g., include artificial 3D objects in the scenes and check the position of actors to determine foreground and background (what is hiding what))</li> </ul>
<p><b>T3.5 Blockchain</b></p>	<ul style="list-style-type: none"> <li>- The tool includes its own on-chain and off-chain storage system; thus, the data lake might seem unnecessary. Maybe we might agree on specific metadata (e.g., VideoMetadataHub model specific for rights)</li> <li>- Inputs come from the Media Assets Manager (MAM) and Audience Building Tool (Campaign Manager)</li> <li>- There are two stages for the contracts: the producer adds rights to media assets, and the user accepts the rights and a Ricardian contract is generated</li> <li>- MAM has its own database and, therefore, its own model for the rights (which might be discussed to follow some standard)</li> <li>- ABT might use some specific videos (e.g., behind the scenes) to generate campaigns and add some sort of NFT ownership as rewarding</li> </ul>
<p><b>T4.1 Location scouting</b></p>	<ul style="list-style-type: none"> <li>- The tool plans to use a database for locations including, among others (ID, country, city, description, and link to associated images/videos/3D objects)</li> <li>- The DB is internal to the tool (not part of the data lake)</li> <li>- Their API should be able to store/retrieve those images, videos and 3D objects in/from the data lake</li> <li>- Authorization would be required (only registered users may access data)</li> </ul>
<p><b>T4.3 Audience Building Tool (ABT)</b></p>	<ul style="list-style-type: none"> <li>- The tool is based on gamification algorithms, which offer activities for end-users to keep them engaged (by analysing data and creating posts on social media networks). Somehow connected to T4.2 (Audience Preference Scouting)</li> <li>- Input data comes from social networks. It is unclear if this info is stored internally in the tool or requires the data lake. The data is expected to be retrieved periodically (e.g., every day) but not in real-time mode</li> <li>- The output of this tool goes to T4.6 by direct feeding (API invocation). Check for potential usage of the data lake to store the content. Some output of T4.3 also goes to T3.5 (blockchain)</li> <li>- The used API retrieves data from the social network in Python (in case we need to store this info in the data lake, a Python library will be needed)</li> <li>- This tool will mainly store raw data and JSON files. For integration with the data lake, we need to list input data sources and retrieval frequency and size from the data sources to know the expected size</li> </ul>
<p><b>T4.4 Lighting Simulation module</b></p>	<ul style="list-style-type: none"> <li>- The simulation requires, from a specific location, images/videos, some configuration files for the model</li> <li>- The model is trained in real-time with this configuration file and images/videos and provides a visual output that does not need to be stored anywhere</li> <li>- There might be a shared repository for various users (same profile) working on the exact location</li> <li>- The API of the tool requires a way to find images/videos by location from the data lake. There should be a way to add those images/videos initially and increase their number later if this might enhance the results</li> <li>- Whenever an output is to be stored, it should potentially include metadata about the location</li> </ul>



#### T4.6 Recommendation engine

- The recommendation tool expects a set of inputs, which might be present in the data lake if used as a storage service. Those inputs refer to video scripts, audience information, which might come from the ABT, and external information (e.g., statistics and metrics from tools such as Netflix)
- The output is displayed in real-time, not stored. However, we might store the AI models (as a large binary file) or clusters of users (as a JSON file)
- Potential usage of CSV files as time series to be accommodated in the data lake

The feedback gathered from the internal consortium partners during the initial and second iterations of consultations has been crucial in refining the data lake and SCENE-O to meet the specific needs of the SCENE tools. The internal discussions across tasks such as **T3.3 (Media Asset Manager)**, **T3.5 (Blockchain Tool)**, and **T4.1 (Location Scouting Tool)** have informed how the architecture should adapt to better support media ingestion, content licensing, and semantic management. For example, the Media Asset Manager (MAM) provided valuable insights into the importance of seamless integration between video storage systems and metadata handling, particularly in leveraging **S3-compatible storage** (potentially MinIO) to support large video files while maintaining metadata consistency in NoSQL databases like MongoDB.

Additionally, the discussions around the blockchain tool (T3.5) revealed that while its primary focus is on managing digital rights and on-chain contracts, certain metadata could benefit from integration with the SCENE-O ontology to standardise rights and licenses across different tools. Similarly, feedback from the **Location Scouting Tool (T4.1)** has emphasised the need for an efficient API interface to store and retrieve location-specific media assets in the data lake, focusing on ensuring semantic enrichment for enhanced content discovery.

This internal feedback has directly influenced the prioritisation of features within the data lake and ontology architecture. By aligning the feedback with the overarching SCENE vision, these refinements ensure that the system meets the immediate technical needs of each task and supports scalability and adaptability for future end-user scenarios.

### 3.4. Platform Technology Overview: Components and Justifications

The data lake platform integrates a carefully selected technology stack optimised to meet the specific requirements of managing multimedia content, semantic annotation, and large-scale data storage. This section outlines the key technological choices for each architecture layer and explains the rationale behind these selections.

#### 3.4.1. Storage Layer

The **Storage Layer** of the SCENE platform is responsible for managing both structured and unstructured data. The primary tool chosen for this layer is **MinIO**, an S3-compatible object storage system.

- **Justification:** MinIO was selected for its ability to handle large-scale multimedia files (e.g., videos, images, audio) efficiently. Its compatibility with the S3 API ensures seamless integration with other tools and allows for scalable storage that can expand as the project grows. Moreover, MinIO's high-performance capabilities, especially with SSD-backed storage, ensure low latency for both read and write operations, which is critical for media-heavy applications like SCENE.

Additionally, MinIO offers the advantage of object-level security, ensuring that sensitive content can be protected according to data governance policies, including compliance with GDPR. The decision to use MinIO



also aligns with SCENE's goal of flexibility and scalability, as it supports various deployment options across cloud and on-premise environments.

### 3.4.2. Analysis Layer

The **Analysis Layer** enables the processing and querying of large datasets, supporting AI and ML tools that interact with the data lake.

- **Selected Tool:** **Trino** was chosen for distributed SQL query execution.
- **Justification:** Trino is a highly scalable SQL query engine that queries large datasets across data lakes, warehouses, and other distributed systems. Its ability to handle complex queries across various data sources, including the metadata and media stored in MinIO, makes it an ideal tool for the SCENE platform. Additionally, Trino's low-latency performance ensures fast data retrieval, which is essential when AI tools need to access data in real-time to generate semantic tags or recommendations.

By selecting Trino, the data lake architecture can support high-throughput analytical tasks and large-scale processing, ensuring that AI and machine learning tools can quickly access the data they need for advanced analysis.

### 3.4.3. Discovery Layer

The Discovery Layer of the Data Lake enables efficient search and retrieval of stored data, leveraging the tagging capabilities provided by the MinIO storage system. MinIO's built-in tagging mechanism allows for attaching user-defined metadata to objects. This foundational approach supports basic search functionalities, making data discoverable by tags such as content type, thematic categories, or production details.

While this method provides essential discoverability, it is a starting point for a more comprehensive semantic search framework envisioned under Task T3.2. The SCENE ontology will enhance this layer by enabling semantic tagging and ontology-driven interactions. This future integration will significantly improve the relevance and accuracy of search results by:

- Contextually enriching metadata using SCENE-O.
- Allowing for sophisticated, ontology-based querying methods.
- Supporting automated content classification and dynamic annotation tools.

This approach ensures that the Discovery Layer meets immediate project requirements while remaining scalable for advanced semantic features. By aligning with the broader objectives of Task T3.2, the Discovery Layer serves as a link between the data storage and user interaction layers, facilitating more intelligent and intuitive data access.

### 3.4.4. Ingestion Layer

The **Ingestion Layer** integrates external and internal data sources into the SCENE data lake, ensuring that media files and metadata are appropriately ingested.

- **Selected Tool:** **Apache NiFi** was selected for its ability to automate data flows and manage data ingestion pipelines.
- **Justification:** Apache NiFi was chosen for its robust, scalable, and flexible data ingestion capabilities. As a highly customisable data flow tool, it allows for the seamless extraction, transformation, and loading (ETL) of data from various sources. NiFi's graphical interface simplifies the creation of complex data flows, which makes it easier to manage the ingestion of different types of media content into the SCENE platform.



Its built-in support for real-time data flow management ensures multimedia content can be quickly and reliably ingested. NiFi also integrates well with both MinIO for storing media files and MongoDB for storing metadata, making it the ideal choice for ensuring the smooth transfer of data between external sources and the internal architecture of SCENE.

### 3.4.5. Operational Tools and AI/ML Integration

The data lake is designed with flexibility in mind, allowing developers to implement and deploy AI models using a variety of frameworks while ensuring that these models are seamlessly integrated into the system. This flexibility is managed through the **Operational Tool Framework**, which orchestrates and schedules AI tasks and workflows. The Operational Tool Framework supports any AI/ML model encapsulated within a Docker container, providing a standardised way to run, manage, and scale these models.

#### 3.4.5.1. Operational Tool Framework

The **Operational Tool Scheduler** is the backbone of the platform's workflow management, ensuring that all AI models and data processing tasks are properly orchestrated across different system components. The lightweight framework uses MongoDB to manage task metadata and is specifically optimised to handle concurrent tasks such as data ingestion, querying, and AI model execution.

- **Justification:** The SCENE project opted for a custom-built Operational Tool Framework to minimise resource usage while maintaining the flexibility to schedule complex workflows. This scheduler ensures that AI models, data processing tasks, and other operational components can be run in parallel without overwhelming the system's resources.

The system allows for high flexibility by encapsulating AI models within Docker containers. Any AI tool or framework, such as **TensorFlow**, **PyTorch**, or other libraries, can implement models if they follow the containerisation guidelines. This containerised approach ensures that models are easy to deploy, scale, and manage without introducing compatibility issues between systems or technologies.

#### 3.4.5.2. AI/ML Model Integration

The SCENE data lake integrates AI/ML models to perform tasks such as semantic tagging, scene classification, and content recommendations. The chosen frameworks for AI model development—**TensorFlow** and **PyTorch**—are suggested for their flexibility, scalability, and rich ecosystem support. Still, developers can implement models using other frameworks if the model is packaged within a Docker container.

- **Justification:** Docker containerisation ensures that models developed using TensorFlow, PyTorch, or any other AI/ML framework can be deployed in a standardised environment, regardless of the underlying toolset. This approach lets developers focus on model accuracy and performance while ensuring seamless integration with the **Operational Tool Scheduler**. The containerised models are scheduled and executed based on the system's workload, allowing for efficient resource management and high performance, even when handling large-scale multimedia data.

AI models are automatically managed using the Operational Tool Framework, including tasks like model execution, monitoring, and scaling. The system also ensures compatibility with the **Neuro-Symbolic AI** tools developed in other parts of the SCENE project (T3.2), providing a holistic and flexible environment for AI-driven content processing.

## 3.5. Proposed API Overview and Rationale

The SCENE platform leverages existing, well-established APIs from the underlying technologies, such as **MinIO's S3-compatible API** and **NiFi's built-in data flow management capabilities**, to minimise the need for custom development and ensure smooth integration with external systems. In cases where custom APIs are



necessary (e.g., for managing operational tools and AI models), SCENE implements lightweight, standardised **REST APIs** to ensure ease of use and flexibility.

### 3.5.1. Data Storage API (MinIO's S3-Compatible API)

The SCENE platform reuses the **S3-compatible API** provided by **MinIO** for managing multimedia content in the **Storage Layer**. This API allows the system to handle large unstructured data, such as video and audio files, without custom implementation.

- **Justification:** MinIO's native **S3 API** is widely used and supported, making creating a new API for object storage unnecessary. Using this standard API, SCENE can ensure compatibility with existing tools that rely on the S3 protocol (e.g., AWS-compatible tools) and can quickly scale storage operations without additional development overhead. This approach also ensures that existing backup, encryption, and data replication solutions in the S3 ecosystem can be easily adopted.

### 3.5.2. Data Ingestion API (NiFi Integration)

The data ingestion API plays a pivotal role in integrating various data sources into the SCENE platform. Traditionally, Apache NiFi has been used due to its robust capabilities for managing high-volume, complex data pipelines. To broaden the accessibility and flexibility of the ingestion layer, **Node-RED** can complement NiFi by providing a more user-friendly interface tailored for non-technical users and smaller, event-driven tasks.

#### Apache NiFi

Apache NiFi is ideal for handling large-scale ingestion tasks with complex data flows, supporting extensive transformation and routing capabilities. Its strengths include:

- **Scalability and Security:** Manages both batch and real-time data securely.
- **Flow-Based Programming:** Offers a visual programming interface suitable for complex data pipelines.
- **Data Lineage:** Ensures comprehensive data tracking and provenance, which is critical for auditing and data management at scale.

#### Node-RED Integration

Node-RED adds agility to the ingestion layer by enabling rapid prototyping and lightweight data flows. It is a flow-based tool similar to NiFi but is often preferred for its simplicity and accessibility for non-technical users. Node-RED is particularly well-suited for:

- **Event-Driven Workflows:** It facilitates API-based and device-triggered workflows, supporting quick integrations and interactive data flows.
- **Ease of Use:** With an intuitive drag-and-drop interface, Node-RED allows users to set up data ingestion processes without extensive technical knowledge. It is also easier to set up than NiFi.
- **Prototype Development:** Allows SCENE users to swiftly test and deploy small-scale workflows, which can later be transferred to NiFi for scaling.

#### Complementary Roles in the Ingestion Layer

The integration of Node-RED alongside NiFi offers a dual approach within the ingestion API, where:

- **Node-RED** serves as an accessible entry point for interactive, API-based data flows, making it easier for non-technical users to engage with the data ingestion layer.
- **Apache NiFi** supports advanced and scalable data processes, handling larger and more complex workflows that require strong lineage and security.

This combination within the SCENE data lake ensures a versatile and robust ingestion layer capable of supporting users across varying technical expertise and processing requirements.



### 3.5.3. Operational Tools API (Custom REST API)

The **Operational Tool Framework** includes a lightweight **REST API** for managing AI models. This API is designed to allow the **management** (execution) of AI models encapsulated in Docker containers and integrated into the SCENE platform's scheduling and orchestration framework.

- **Supported Methods:**
  - PUT for deploying new models (Docker containers) into the Operational Tool Framework.
  - DELETE for removing existing models.
  - GET to retrieve information on active or scheduled AI models.
  - POST to update a model
- **Justification:** While the existing Docker API allows container management at a low level, the **Operational Tools API** is necessary to provide higher-level management features specific to the SCENE project. This API abstracts away the complexity of container orchestration and ensures that AI models can be integrated into the SCENE ecosystem in a standardised manner. It also allows for easy scaling and monitoring of models without requiring users to interact directly with Docker.

### 3.5.4. Semantic Mapping and Ontology API

In general, the **Ontology API** reuses existing **SPARQL** query capabilities for semantic search and data retrieval, enabling interaction with the SCENE-O ontology. Relying on standard semantic web technologies like SPARQL and RDF, this API allows the platform to perform ontology-driven queries without custom semantic APIs.

- **Justification:** The supported RDF data formats ensure compatibility with existing ontology tools and services. This standardisation allows SCENE to integrate with other semantic web technologies and ontological systems seamlessly. Instead of reinventing query mechanisms, the platform utilises well-established methods for querying semantic data, ensuring robust and efficient interactions with the ontology.

### 3.5.5. External System Integration API

SCENE data lake primarily uses RESTful APIs and S3-compatible APIs for integration with external systems, leveraging the existing standards of **MinIO** and external media systems to transfer content and metadata.

- **Justification:** SCENE's reliance on existing RESTful APIs and the S3 protocol for external system integration ensures that third-party tools can easily interact with the platform without requiring extensive custom integration work. By adopting widely-used standards, SCENE can seamlessly connect to legacy systems, cloud platforms, and external media asset managers.

## 3.6. GDPR Compliance and Data Privacy Considerations

The data lake adheres to strict **GDPR compliance** and **data privacy regulations** to ensure that all personal data processed within the system is handled securely and ethically. The platform incorporates several key measures, including **data anonymisation**, **pseudonymisation**, and **encryption**, to safeguard sensitive information throughout its lifecycle. For instance, using robust encryption protocols, personal data included in multimedia content and associated metadata is protected during transmission and storage. Furthermore, the platform enforces **role-based access control** (RBAC) and secure API interactions, ensuring that only authorised personnel can access sensitive data.

In addition, the SCENE platform incorporates user-centric privacy measures, such as **informed consent** and mechanisms for exercising **data subject rights**. This ensures that individuals can access, correct, or request the deletion of their data following GDPR requirements. The platform also implements **automated data retention policies** to delete or anonymise personal data once it is no longer needed, reducing the risk of non-compliance.



For a more detailed discussion of SCENE’s privacy and data management strategies, including specific protocols for handling sensitive data and GDPR compliance, please refer to the project’s **Data Management Plan (DMP)**. The DMP outlines the full scope of data governance practices, security measures, and legal obligations that the platform follows, ensuring that all data processing activities meet the highest privacy standards.

### 3.7. Potential Challenges and Mitigations

As with any complex platform integrating multiple technologies, the data lake faces several potential challenges during its development and operation. These challenges span technical, operational, and compliance areas, but proactive mitigation strategies have been developed to address them.

#### 1. Ontology Alignment and Integration

**Challenge:** One of the core challenges in the SCENE platform is ensuring the successful alignment and integration of multiple ontologies from diverse sources. Given the wide variety of multimedia formats and metadata standards used in the film-making industry, there is a risk of **inconsistent or incomplete data mapping** to the SCENE-O ontology.

**Mitigation:** To address this, SCENE employs **Neuro-Symbolic AI** techniques that automate much of the ontology alignment process by recognising patterns in metadata and content. Manual refinement through semantic annotation tools also ensures that any edge cases or inconsistencies are handled. Regular testing of ontology converters will be conducted to ensure accurate mappings across all multimedia formats.

#### 2. Scalability and Performance Bottlenecks

**Challenge:** As the platform scales, particularly with ingesting large multimedia datasets (e.g., high-resolution videos), there may be **performance bottlenecks** in data processing, querying, and storage. High-demand queries or multiple simultaneous AI processes could strain system resources.

**Mitigation:** SCENE addresses this risk using **scalable cloud-based solutions** like **MinIO** for object storage and **Trino** for distributed querying. Additionally, Dockerized AI models ensure that tasks can be scaled horizontally, allowing the system to manage heavy loads more effectively. The platform also employs **load balancing** and **task scheduling** through the Operational Tool Framework to optimise resource use and prevent bottlenecks.

#### 3. Data Privacy and GDPR Compliance

**Challenge:** Ensuring **GDPR compliance** while processing large amounts of multimedia data, particularly if it includes personal information, is a significant legal challenge. Failure to handle personal data in line with privacy regulations could result in compliance issues and potential penalties.

**Mitigation:** The SCENE platform employs **data anonymisation**, **pseudonymisation**, and **encryption** techniques to mitigate the risk of handling sensitive personal data. It also automates key GDPR requirements, such as **data retention policies** and **user consent management**, ensuring that personal data is securely managed and deleted when no longer necessary. Furthermore, periodic audits and continuous monitoring ensure the platform complies with GDPR and other relevant data privacy regulations.

#### 4. Interoperability with External Systems

**Challenge:** The SCENE platform might interact with various external systems, including legacy media asset management systems, production tools, and distribution channels. Ensuring **interoperability** between these systems and the SCENE platform, particularly when handling different data formats and standards, presents a significant challenge.



**Mitigation:** To mitigate interoperability issues, the SCENE platform relies on a **modular API-based architecture** that allows external systems to interact with the platform using industry-standard protocols. The data lake also incorporates **data format converters** to handle diverse multimedia file types, and its scalable architecture ensures that new external systems can be integrated without significant disruptions.

## 5. AI Model Accuracy and Ethical Use

**Challenge:** While the SCENE platform heavily relies on AI for tasks, there is a risk that **AI models may produce inaccurate or biased results**, particularly when dealing with diverse cultural or contextual content. Additionally, ensuring the **ethical use of AI** models is a critical challenge in media management.

**Mitigation:** SCENE mitigates these risks by adopting **continuous training** based on real-world feedback and regularly updating its AI models with diverse datasets to reduce bias. Moreover, ethical guidelines are followed while developing and deploying AI tools, ensuring sensitive content is handled responsibly.

## 6. Long-Term Maintenance and Evolution of the Ontology

**Challenge:** Ontological concepts and relations in SCENE-O might change as new media formats, production techniques, and user needs emerge. There is a challenge in ensuring the **long-term maintenance and adaptability** of the ontology without disrupting the system's functionality.

**Mitigation:** SCENE-O has been designed flexibly, allowing for easy updates and extensions as new concepts and relationships are identified. The platform's **versioning system** ensures that changes to the ontology are managed systematically, with backwards compatibility maintained where necessary to avoid disruptions.

By identifying these potential challenges and implementing targeted mitigation strategies, the SCENE platform is well-positioned to manage risks and ensure the system remains robust, scalable, and compliant throughout its development and operational phases.

# 4. Exploration of film-related ontologies and Proposed Solution

This section briefly introduces the main concepts related to semantics and ontologies in the Film Industry (FI) sector. Ontology analysis involves studying existing ontologies in the film industry to understand how data is structured and related. This analysis will focus on identifying key ontologies and standards for describing film-related data, especially those using the JSON-LD standard. This section presents the film-related ontologies available in the literature, which will be used as a basis for the design of the SCENE-O ontology. Detailed information about the design of the SCENE-O ontology will be available in D3.2.

A basic knowledge of semantics and related vocabulary and standards is recommended to understand the work performed in this document entirely. The fundamental aspects of semantic concepts have been shifted to the end to facilitate the readership and focus on the ontology and the FI domain (see Annex 8.3). The user is encouraged to read them before continuing if unfamiliar with such terminology.

## 4.1. Film-related ontologies

A concise summary of key ontologies relevant to the film-making industry is outlined in Table 22. Each ontology is briefly described to provide an overview of its significance and application within film-related metadata and semantics. For readers interested in a more in-depth exploration, including specific examples and technical details, these are available in the annexes for further reading. This approach ensures a streamlined and accessible introduction while providing supplementary material for those who seek a deeper understanding of the individual ontologies.



Table 22. Key Ontologies and Semantics in the film-making Industry

Entity (ontology/project/ initiative)	Description
<a href="#">Schema.org</a>	<p>Schema.org is a collaborative community activity with a mission to create, maintain, and promote schemas for structured data on the Internet. It includes a comprehensive ontology for movies and related entities.</p> <p>For more information, see Annex 6.D</p>
<a href="#">EBUCorePlus</a>	<p>EBUCorePlus is a powerful extension of the original <a href="#">EBUCore</a> standard, merged with <a href="#">EBU CCM (Class Conceptual Model)</a>, both released in 2020, offering a more comprehensive model for managing audiovisual metadata. It includes descriptive, technical, and production-related metadata and supports a wide range of linked data technologies such as RDF, OWL, and JSON-LD</p>
<a href="#">Dublin Core</a>	<p>Dublin Core is a simple yet versatile metadata standard used widely across various domains, including libraries, museums, and archives. Its primary strength lies in its simplicity and interoperability, making it applicable for describing a wide range of resources, including audiovisual content such as films and TV shows. Dublin Core provides both a <b>Simple</b> and <b>Qualified</b> metadata set, which extends the basic core by adding refinements to elements. Dublin Core is highly compatible with <b>Linked Data</b> and <b>JSON-LD</b>.</p>
<a href="#">Ontology for Media Creation (OMC) from MovieLabs</a>	<p>The <b>Ontology for Media Creation (OMC)</b> developed by <b>MovieLabs</b> is a key framework for standardising the metadata associated with media production and creation. It provides an essential foundation for managing the media lifecycle, from pre-production and production to post-production, distribution, and archiving. The OMC is designed to integrate seamlessly with modern media workflows, especially as digital content creation becomes increasingly complex. OMC also includes the Ontology for Media Distribution (OMD) and the ontology for Creative Works.</p>
<a href="#">IPTC VIDEOMETADATAHUB</a>	<p>The <b>IPTC Video Metadata Hub (VMHub)</b> is a comprehensive framework developed by the <b>International Press Telecommunications Council (IPTC)</b> for managing video metadata across multiple standards. It aims to create a unified schema that allows video metadata to be exchanged seamlessly between different formats, such as <b>MPEG-7</b>, <b>Schema.org</b>, <b>Apple QuickTime</b>, and <b>EBU Core</b>. This standardisation simplifies workflows for content creators, distributors, and archivists, enabling easier search, annotation, and distribution of video content across various systems</p>
<a href="#">Core Ontology for Multimedia Annotation (COMM)</a>	<p>COMM was designed to overcome the limitations of existing multimedia standards, such as MPEG-7, by re-engineering its concepts using the DOLCE foundational ontology. COMM supports conceptual clarity and soundness, allowing extensibility in multimedia annotation tasks. Its primary goal is to comprehensively represent the structure and semantics of multimedia objects by decomposing media into fragments and applying semantic annotations. While COMM is based on widely recognised standards like DOLCE and MPEG-7, its lack of online support, broken links to the ontology, and insufficient community documentation make it difficult to implement in contemporary projects. Due to these challenges, SCENE will not consider COMM for further development in its framework</p>
<a href="#">Entertainment Identifier Registry (EIDR)</a>	<p><b>EIDR</b> is a globally recognised unique identifier system used primarily in the media and entertainment industry to track and manage audiovisual content. It is often compared to the publishing industry's identifiers like DOI (Digital Object Identifier). The <b>EIDR</b> system assigns unique IDs to audiovisual works, including</p>



movies, TV shows, episodes, and even trailers or edits. These IDs ensure content can be consistently referenced across platforms, systems, and workflows.

<p><a href="#"><u>OntoFilm</u></a></p>	<p>OntoFilm aims to model the processes and workflows in film production. The ontology addresses the three key phases of film production: <b>pre-production</b>, <b>on-set production</b>, and <b>post-production</b>. It conceptualises important elements such as movie scripts, scene planning, camera operations, filming conventions (like the 180-degree rule), and post-production workflows. OntoFilm's goal is to standardise how these processes are annotated, thus bridging the high-level creative input from directors and the low-level technical metadata generated throughout the production and post-production phases. However, despite this conceptual framework, the ontology lacks comprehensive online documentation or direct access to an OWL or RDF file, making it difficult to apply directly in contemporary projects like SCENE.</p>
<p><a href="#"><u>The DEEP FILM Access Project: Ontology and Metadata Design for Digital Film Production Assets</u></a></p>	<p><b>DFAP</b> (2014) aimed to manage and integrate the vast data sets generated by industrial digital film productions. By capturing automatically generated metadata (e.g., from camera equipment) and manually recorded documentation from film professionals, the project sought to create a unified, searchable, and contextually rich data set for film productions. The project used Sally Potter's 2012 film <i>Ginger &amp; Rosa</i> as a case study and aimed to establish methods for archiving and reusing these complex data sets. The goal was to enable deep metadata exploration of film assets, including scene descriptions, equipment used, personnel involved, and production decisions, by leveraging film-specific taxonomies and processes.</p> <p>However, despite its ambition, the DFAP lacks sustained support and detailed public documentation. No accessible ontologies or metadata frameworks from this project have been made available. Given the absence of accessible metadata models or continued development, DFAP cannot be considered for the SCENE project due to its lack of continuity and usable implementation resources.</p>
<p><a href="#"><u>YAGO2</u></a></p>	<p>YAGO2 [13] is a large-scale ontology that integrates information from sources like Wikipedia, WordNet, and GeoNames. It extends its data coverage to a wide range of entities, including films, to represent real-world knowledge. Film-related entities and their associated attributes, such as actors, directors, production dates, and genre classifications, are included. This makes it useful for applications like recommendation systems or semantic searches involving movies and entertainment. Moreover, YAGO2 provides temporal and spatial grounding for its entities, which is valuable for contextual and historical analysis of films and related media.</p>
<p><a href="#"><u>CAMO</u></a></p>	<p>The Context-Aware Movie Ontology (CAMO) [14] is a framework developed to represent movie-related concepts and relationships, emphasising contextual features comprehensively. CAMO integrates a broader range of representational and interactional features, such as story quality, direction, music, and visual effects, to support nuanced applications like recommendation systems. It encompasses data for 1,103 movies, creating an enriched knowledge base for the domain.</p>

Additional ontologies might also be relevant to consider, as they, while not specific to the film-making industry, provide valuable contributions in related areas. MPEG-7 plays a significant technical role in describing multimedia content. FOAF enables linking people and organisations, and PROV-O helps track data lineage and transformations, making these ontologies useful for various aspects of film production and management.



Table 23. Related ontologies linked with the FI industry

Ontology/thesauri	Description
<a href="#">MPEG-7 (Multimedia Content Description Interface)</a>	<p>MPEG-7 is an ISO/IEC standard for describing multimedia data content, including audio, video, images, and 3D models. It provides a standardised way to annotate and categorise multimedia, enabling efficient searching, indexing, and retrieval. In the context of the film-making industry, MPEG-7 is particularly useful for cataloguing large video archives, tagging scenes or frames, and providing detailed metadata about audio-visual elements such as soundtracks, special effects, and even scene transitions. By integrating MPEG-7 into a film production workflow, producers and archivists can improve the discoverability of assets, facilitate post-production tasks, and create more dynamic, metadata-driven editing and archiving systems. This makes MPEG-7 a relevant tool for managing technical aspects of film metadata in an interoperable and structured way.</p>
<a href="#">FOAF (Friend of a Friend)</a>	<p>FOAF (2014) is a decentralised ontology representing relationships between people, organisations, and projects. In the film-making industry, FOAF can be leveraged to create a social graph that links directors, actors, crew members, studios, and other stakeholders involved in the production process. By using FOAF, a production company can map out collaborations, track roles, and analyse professional networks across various projects. This can aid in identifying key personnel, ensuring cohesive collaboration across departments, and even managing reputation systems for hiring or contract work. Moreover, FOAF’s ability to integrate with other linked data frameworks, such as Schema.org, can enhance the metadata richness by combining professional networks with more traditional film-related data</p>
<a href="#">PROV-O (Provenance Ontology)</a>	<p>PROV-O (2013) is a W3C recommendation for representing provenance information, which includes the origins, transformations, and usage of data or physical assets. In film-making, provenance is critical for tracking the lifecycle of media assets, from raw footage through editing, distribution, and archiving. Using PROV-O, film producers can maintain a detailed history of every piece of content, including edits, effects, and metadata changes. This ensures the integrity and transparency of the production process and facilitates reuse and compliance with legal or contractual obligations. For example, provenance data could prove invaluable when verifying ownership rights or ensuring that content has been handled according to specific licensing terms. Additionally, PROV-O can be integrated into media asset management systems to track how digital assets evolve during collaborative editing or post-production workflows.</p>

Table 24 provides an overview of relevant thesauri and vocabularies commonly used in the cultural heritage domain, which may also offer useful references for certain aspects of film-making, such as set design, art direction, or location selection. While these resources are not directly related to the filmmaking process, their structured approach to describing cultural and geographic entities could provide valuable insights. Given their peripheral relevance to the project, no further details or annexes are provided, but their inclusion highlights potential cross-domain synergies.

Table 24. Related thesauri and vocabulary from the cultural heritage perspective

Ontology/thesauri	Description
<a href="#">Getty Thesaurus of Geographic Names (TGN)</a>	<p>TGN provides standardised names and associated information for places important to studying art, culture, and history, including filming locations. TGN may be linked to GIS (Geographical Information System), maps, and other</p>



	geographic resources. The TGN hierarchy can be <a href="#">browsed</a> and displayed in a semantic view.
<a href="#">Getty Vocabulary Program</a>	The Getty Vocabulary Program includes several thesauri related to art and cultural heritage. Terms from these thesauri are often used to describe aspects of films, especially in the context of art direction, set design, and historical accuracy. It includes AAT (Art & Architecture Thesaurus), TGN (Getty Thesaurus of Geographic Names), ULAN (Union List of Artist Names), IA (Iconography Authority), and CONA (Cultural Objects Name Authority). <a href="https://www.getty.edu/research/tools/vocabularies/vocab_what_we_do.pdf">https://www.getty.edu/research/tools/vocabularies/vocab_what_we_do.pdf</a> <a href="https://www.getty.edu/research/tools/vocabularies/">https://www.getty.edu/research/tools/vocabularies/</a>
<a href="#">CIDOC Conceptual Reference Model (CIDOC CRM)</a>	The CIDOC CRM is an ontology developed to facilitate the integration, mediation, and interchange of cultural heritage information. It provides a robust framework for describing the relationships between cultural artifacts, events, and actors. CIDOC CRM could help structure and manage complex metadata regarding props, set designs, and historically accurate costumes in film-making, particularly in projects involving historical or culturally significant content. Currently, there are updated specifications in PDF format, but no related RDF/OWL/N3 file is provided (the latest review in Linked Open Vocabularies- <a href="#">LOV</a> is from 2016)
<a href="#">Europeana Data Model (EDM)</a>	The Europeana Data Model integrates and structures data about Europe’s cultural heritage collections. EDM can help integrate metadata from various cultural institutions for films focusing on European cultural history or heritage. By aligning film metadata with EDM, production teams can link their projects to larger European heritage databases, creating richer contextual connections. EDM is based on Dublin Core, SKOS and CIDOC-CRM. <b>Europeana Pro</b> refers to the platform and portal that provides professional resources, documentation, and tools related to Europeana's digital infrastructure, including the EDM.

In conclusion, this section has presented a comprehensive state-of-the-art review of existing film-related ontologies and standards, which aligns with the first objective of **Task T3.1**: identifying and analysing available ontologies in the film-making industry. The ontologies and vocabularies discussed, such as Schema.org, EBUCorePlus, Dublin Core, and the Ontology for Media Creation (OMC), are central to structuring and managing film-related data. By exploring their relevance and applicability to the SCENE project, this section lays a solid foundation for the subsequent development of **SCENE-O**, a scalable and agnostic ontology tailored to meet the specific needs of the film-making industry. Additionally, by including standards like JSON-LD, which ensure linked data compliance, the ontologies reviewed will facilitate dynamic tagging and advanced semantic techniques for film content management.

While this section focuses on the state-of-the-art and identifying potential candidates for consideration, the more technical aspects of **SCENE-O's development**, such as evaluating project-specific requirements, semantic clustering of knowledge, and creating annotation tools, will be addressed in later chapters. These future sections will integrate the proposed ontologies, evaluate the syntactic and semantic characteristics of the SCENE data sources, and outline the roadmap for implementing SCENE-O within the project's pilots. This structured approach ensures that all facets of Task T3.1, including ontology development and deployment, are systematically addressed as the document progresses.

## 4.2. Proposed Ontology: SCENE-O

The SCENE ontology designed within the SCENE project aims to capture film-related entities and their relations in order to facilitate search and information retrieval from the Data Lake implemented within the project and provide them to the SCENE platform’s tools. The following figure (Figure 16) presents the visualisation of the ontology as it is available via the Webvowl tool.

The SCENE-O ontology will be used as a baseline concept included in the already available literature film-related ontologies presented in section 4.1, while it will be extended with additional entities and relations resulting from the requirements of the SCENE’s platform’s tools. The ontologies whose entities and relations are included in the SCENE-O ontology are the following:

- Schema.org
- Ontology for Media Creation (OMC) from MovieLabs
- OntoFilm
- MPEG-7
- YAGO
- Core Ontology for Multimedia Annotation (COMM)
- CAMO

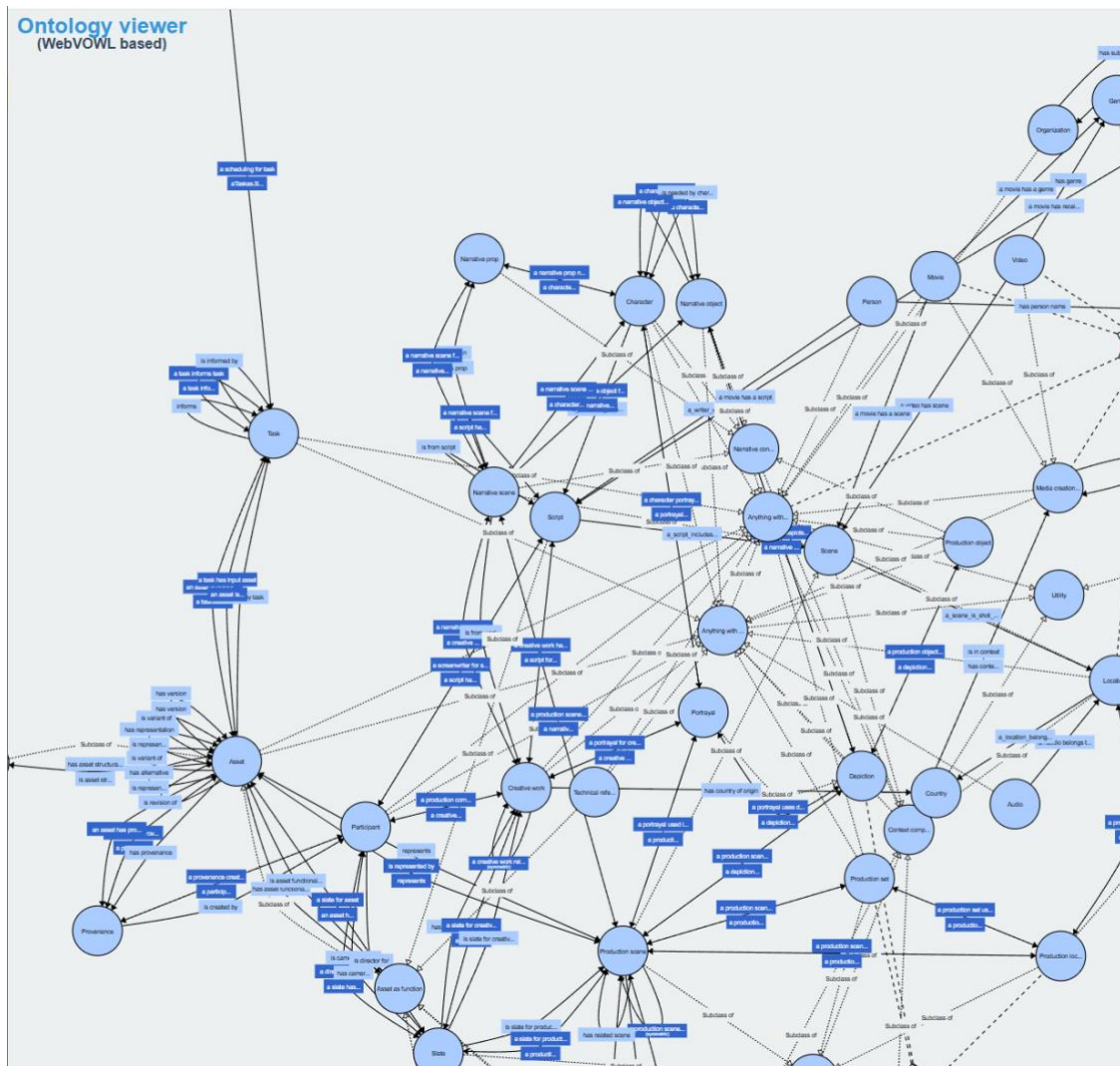


Figure 16. SCENE ontology displayed in the ontology viewer

The following table (Table 25) presents some important entities in the SCENE-O ontology, which is re-used from the aforementioned ontologies. A detailed presentation of the SCENE-O ontologies entities and their creation is presented in D3.2.

Table 25. Important entities re-used from film-related ontologies and included in SCENE-O

Entity	Definition and Key Properties
Movie	<p><b>Definition:</b> A work of visual storytelling, usually produced for cinema, TV, or streaming platforms</p> <ul style="list-style-type: none"> <li>• <b>name:</b> The movie's title.</li> <li>• <b>director:</b> The director of the movie (Person entity).</li> <li>• <b>actor:</b> Actors who appeared in the movie (Person entity).</li> <li>• <b>producer:</b> Producers of the movie.</li> <li>• <b>genre:</b> The genre of the movie (e.g., "Action", "Comedy").</li> <li>• <b>datePublished:</b> The release date of the movie.</li> <li>• <b>duration:</b> The movie's duration, typically in ISO 8601 format.</li> <li>• <b>productionCompany:</b> The production company responsible for the film.</li> <li>• <b>aggregateRating:</b> Aggregated user ratings for the movie (AggregateRating entity).</li> <li>• <b>location:</b> The location where the movie has been shot.</li> <li>• <b>plot:</b> A brief summary or description of the movie's storyline.</li> </ul>
Organization	<p><b>Definition:</b> A legal entity representing a production company, distributor, or studio</p> <ul style="list-style-type: none"> <li>• <b>name:</b> The organisation's name.</li> <li>• <b>location:</b> The geographical location of the organisation.</li> <li>• <b>foundingDate:</b> The date the organisation was founded.</li> <li>• <b>sameAs:</b> Links to official websites or external sources (e.g., IMDb or Wikipedia).</li> <li>• <b>logo:</b> A logo image for the organisation.</li> </ul>
Asset	<p><b>Definition:</b> Represents a physical or digital object specific to the creation of the creative work (e.g., video files, images)</p> <ul style="list-style-type: none"> <li>• <b>identifier:</b> Unique identifier for the asset</li> <li>• <b>name:</b> Human-readable name for the asset</li> <li>• <b>description:</b> A human-readable description of the asset</li> <li>• <b>version:</b> Information about the version of the asset</li> <li>• <b>provenance:</b> Tracks the origin or history of the asset</li> </ul>
Character	<p><b>Definition:</b> Represents a sentient entity in the script whose specific identity is important to the narrative (e.g., actors)</p> <ul style="list-style-type: none"> <li>• <b>identifier:</b> Unique identifier for the character</li> <li>• <b>name:</b> Primary name used for the character</li> <li>• <b>description:</b> A brief description of the character</li> <li>• <b>profile:</b> Physical and background characteristics of the character</li> </ul>
Actor	<p><b>Definition:</b> The actors participating in a movie</p> <ul style="list-style-type: none"> <li>• <b>name:</b> The actor's name.</li> <li>• <b>acted in:</b> A property linking actors to the movies they appeared in.</li> <li>• <b>birth/death Dates:</b> Temporal data providing lifespan information, which is especially useful for historical contexts.</li> </ul>
User Review	<p><b>Definition:</b> The reviews provided by the users about a movie.</p> <ul style="list-style-type: none"> <li>• <b>User ID:</b> A unique identifier for the user.</li> <li>• <b>Rating:</b> The rating given by the user, often on a scale from 1 to 5 or 1 to 10.</li> <li>• <b>Sentiment Polarity:</b> The sentiment score of the review, indicating whether the review is positive, neutral, or negative.</li> <li>• <b>Text:</b> The content of the review, including subjective comments on various aspects of the movie.</li> </ul>



The SCENE-O ontology will be leveraged by the corresponding SCENE's tools (i.e., Media Asset Manager, Audio simulation engine, Location scouting, Lighting simulation, Audience Building tool, Chatbot) for querying information from the data lake, through the use of the Data Lake's tools and the semantic layer.

This layer aims to enhance the ontology's functionality by providing meaningful context to raw data, ensuring consistent definitions across datasets, simplifying querying, improving data governance, and enabling faster insights. While a data lake stores vast amounts of raw, unstructured, or semi-structured data, a semantic layer helps interpret this data by providing meaningful definitions and relationships. It enables users to query data not just by tables or columns but through conceptual entities defined in the ontology. This context allows users to understand how data relates, making it more accessible. For example, if the user wants to find all movies directed by Christopher Nolan, they will be able to write in the Chatbot available in the SCENE platform a query in natural language that the semantic layer will interpret by using the Director concept in the SCENE ontology, and it can fetch the appropriate movie entries, while it is not needed by the user to manually identify where the director data resides.

## 5. Exploration of ontology design and related tools

This section aims to present a literature review on the ontology design methods and the related tools implemented within T3.1 and T3.2. The literature review is the first step conducted before the implementation of an ontology, identifying the relevant concepts available in literature, methodologies and tools that could be extended or enhanced during the SCENE project and the extension of the SCENE-O ontology. Information about the design of the corresponding tools implemented within the SCENE project and the SCENE ontology is available in D3.2.

To enable knowledge-driven applications, ontology conversion and alignment tools play complementary roles. Ontology conversion ensures that data can be accurately translated across different ontology formats, preserving the intended meanings while facilitating seamless integration. At the same time, knowledge graphs (KGs) capture complex relationships among entities and their attributes, providing a structured way to represent real-world information. Many advanced applications, such as recommendation systems, semantic search, and question-answering, rely on KGs for relevant, context-rich data. However, no single KG is comprehensive enough to support these applications with all necessary information, making entity alignment across different KGs essential. By combining ontology conversion with entity alignment, organisations can bridge diverse KGs, enriching the information available for more accurate and intelligent data-driven solutions.

Within the SCENE project, the SCENE ontology has been designed, reusing parts of the available film-related ontologies described in section 4, based on the requirements of the project and on the main entities and relations included in them. In addition to the SCENE-O ontology, alignment and conversion tools will be implemented to integrate knowledge in a single knowledge graph. More specifically, the ontology conversion tool will allow the introduction of ontologies from different formats into the SCENE-O format before their alignment. Different ontologies may represent similar concepts in varying formats or terminologies; therefore, the alignment part will expand the SCENE ontology since the newly introduced ontologies might include entities and relations that are either unavailable in the SCENE ontology or not included with the same terms.

This section presents the literature review on the ontology conversion and alignment tools available in the literature. The implementation of these tools will be presented in D3.2, and a comparative analysis will be conducted.



## 5.1. Ontology and Related Tools Specification

Two tools have been used to design and create the SCENE-O ontology: (a) Protégé, and (b) WebViewOWL. Each tool has been used for a specific task, which is for (a) the design of the ontology and for (b) the visualisation of the ontology created. The following table (Table 26) presents the tools available for the design and visualisation of the ontology, providing information about them to the reader and explaining how the two aforementioned tools were used during the design of the SCENE-O ontology.

Table 26. Reference terminology in the data management domain

Tool	Description
<p><a href="#">Protégé</a></p>	<p>Protégé is a free, open-source platform developed by Stanford University for creating, editing, and managing ontologies. It provides a user-friendly graphical interface, allowing novice and expert users to construct domain models and knowledge-based applications using ontologies. Protégé supports various ontology languages, including the Web Ontology Language (OWL) and Resource Description Framework Schema (RDFS), making it a versatile tool for semantic web development and knowledge representation.</p> <p>Protégé provides a complete set of capabilities for successful ontology design. Its key features include a class hierarchy editor for designing and organising concepts and a property editor for expressing connections between classes and persons. The tool supports OWL and RDF, allowing users to build semantically rich ontologies. The graphical user interface of Protégé allows for simple modification of ontology pieces using drag-and-drop capabilities.</p> <p>Protégé’s accessibility, adaptability, and broad feature set may lead someone to utilise it for ontology creation. Protégé, a free, open-source tool with an easy-to-use interface, reduces the entrance barrier for people and organisations seeking to construct and maintain ontologies without making a large financial commitment. Its support for different ontology languages, such as OWL and RDFS, makes it appropriate for various applications, from simple taxonomies to large knowledge representations. Protégé’s flexibility via plugins enables users to modify the tool to their individual requirements, while its visualisation features aid in understanding and conveying complicated ontological systems. The built-in reasoning engines allow users to evaluate their ontologies and infer new information, improving the generated models’ quality and utility. Furthermore, Protégé’s collaborative capabilities enable team-based ontology construction, making it an ideal tool for both individual researchers and large-scale initiatives in academia and business. The vibrant community around Protégé also offers a multitude of information, tutorials, and help, making it simpler for newbies to learn and experienced users to tackle complicated issues.</p>
<p><a href="#">OWLAPI</a></p>	<p>The OWLAPI is a robust Java library that creates, manipulates, and serialises OWL (Web Ontology Language) ontologies. OWLAPI, created to support the most recent OWL 2 specification, offers a complete collection of tools and interfaces that allow developers and knowledge engineers to interact with ontologies programmatically. It allows you to import and save ontologies in multiple forms, such as RDF/XML, OWL/XML, and Turtle, as well as execute reasoning tasks and query ontological data.</p> <p>OWLAPI offers a wide set of features that can enhance the ontology design process, including:</p> <ul style="list-style-type: none"> <li>• <b>Ontology I/O:</b> Support for loading and saving ontologies in various formats (RDF/XML, OWL/XML, Turtle, etc.)</li> <li>• <b>Axiom manipulation:</b> Tools for adding, removing, and modifying axioms within the ontology</li> <li>• <b>Entity management:</b> Creation and manipulation of classes, properties, and individuals</li> </ul>



- **Reasoning support:** Integration with popular reasoners for consistency checking, classification, and inference
- **Change tracking:** Mechanisms to monitor and manage ontology changes over time
- **Modularization:** Tools for extracting and managing ontology modules
- **Merging and comparison:** Features for combining ontologies and identifying differences between versions
- **Annotation support:** Ability to add and manage metadata annotations
- **Query capabilities:** Interfaces for querying ontological data
- **Profiles:** Support for OWL 2 profiles (EL, QL, RL) for different expressivity needs
- **Validation:** Tools for checking ontology consistency and adherence to OWL 2 specifications

The extension and flexibility of OWLAPI are among its main advantages. Users may customise its interfaces, opening the door to creating ontology management-specific tools and applications. Moreover, the API has integrated reasoner interfaces that enable well-known reasoning engines like Hermit and Pellet. These reasoning engines may be used to categorise ideas, verify ontology coherence, and deduce new information. Furthermore, OWLAPI offers tools for manipulating axioms and managing changes, making it a vital resource for ontology builders who must maintain and develop intricate knowledge systems over time

### WebviewOWL

A web-based application called WebviewOWL made it easier to create, visualise, and amend ontologies using the Web Ontology Language (OWL). With the use of this tool's user-friendly graphical user interface, knowledge engineers, domain experts, and ontology designers may deal with intricate ontological structures without needing to be well-versed in OWL syntax. WebviewOWL provides a user-friendly browser environment for various ontology development activities, such as class hierarchy building, property declaration, and instance population.

Because of its accessibility and user-friendly interface, WebviewOWL may be chosen for ontology visualisation since it reduces the entry barrier for individuals unfamiliar with ontology development. Because the tool is web-based, it doesn't require complicated program installs, making it perfect for collaborative projects and rapid starts. Users may benefit from its ability to visually display ontology structures, facilitating communication and understanding of intricate links within their area of interest. WebviewOWL's collaborative capabilities allow teams working on knowledge representation projects to collaborate in real-time, which might expedite the ontology creation process. Its compatibility with well-known ontology tools and reasoners further enables users to use sophisticated features without compromising usability, making it a flexible option suitable for novice and expert ontology builders.

The ability of WebviewOWL to produce interactive ontology structure visualisations is one of its primary benefits since it makes it simple for users to understand the connections between concepts and attributes. Large-scale ontology creation projects benefit greatly from the tool's collaborative features, which enable numerous users to work on the same ontology at once

### TopBraid Composer

TopBraid Composer is a tool for developing, managing, and testing ontologies. It provides a comprehensive environment for working with Semantic Web standards such as RDF, RDFS, OWL, and SPARQL. The tool is used for modelling and editing ontologies, which are essential in enabling data interoperability, classification, and search across systems. Its graphical user interface allows users to visualise relationships between classes, properties, and individuals, making designing and navigating complex ontologies easier. In addition to ontology editing, TopBraid Composer supports advanced features like inference, rules-based reasoning, and integration with various data sources, enhancing its utility for professionals in fields like data science, information management, and



enterprise architecture. It also offers advanced reasoning capabilities, allowing users to validate their models and ensure logical consistency. With features like integration with data sources and support for SHACL constraints, TopBraid Composer is a robust solution for building interoperable, scalable knowledge systems.

The flexibility and teamwork support of TopBraid Composer are among its main advantages. Users may work together across teams on large-scale projects and easily exchange their ontology models. It allows for processes that cover the whole ontology and data management lifecycle, from creation to deployment, through integration with other [TopQuadrant](#) technologies. Additionally, the tool allows users to run SPARQL queries right within the interface, allowing them to improve their ontologies in real-time, depending on user input. TopBraid Composer's feature-rich design has made it the preferred choice for businesses wishing to use ontologies and semantic technologies to improve data integration, knowledge management, and decision-making

## 5.2. Exploration of ontology conversion tools

Ontologies are frameworks that organise knowledge in specific fields and help structure information to be understood and used consistently. However, different groups often create unique ontologies based on their specific needs, making it difficult to share and combine information across different systems.

The literature on ontology conversion and alignment tools has highlighted their critical role in ensuring interoperability and efficient data integration across various domains, particularly in fields that rely on complex data interactions like healthcare, artificial intelligence, and the semantic web [15] [16]. These tools enable automated or semi-automated alignment of different ontologies, facilitating data translation and integration without requiring extensive manual intervention. For example, an ontology can include the entity creative work and a second one the entity asset, while both have the same definition. Studies have classified alignment approaches into categories based on the type of input, such as schema-based, instance-based, and usage-based alignment, each with strengths suited to specific data contexts. Schema-based approaches are widely used in artificial intelligence and semantic web applications, which help querying, reasoning, and data integration.

The tools available in the literature for ontology conversion are presented in the following table (Table 27).

*Table 27. Summary of ontology conversion tools*

Tool	Description
<b>Protégé</b>	A widely-used open-source ontology editor and knowledge-base framework primarily supporting OWL ontologies. It can be used for developing, visualising, and converting ontologies between various formats, such as OWL and RDF/XML. Its extensibility allows for integration with plugins for additional functionalities like ontology conversion
<b>ROntology Build and Transform</b>	ROBOT [17] is a command-line tool for automating ontology workflows, including conversion between formats. ROBOT supports several formats, such as RDF/XML, Turtle, and Manchester Syntax, making it useful for processing and converting ontologies
<b><a href="#">Ontology Converter</a></b>	A JVM-based command-line utility to transform RDF-based ontologies into OWL2-DL format. It works with individual ontology files or directories of multiple documents, automatically generating required OWL structures. Supported formats include Turtle, RDF/XML, JSON-LD, and Manchester Syntax
<b>OBO-Format Tools</b>	The OBO (Open Biomedical Ontologies) [18] format tools facilitate ontology creation, conversion, and management in the OBO format, which is commonly used in biological and biomedical ontologies. Here are some key points about



	tools like ROBOT (a command-line ontology toolkit) that can convert OBO format to various formats such as RDF/XML, OWL, JSON, or Turtle. It supports bidirectional conversion, allowing for seamless integration of OBO data into broader ontology workflows. It enforces strict document structure rules to maintain consistency but provides options to bypass strict checks when necessary. It also enables compliance with OBO Foundry principles, including standardised naming conventions, versioning, and documentation. They ensure that ontologies are sharable and reusable across diverse applications.
<b>OWL API</b>	A Java-based library and toolkit for manipulating OWL ontologies, including conversion to and from various OWL serialisation formats. It is suitable for developers working programmatically on ontology transformation tasks

### 5.3. Exploration of ontology alignment tools

Ontology alignment, on the other hand, involves finding matches between similar elements in different ontologies, such as classes or properties, so that information can be integrated and used across systems. Both processes can be complex because different ontologies often have very different structures and levels of detail. Over the years, many tools and techniques have been created to help with these tasks, from rule-based systems to machine-learning approaches. Exploring these tools and understanding how they work can shed light on their strengths, weaknesses, and the future of ontology alignment and conversion research.

In terms of tool evaluation, various tools have been developed to perform ontology matching and alignment, with notable research progress in automation. For example, tools like Falcon-AO [19], LogMap [20], and AML (AgreementMakerLight) [21] [22] have shown robust performance across different domains by combining linguistic, structural, and machine learning-based matching methods. These tools have been evaluated through benchmarks such as the [Ontology Alignment Evaluation Initiative](#) (OAEI), which offers structured datasets for testing alignment accuracy and effectiveness. However, challenges remain, especially in complex scenarios where single-entity alignment is insufficient and more nuanced, multi-layered mappings are required. Researchers are also exploring advanced algorithms incorporating user validation and crowd-sourced assessments to improve alignment quality further, as manual validation remains crucial in high-stakes or domain-specific applications like healthcare.

Knowledge graph alignment is the foundation for integrating multiple KGs, ensuring consistency, and enhancing the utility of linked data. It plays a fundamental role in KG construction, fusion, and many downstream applications mentioned above. Below are five significant methods for knowledge graph alignment: BERT-INT, CG-MuAlign, EMGCN, MultiKE, and DAAKG. These methods represent a range of supervised, unsupervised, and semi-supervised approaches, each leveraging different aspects of knowledge graphs such as entity attributes, relationships, and neighbourhood information.

The tools available in the literature for ontology alignment are presented in the following table (Table 28).

Table 28. Summary of ontology alignment tools

Tool	Description
<b>BERT-INT</b>	BERT-INT [23] utilises a multilingual BERT ( <i>Bidirectional Encoder Representations from Transformers</i> ) model to embed the side information (entity names, descriptions, and attributes) rather than relying on graph structures like traditional methods. This approach allows for fine-grained semantic matching between entities by directly computing interactions between their BERT embeddings. In simple words, it computes embeddings for all entities' properties, computes different interactions between them (Instead of aggregating neighbour



information through graph neural networks), and then combines their similarity vectors to get the final similarity score of 2 entities by even taking into consideration multi-hop neighbours' interactions. The model is supervised and trained on pre-aligned entity pairs only in the embedding stage. The ability to generalise to unseen data sets makes BERT-INT a robust and versatile method for knowledge graph alignment, particularly in cross-lingual contexts. However, while BERT-INT can theoretically be trained on one dataset and applied to another, its accuracy may be slightly lower in practice than methods specifically optimised for each dataset.

<b>CG-MuAlign</b>	CG-MuAlign [24] aligns multiple types of entities across knowledge graphs by collectively leveraging attribute and neighbourhood information. First, for preprocessing, fastText [25] is used to encode string features and numerical features to keep the original value except for the time values. It employs a combination of cross-graph and self-attention mechanisms to handle data incompleteness, to aggregate efficiently positive neighbourhood information, and to remain also sensitive to strong negative evidence to distinguish similar but different entities. The model includes two GNN (Graph Neural Network) encoders and a loss layer, where each GNN aggregates neighbourhood information in a multi-layer approach, effectively combining attribute and neighbourhood data with the mentioned attention mechanisms. This supervised method is noted for its fast-training process and scalability, performing 20 times faster than other deep learning methods.
<b>EMGCN</b>	EMGCN [26] offers an unsupervised approach to KG alignment, distinguishing itself by not requiring prior knowledge. The method leverages machine translation to handle cross-lingual attribute information. It employs an iterative refinement process to mitigate noise, making it a powerful tool in scenarios where labelled data is scarce or unavailable. EMGCN first builds the relation networks of the input KGs, translates the names of entities to English, performs word embedding, and then uses multi-layer GCN (Graph Convolutional Networks) embedding with alignment matrices, integrating relational and attribute-value similarities to produce the final alignment matrix. The GCN model consists of several layers, each encoding network topology in multiple orders.
<b>MultiKE</b>	MultiKE [27] is a flexible framework for learning comprehensive entity embeddings in both supervised and unsupervised modes. It utilises three representative views—name, relation, and attribute—to create detailed entity representations. The name view uses pre-trained word or character embeddings with an encoder, the relation view employs the TransE model, and the attribute view uses convolutional neural networks (CNNs). Cross-KG identity inference uses seed alignments for entities and soft alignments for relations and attributes based on name and semantic similarities. The model is trained using entity seed pairs to guide the embedding process, ensuring that embeddings accurately reflect cross-KG identity inference. MultiKE combines these view-specific embeddings through weighted averaging, shared space learning, or in-training combinations for the final result, ensuring high-quality representations. Despite its high performance, MultiKE faces challenges in scaling up to very large datasets.
<b>DAAKG</b>	DAAKG [28] combines deep learning with active learning to align entities and schemata across KGs. This supervised method relies on pre-existing aligned pairs and human annotators to add newly-labelled element pairs. The method first leverages a KG embedding model to encode the semantics of entities, relations, and classes in a KG and then a joint embedding model to align these elements between two KGs based on their embeddings. The joint-alignment model is actively trained by selecting high-similarity element pairs as additional

supervision signals and fine-tuning the model by adjusting parameters specifically for newly-labelled pairs, instead of treating all element pairs equally. While batch active learning uses approximation algorithms to efficiently select the best element pairs for human annotation from a generated candidate pool, an alignment graph is used to model the relatedness of element pairs. This helps decide which pairs can be inferred without re-training the model by measuring their inference power, which indicates how the entity embeddings in newly-labelled pairs affect those in unlabelled pairs.

<b>YAM++</b>	YAM++ [29] focuses on enhancing alignment precision through machine learning techniques. It learns from previous alignments, improving its matching algorithms over time. YAM++ is valuable for dynamic data environments where ontologies change frequently and require ongoing adaptation. Studies indicate that YAM++ performs well in environments requiring high accuracy but is often slower than other tools due to its complex computations.
<b>COMA++</b>	COMA++ provides an extensive library of matching techniques and supports both schema-based and instance-based matching. It is particularly effective in scenarios requiring advanced similarity measures, such as geographic information systems (GIS) and large-scale data integration projects.
<b>Falcon-AO</b>	Falcon-AO [30] is another prominent tool which uses a hybrid approach by combining lexical and structural matching techniques, focusing on the high-speed processing of small to medium-sized ontologies. Falcon-AO includes a "divide-and-conquer" strategy to handle the alignment tasks more efficiently, making it a preferred choice for projects that require quick results without sacrificing accuracy. Falcon-AO's effectiveness is evident in applications involving web-based ontologies and semantic search applications, where real-time data processing is a priority
<b>AgreementMakerLight (AML)</b>	AgreementMakerLight (AML) [21] is recognised for its adaptability and high performance in multi-domain ontology matching. AML utilises multiple matching algorithms, including lexical, structural, and instance-based techniques, to improve alignment accuracy. It also features a user-friendly interface for manual adjustments and refinements, making it suitable for domain-specific applications where user validation is essential. AML has consistently ranked among the top tools in OAEI evaluations, demonstrating strong results in domains such as medicine and environmental science, where ontologies are often large and complex. However, there are no usages in film-related ontologies.
<b>LogMap</b>	LogMap [20] has gained attention for its scalability and accuracy in handling large ontologies. LogMap employs a combination of logical reasoning and linguistic matching, allowing it to detect inconsistencies while generating alignments

## 5.4. Exploration of ontology design methodology

The scope of this section is to present the methodologies available for the ontology design, and the methodology followed for the design of SCENE ontology.

Numerous methodologies have been developed to guide the design and development of ontologies. These methodologies offer systematic approaches to determine which concepts to include, how to define relationships, and ways to ensure the ontology meets specific requirements for the intended use.

Table 29. Summary table of the ontology design methodologies available

Tool	Description
------	-------------



<p><b>Methontology</b></p>	<p>Methontology [31] is one of the earliest and most widely adopted methodologies for ontology development. Created by Gómez-Pérez and colleagues, it provides a structured framework for each stage of ontology design, including specification, conceptualisation, formalisation, and implementation. Methontology is particularly valuable when creating complex ontologies, as it emphasises iterative development and provides specific guidelines on knowledge acquisition and validation. In a cinema ontology, this might involve gathering data on films, directors, and genres from trusted sources, defining key concepts, and refining them based on expert feedback.</p> <p>Methontology, while traditionally used for building ontologies, also supports extension through modularisation techniques, encouraging the reuse of well-defined ontology components. By organising an ontology into self-contained modules, Methontology allows for more manageable extensions. For instance, a cinema ontology could be organised into separate modules for "People," "Movies," "Awards," and "Genres." If new genre classifications or award categories emerge, they can be added as separate modules without altering the core ontology, enabling flexibility in a structured manner</p>
<p><b>On-To-Knowledge</b></p>	<p>Developed for use in knowledge management applications, the On-To-Knowledge methodology supports [32] ontology design by identifying and structuring domain knowledge most relevant to the target users. This methodology incorporates a lifecycle-based approach, including phases like requirements analysis, which helps ensure that the ontology aligns closely with users' needs. Applied to cinema, On-To-Knowledge would involve identifying the most relevant aspects of film data for users (e.g., actors, release dates) and structuring the ontology to support effective search and retrieval.</p>
<p><b>Enterprise Ontology</b></p>	<p>This methodology emphasises defining ontologies for business applications by focusing on the precise identification and documentation of domain knowledge. Enterprise Ontology [33] has been widely used in industries with well-defined data standards, making it beneficial for a domain like cinema, where standard vocabularies exist for cataloguing movies, genres, and cast information. This approach would ensure that the ontology is both comprehensive and aligned with widely accepted standards within the film industry</p>
<p><b>NeOn Methodology</b></p>	<p>The NeOn methodology [34] is a flexible, modular approach designed to support the construction of networked ontologies, allowing multiple ontologies to interoperate. This method benefits complex domains like cinema, where information about movies, actors, and directors can often overlap with other domains like literature (for book adaptations) or music (for soundtracks). NeOn facilitates the reuse of existing ontologies and encourages collaboration, making it an ideal choice for building interconnected, cinema-focused ontologies that might need to integrate with other entertainment databases.</p> <p>The NeOn methodology is particularly suited for extending ontologies because of its focus on networked, modular ontologies. NeOn encourages reusing and linking ontologies rather than creating new structures from scratch, making adding new modules or sub-ontologies for emerging topics easier. In practice, for a cinema ontology, NeOn could enable the integration of additional data sources, such as online movie databases or social media platforms, to enrich the knowledge base with real-time updates on actors, releases, and reviews [34]. By treating ontology components as modules, NeOn supports ongoing expansion without disrupting existing ontology functions</p>
<p><b>Pattern-Based Extension</b></p>	<p>Another approach involves using ontology design patterns, which are reusable solutions for common modelling issues in specific domains. These patterns can serve as templates for extending ontologies consistently and help ontology</p>



	<p>developers anticipate issues when new concepts are added. In the case of a cinema ontology, patterns for “person-role” or “event-location” relationships could be employed to structure additional information about temporary roles or filming locations without altering the core ontology structure. Pattern-based extension is especially useful when new data must be incorporated in ways that are consistent with existing relationships, avoiding redundancy or conflicts. [35]</p>
<p><b>Incremental Approach and Change Management</b></p>	<p>Noy and Klein [36] introduced an incremental approach that allows for changes to be systematically introduced and evaluated, ensuring that updates do not create inconsistencies or conflicts. This method provides a framework for documenting changes, validating updates, and resolving conflicts that arise from new data or modified definitions. In a cinema ontology, where new actors, films, and events are continually added, an incremental approach with robust change management can help maintain data integrity and avoid disruptions in knowledge retrieval.</p>
<p><b>Ontology Versioning and Evolution</b></p>	<p>As ontologies grow, versioning becomes crucial to track changes, revert if necessary, and allow users to access historical ontology versions. The Ontology Versioning and Evolution approach facilitates the structured tracking of ontology modifications, ensuring that updates do not disrupt legacy applications or services relying on previous versions. This is particularly useful in a dynamic field like cinema, where continuous updates are common due to new movie releases, genre evolutions, and actor data. [37]</p>

In many domains, ontologies must evolve as new concepts, relationships, or data sources become relevant. The process of extending an ontology—adding new terms, updating relationships, or refining definitions—requires a methodological approach to ensure that the changes maintain consistency and usability. In the context of a cinema ontology, this might involve adding new genres, recognising emerging types of relationships (such as new multimedia formats), or incorporating evolving concepts like streaming platforms and digital releases.

The extension of an ontology is as critical as its initial design, especially in domains that frequently evolve, such as cinema. Methodologies like NeOn and Methontology support modular, reusable designs that can accommodate updates efficiently. Pattern-based approaches and versioning strategies offer additional flexibility, ensuring that ontologies remain consistent, reliable, and aligned with their intended purposes even as new data emerges. By carefully selecting and applying these extension methodologies, ontology designers can create a sustainable, adaptable knowledge structure that continues to meet users' needs over time. The NeOn Methodology has been followed to design and extend the SCENE-O ontology.

## 6. Data Lake Implementation

As described in previous chapters, there is no single approach to implementing a data lake, and data engineers and designers have to analyse the different needs of input and output data, storage and analytics needs, etc., with members of an organisation. In a very general sense, the different steps are:

1. **Discovery:** The different systems of an organisation or the subsystems of a specific solution are explored, creating data diagrams. This is the most essential phase, and the main aim is to understand how data flows, the different data schemas, and the best strategy for consuming data in the data lake.
2. **Setup:** This phase builds the data lake infrastructure in an independent/separate way from the core IT infrastructure of the solution. Initially, data is stored in raw format to verify the ingestion phase, and later on, refinements to the data can be specified, either by filtering existing data or by adding

additional external sources. Object storage like S3 or HDFS is highly recommended, and it is crucial to estimate the initial size of data and the growth rate to scale the data lake properly.

3. **Migration:** In this phase, users (mainly producers, but sometimes also consumers) are contacted to migrate their data into the data lake. Data ingestion can be done through batch processing or streaming jobs (preserving the original format). The best way to ingest the data should be chosen so that all relevant data is stored effectively. This step can require an extended amount of time depending on the data to be ingested; furthermore, the establishment of protocols to check that all needed data is being ingested is also important
4. **Governance:** this is the data management phase, and here, data is treated as a relevant (business) asset so that it can be trusted across the different departments of an organisation for their purposes or from other subsystems within the company. Here, the concept of **lineage** is very important:
  - a. **Schematic lineage** keeps the metadata (schemas) about the data structures
  - b. **Semantic lineage** adds metadata about the meaning of the data
  - c. **Data lineage** refers to maintaining metadata of the data's origin and any intermediate modification for tracing and audit purposes.
5. **Monitoring:** This step is responsible for monitoring the proper behaviour of the data lake and detecting failures as soon as possible; if possible, it works in a pre-emptive (predictive) mode instead of a reactive one. This is typically done by configuring alerts that detect improper behaviour or anomalies in the system. Usually, an associated dashboard is developed to provide a visual status of the data lake.
6. **Analytics:** Analysing data is the primary goal of a data lake, so part of the data in the data lake is typically used by one or more analytics engines or even reporting tools. Data scientists will use the data in the data lake to build their ML models.



Figure 17. Methodology for building a data lake

In the following sections, we will describe the different components of the data lake that have been envisioned at this phase of the project and should support the SCENE platform. However, it might evolve with the project and the steps mentioned above.

## 6.1. Architecture and building blocks

The data lake architecture was defined as part of WP2 in deliverable D2.6. This architecture diagram represented the layered organisation of SCENE’s data management system, with each layer serving a distinct function in data ingestion, storage, processing, and retrieval. Below, each component block in the architecture is mapped to the (open-source) tools supporting it, establishing a cohesive view of how they fit within the data lake’s design (see Figure 18).

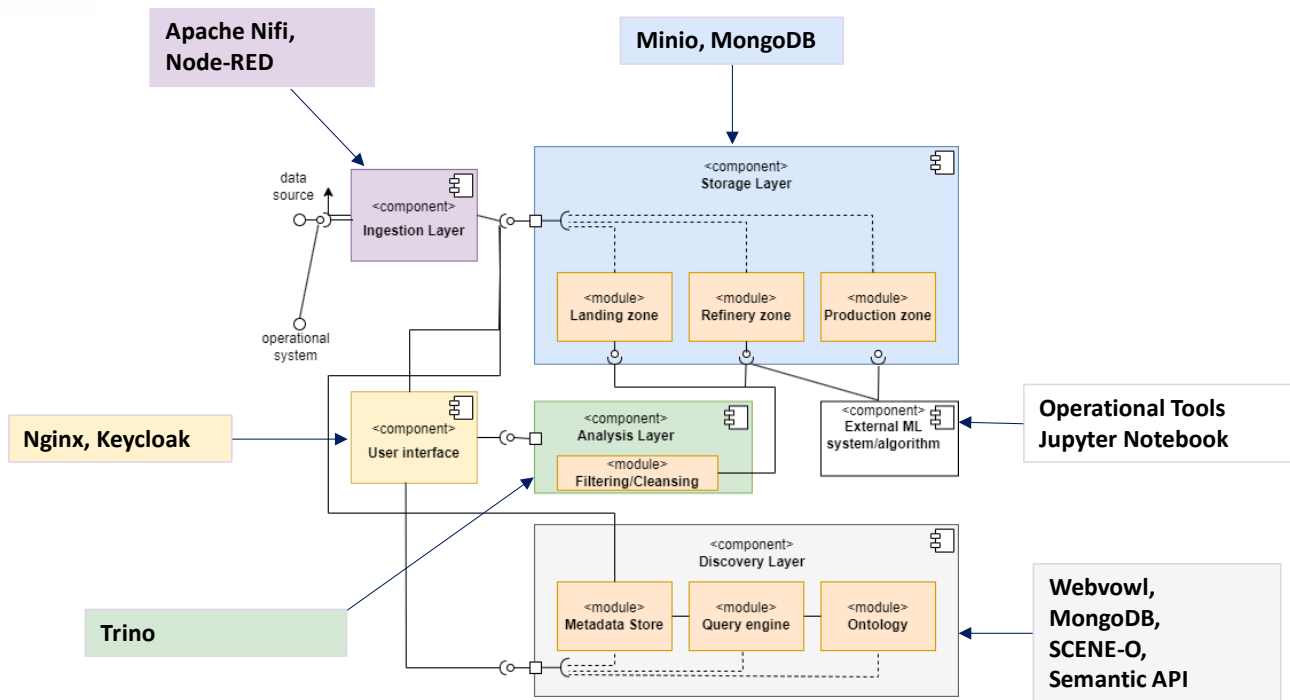


Figure 18. Mapping of software tools and architecture components

### Ingestion Layer

The **Ingestion Layer** collects and integrates data from various sources into the data lake. In SCENE, this layer is managed by two primary components: **Apache NiFi** and **Node-RED**. Apache NiFi handles complex, large-scale data ingestion workflows, managing high-volume data ingestion with robust security and data transformation capabilities. In contrast, Node-RED supports lighter-weight, API-driven integrations and quick prototyping, offering a flexible solution for interactive or event-based workflows. These tools provide complementary capabilities, covering heavy-duty data processing and agile data integrations.

### Storage Layer

The **Storage Layer** is designed to provide a secure and structured environment for raw, refined, and processed data, represented by modules such as the **Landing Zone**, **Refinery Zone**, and **Production Zone**. In SCENE, **MinIO** serves as the core component of this layer, acting as the primary storage solution for multimedia assets and ontology-related metadata. MinIO’s S3-compatible storage and scalability make it suitable for handling large volumes of diverse data, meeting the needs of all three storage zones. Additionally, **MongoDB** complements this layer by storing operational and flexible metadata related to SCENE-O, enabling schema-free storage of various metadata formats required for semantic workflows.

### Analysis Layer

The **Analysis Layer** filters, cleanses, and processes data to support analytical and semantic requirements. This layer aligns with the functionalities provided by **Trino**, which serves as SCENE’s SQL-based query engine, enabling data analytics and querying over SCENE’s data lake. Trino allows SCENE users to filter, cleanse, and process data within the data lake, making it accessible and useful for ontology-driven insights and semantic analysis. With its capability to handle complex SQL queries on large datasets, Trino enables flexible and efficient data exploration aligned with SCENE’s analytical objectives.

### Discovery Layer

The **Discovery Layer** facilitates data search, retrieval, and knowledge discovery through components like **Metadata Store**, **Query Engine**, and **Ontology**. **MongoDB** partially supports this layer as a metadata store,



providing access to structured metadata and operational information that enhances the discoverability of data in SCENE. **Trino** also contributes as a query engine within this layer, allowing users to perform efficient SQL-based searches across the data lake. Furthermore, **WebVOWL** represents part of the ontology module, providing a user-friendly interface to explore SCENE-O and other relevant ontologies visually. Together, these components form a comprehensive discovery layer that supports SCENE's semantic search and data retrieval needs.

### User Interface

The **User Interface** block is the primary access point for SCENE's components, ensuring secure and user-friendly interactions. In SCENE, **Nginx** acts as a central proxy, handling HTTP/HTTPS requests and serving as a general front-end for SCENE's services. It manages access to the embedded UIs of each component, including **MinIO's UI for storage management**, **Trino's query interface**, **Mongo Express for MongoDB management**, and **WebVOWL for ontology visualisation**. This integrated approach allows users to interact with SCENE's components through a single, secure access point while still utilising the unique user interfaces provided by each service.

### External ML System/Algorithm

The **External ML System/Algorithm** component indicates the potential integration of external machine learning systems for advanced data processing and analysis. While not fully developed within SCENE's current architecture, Jupyter Notebook serves as a prototype environment for running analytical scripts and experiments. Jupyter enables users to connect to data stored in MinIO, perform custom processing, and test machine learning workflows. Future integrations could expand on this by connecting Jupyter or similar tools with external ML systems to enhance SCENE's analytical and predictive capabilities.

All SCENE data lake architecture components (tools) have been deployed as Docker instances using a centralised docker-compose file, allowing for rapid deployment, scalability, and simplified management. This approach enables SCENE to maintain a flexible, modular infrastructure where each service is isolated and can be easily updated or replaced. The data lake platform integrates a suite of open-source tools, such as Apache NiFi, MinIO, Trino, MongoDB, Node-RED, Keycloak, and WebVOWL, each fulfilling critical functions within the data lake's layered architecture. When possible, some of the tools have been customised with SCENE's look and feel (e.g., Webvowl).

In addition to these open-source components, SCENE includes proprietary resources such as the **common front-end**, the **SCENE ontology** and the **Operational Tools**, designed to meet the project's specific semantic and data management needs.

The following subchapters describe each component's role within the data lake platform.

#### 6.1.1. Nginx Component overview (Proxy Layer- user interface)

Nginx serves as SCENE's primary proxy, managing secure access to multiple services across the platform, including MinIO, NiFi, Trino, and WebVOWL. Operating as an HTTP and HTTPS reverse proxy, Nginx facilitates secure SSL-encrypted connections, ensuring all user interactions with SCENE's data and ontological resources are safeguarded. Configured with SSL certificates, Nginx aligns with SCENE's commitment to data security and user authentication. As a gateway to SCENE's data lake services, Nginx is essential for managing access while enabling a seamless, unified user experience across the platform.

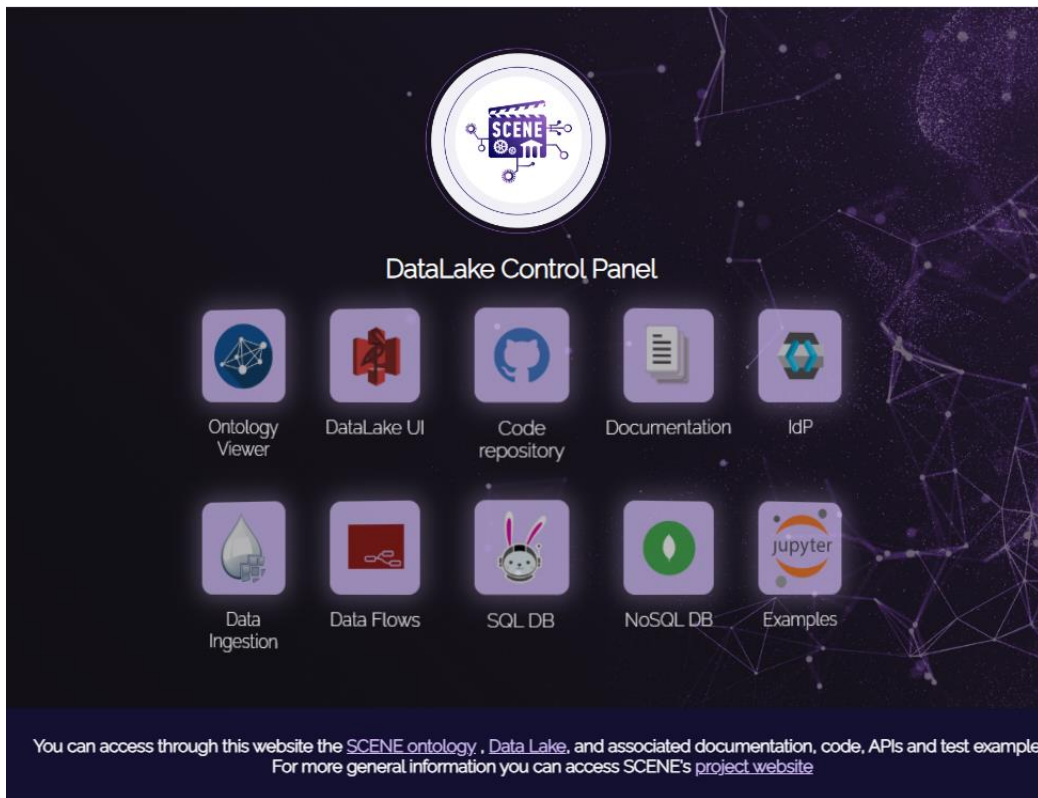


Figure 19. Data Lake web-based Control Panel

The following table (Table 30) details Nginx’s configuration, function, and planned enhancements within the SCENE architecture.

Table 30. Nginx's implementation details

Component	Nginx
Associated Layer	<ul style="list-style-type: none"> <li>Proxy Layer and general UI</li> </ul>
Function	<ul style="list-style-type: none"> <li>Operates as a reverse proxy, routing HTTP and HTTPS requests to SCENE services (MinIO, NiFi, Trino, WebVOWL).</li> <li>Manages secure, SSL-encrypted access to SCENE’s components.</li> <li>Provides centralised access management, streamlining user interactions with SCENE’s data and services.</li> </ul>
Configuration	<ul style="list-style-type: none"> <li>Configured in docker-compose to forward HTTP/HTTPS requests to backend services.</li> <li>SSL Configuration: Uses single and wildcard SSL certificates (/etc/ssl/certs) to secure multiple SCENE domains.</li> <li>Handles authentication files (nifi.htpasswd)</li> <li>Port configurations (80 for HTTP, 443 for HTTPS) ensure that Nginx can manage secure access effectively.</li> <li>Includes front-end HTML page to access the associated services (Minio, WebVowl, etc.). The page used CSS to provide SCENE look&amp;feel</li> <li>Access ports of the different services can be omitted and let only access through the virtual servers in nginx</li> </ul>
Role in SCENE	<ul style="list-style-type: none"> <li>Acts as the central access point, managing secure communication between users and SCENE’s services.</li> </ul>



	<ul style="list-style-type: none"> <li>• Enables consistent and secure access to SCENE’s components by enforcing SSL encryption (and role-based access, when needed).</li> <li>• Supports SCENE’s data security objectives by controlling and routing all external access, particularly for sensitive ontology-driven workflows.</li> </ul>
<p><b>Specific Goals and Requirements</b></p>	<ul style="list-style-type: none"> <li>• <b>Data Security:</b> Ensures all data and service interactions are SSL-encrypted, protecting sensitive multimedia and ontological data.</li> <li>• <b>Centralized Access Management:</b> Consolidates access to SCENE services, creating a unified and secure gateway.</li> <li>• <b>Role-Based Control:</b> Configured to support role-based access via Keycloak, securing user interactions (if needed).</li> <li>• <b>Scalability and Flexibility:</b> Easily adaptable to add or remove services as SCENE’s needs evolve, with minimal reconfiguration required.</li> </ul>
<p><b>Interactions</b></p>	<ul style="list-style-type: none"> <li>• <b>Keycloak:</b> Can work with Keycloak to delegate authentication for SCENE’s services, ensuring consistent role-based access.</li> <li>• <b>MinIO, NiFi, Trino, WebVOWL:</b> Routes user requests securely to these services, enabling centralised and secure data access.</li> <li>• <b>SSL Certificates:</b> SSL certificates are utilized to secure communications for SCENE’s various services, protecting user data and service interactions.</li> <li>• <b>API Access Management:</b> Secures access to SCENE’s APIs, supporting a seamless user experience across SCENE’s ontology and data lake functionalities.</li> </ul>
<p><b>Specific Configuration Relevant to T3.1</b></p>	<ul style="list-style-type: none"> <li>• <b>Proxy Configuration for Ontology Tools:</b> Configured to route requests to WebVOWL and other ontology tools, enabling seamless, secure visualisation of SCENE-O.</li> <li>• <b>Secure API Gateway:</b> Provides a secure gateway for SCENE-O-related APIs, ensuring consistent access control and data protection for ontology-related tasks.</li> </ul>
<p><b>Expected Outcomes</b></p>	<ul style="list-style-type: none"> <li>• <b>Enhanced Security and Accessibility:</b> Provides secure, streamlined access to SCENE’s services, aligning with SCENE’s security requirements.</li> <li>• <b>Unified User Experience:</b> Ensures consistent and seamless user interaction with SCENE’s components through a single, secure access point.</li> <li>• <b>Controlled Access to Ontological Data:</b> Protects SCENE-O and other ontology-based workflows, ensuring that only authorised users can access sensitive metadata and semantic tools.</li> <li>• <b>Data Integrity and Compliance:</b> Supports SCENE’s GDPR compliance by securing data interactions and managing access centrally.</li> </ul>
<p><b>Future Enhancements</b></p>	<ul style="list-style-type: none"> <li>• <b>Keycloak Integration Expansion:</b> Further integration with Keycloak for more refined access control and enhanced security.</li> <li>• <b>Load Balancing:</b> Implement load balancing within Nginx to handle increased traffic as SCENE’s data grows.</li> <li>• <b>Enhanced Security Protocols:</b> Add security headers and multi-factor authentication options to strengthen data security.</li> <li>• <b>Logging and Monitoring:</b> Improve logging and monitoring capabilities to provide detailed insights into access patterns and enhance security response.</li> </ul>



### 6.1.2. Keycloak Component Overview (Authentication and Access Management)

The Authentication and Access Management Layer in SCENE is essential for securing access to data and services. Keycloak, an open-source Identity and Access Management (IAM) tool, helps provide centralised authentication and authorisation for SCENE, securely managing user access across all components. Keycloak offers a robust foundation for role-based access control, ensuring secure handling of sensitive data in line with SCENE’s requirements. As the project scales, Keycloak is planned for integration with additional services, allowing granular user permissions and enhancing secure data access throughout the SCENE platform.

Below is a detailed table (Table 31) that outlines Keycloak’s functions, configurations, and planned enhancements.

Table 31. Keycloak's implementation details

Component	Keycloak
Associated Layer	<ul style="list-style-type: none"> <li>Authentication and Access Management Layer</li> </ul>
Function	<ul style="list-style-type: none"> <li>Can act as SCENE’s Identity Provider (IdP), handling user authentication and access control.</li> <li>Provides centralised, secure access management for SCENE’s components.</li> <li>Manages roles and permissions, allowing for role-based access control as the project scales.</li> </ul>
Configuration	<ul style="list-style-type: none"> <li>Configured via docker-compose with initial admin credentials, allowing for secure local access.</li> <li>Uses Nginx as a proxy to manage access securely, aligning with SCENE’s SSL configuration.</li> <li>Environment configured with KC_HOSTNAME environmental variable to include how the IdP is accessed through Nginx from the outside.</li> <li>Flexible enough to scale to more complex user roles and permissions as additional user groups are added to SCENE.</li> </ul>
Role in SCENE	<ul style="list-style-type: none"> <li>Provides a secure, centralised authentication solution for all SCENE components, supporting consistent access control.</li> <li>Ensures data access is restricted to authorised users, contributing to the secure handling of multimedia and ontology metadata.</li> <li>Planned to extend role-based access control across SCENE components, especially those interacting with sensitive data.</li> <li>Note that this Keycloak is specially targeting IdP services for the data lake, but can be integrated in the whole SCENE platform</li> </ul>
Specific Goals and Requirements	<ul style="list-style-type: none"> <li><b>Scalable Access Control:</b> Supports growing users and roles as SCENE expands.</li> <li><b>Role-Based Permissions:</b> Required for managing different access levels based on user roles.</li> <li><b>Integration with SCENE Services:</b> Seamlessly integrates with SCENE’s services (e.g., MinIO, NiFi) to provide uniform and secure access.</li> <li><b>Compliance and Security:</b> Ensures secure access to GDPR compliance, particularly relevant to ontology-linked data.</li> </ul>
Interactions	<ul style="list-style-type: none"> <li><b>Nginx:</b> Works with Nginx to manage secure connections and serve as the proxy for Keycloak’s authentication services.</li> <li><b>MinIO:</b> Future integration planned for access control delegation to Keycloak, enhancing security.</li> </ul>



	<ul style="list-style-type: none"> <li>• <b>NiFi (and Trino):</b> Planned to provide role-based access for these services, allowing tailored permissions and secure workflows.</li> </ul>
<b>Specific Configuration Relevant to T3.1</b>	<ul style="list-style-type: none"> <li>• <b>Role-Based Access for Ontology Tools:</b> Configured to manage access control over ontology-related data and tools, ensuring users have appropriate permissions to access sensitive SCENE-O data.</li> <li>• <b>Flexible User Realm Setup:</b> Designed to support realm-based configurations, allowing different access groups for SCENE participants as required for T3.1 and other SCENE tasks.</li> <li>• <b>Integration Readiness for Secure API Access:</b> Configured to extend secure API-based access for SCENE-O and other ontology-driven tools.</li> </ul>
<b>Expected Outcomes</b>	<ul style="list-style-type: none"> <li>• <b>Consistent and Secure Access Management:</b> Ensures secure and consistent access across SCENE, centralising authentication and authorisation.</li> <li>• <b>Scalable User Management:</b> Provides a flexible, scalable framework that can accommodate more users and roles as SCENE grows.</li> <li>• <b>Data Protection and Compliance:</b> Aligns with GDPR and other data protection policies, supporting secure access to sensitive multimedia and ontology data.</li> <li>• <b>Streamlined Access for Ontology Tools:</b> Centralizes access control for SCENE-O and related tools, enhancing security and user management in semantic workflows.</li> </ul>
<b>Future Enhancements</b>	<ul style="list-style-type: none"> <li>• <b>Expanded Role-Based Access Control:</b> Implement detailed, role-based permissions as more SCENE tools and data are integrated.</li> <li>• <b>Integration with MinIO and Other SCENE Services:</b> Extend Keycloak's access management to MinIO, NiFi, and Trino for centralised user management.</li> <li>• <b>Enhanced Security Protocols:</b> Add multi-factor authentication and enhanced auditing features as SCENE scales.</li> <li>• <b>User Group and Policy Extensions:</b> Develop detailed user group configurations and policy frameworks tailored for ontology-related permissions and SCENE-O access.</li> </ul>

### 6.1.3. Apache NiFi Component Overview (Ingestion Layer)

The Ingestion Layer in SCENE is vital in managing data flows, ensuring smooth ingestion, transformation, and routing of diverse data sources into the data lake. Apache NiFi was chosen for its ability to handle real-time and batch data and manage complex data pipelines in a scalable, flexible, and user-friendly manner. NiFi supports SCENE's diverse data ingestion needs, from structured to unstructured data, making it essential for data processing, transformation, and automation. Configured to operate with SSL security and integrated through Nginx, NiFi enables the continuous ingestion and orchestration of multimedia and metadata for SCENE-O and other semantic components, aligning with the requirements and goals of T3.1.

The table below provides a detailed overview of NiFi's function, configuration, role in SCENE, and planned enhancements to support the growing needs of SCENE's data lake architecture.

Table 32. Apache NiFi's implementation details

Component	Apache Nifi
Associated Layer	<ul style="list-style-type: none"> <li>• Ingestion Layer</li> </ul>



<p><b>Function</b></p>	<ul style="list-style-type: none"> <li>• Manages data ingestion workflows, allowing real-time and batch processing.</li> <li>• Enables data transformation and routing to ensure smooth data pipeline operation.</li> <li>• Supports secure data transfer into the data lake, handling structured and unstructured data.</li> </ul>
<p><b>Configuration</b></p>	<ul style="list-style-type: none"> <li>• Configured through docker-compose, running on HTTPS on port 8443 to enhance security.</li> <li>• Access is not controlled via Nginx but internally (internal authorisers that could be later delegated to other IDPs).</li> <li>• Initial setup with single-user access for ease of local development; planned for integration with Keycloak for future role-based access control.</li> <li>• Certificates for SSL require KeyStore and Truststore</li> <li>• Requires to know how NiFi is accessed through the proxy (nginx) via the NIFI_WEB_PROXY_HOST environment variable</li> </ul>
<p><b>Role in SCENE</b></p>	<ul style="list-style-type: none"> <li>• Serves as SCENE’s core data ingestion service, moving raw and processed data into the storage layer (MinIO).</li> <li>• Automates the data flow to streamline and standardise ingestion processes for SCENE-O and metadata tagging requirements.</li> <li>• Provides the basis for scalable and flexible data pipelines, supporting ad-hoc and routine ingestion needs across various data sources and formats.</li> </ul>
<p><b>Specific Goals and Requirements</b></p>	<ul style="list-style-type: none"> <li>• <b>Scalability:</b> Supports expanding data needs with batch and real-time ingestion capabilities.</li> <li>• <b>Data Quality and Transformation:</b> Ensures that ingested data is transformed and cleansed before storage.</li> <li>• <b>Security:</b> Operates over HTTPS, with plans for role-based access via Keycloak for enhanced security.</li> <li>• <b>Integration and Flexibility:</b> Must handle multiple data types and integrate with SCENE-O for ontology-based metadata ingestion.</li> </ul>
<p><b>Interactions</b></p>	<ul style="list-style-type: none"> <li>• <b>MinIO:</b> Directs ingested data to MinIO for storage and access by other SCENE components.</li> <li>• <b>Nginx:</b> Utilizes Nginx as a proxy to manage secure connections and user authentication.</li> <li>• <b>Keycloak (Future):</b> Planned for access control integration, allowing role-based permissions for various ingestion tasks.</li> <li>• <b>Trino:</b> Enables ingestion of data that will be later analysed and visualised for ontology support and semantic annotation.</li> </ul>
<p><b>Specific Configuration Relevant to T3.1</b></p>	<ul style="list-style-type: none"> <li>• <b>Data Routing and Transformation:</b> Configured to route data into predefined buckets in MinIO, based on type (e.g., multimedia vs. metadata), which is crucial for ontology alignment.</li> <li>• <b>Flexible Pipeline Support:</b> NiFi’s flexible pipeline setup supports SCENE-O-related tasks like metadata tagging and semantic annotation preparation.</li> <li>• <b>Integration with Annotation Workflow:</b> Enables direct handling and transformation of ontology-related data, aiding in real-time data processing for SCENE’s ontology needs.</li> </ul>
<p><b>Expected Outcomes</b></p>	<ul style="list-style-type: none"> <li>• <b>Streamlined Data Ingestion:</b> Provides a reliable, automated data ingestion mechanism that supports SCENE-O and metadata tagging needs.</li> </ul>



	<ul style="list-style-type: none"> <li>• <b>Scalable and Secure Data Pipelines:</b> Allows data to be ingested securely and at scale, ensuring that SCENE’s data lake can handle high-frequency and large-volume data inputs.</li> <li>• <b>Enhanced Data Processing for Ontology Tasks:</b> Ensures that data is properly processed and structured for ontology-driven queries and tagging, making it readily accessible for semantic annotation and retrieval.</li> </ul>
<b>Future Enhancements</b>	<ul style="list-style-type: none"> <li>• <b>Keycloak Integration:</b> Planned integration with Keycloak for role-based access control, ensuring secure, flexible permissions.</li> <li>• <b>Enhanced Data Flow Control:</b> Implement advanced flow versioning with NiFi Registry to allow rollback and reuse of data flows.</li> <li>• <b>Expansion to Kubernetes:</b> Deploying NiFi within Kubernetes for enhanced scalability, load balancing, and management as ingestion requirements grow.</li> <li>• <b>Custom Data Pipeline Extensions:</b> Build custom processors to handle specific SCENE ontology requirements, such as automated metadata extraction and tagging.</li> </ul>

### 6.1.4. Node-RED Component Overview (Ingestion Layer)

Node-RED is a flow-based development tool that enables rapid integration and automation within SCENE’s data ingestion workflows. While similar to Apache NiFi in its ability to manage data flows, Node-RED is more lightweight and often favoured for quickly prototyping or integrating devices and APIs. Apache NiFi excels in handling large-scale, enterprise-level data ingestion with robust data lineage tracking and security features, while Node-RED is typically chosen for simpler or more interactive data flows. Together, they offer complementary capabilities: NiFi manages high-volume, complex data processing with strict controls, whereas Node-RED provides a flexible and easy-to-deploy option for interactive and lightweight integrations.

The following table provides an overview of Node-RED’s configuration, role in SCENE, and potential enhancements.

Table 33. Node-RED implementations details

Component	Node-RED
<b>Associated Layer</b>	<ul style="list-style-type: none"> <li>• Ingestion Layer</li> </ul>
<b>Function</b>	<ul style="list-style-type: none"> <li>• Enables rapid prototyping and lightweight data flows, ideal for API integrations and interactive workflows.</li> <li>• Simplifies data routing and automation for smaller, event-driven processes.</li> <li>• Complements NiFi by offering a user-friendly environment for quickly building and modifying data flows.</li> </ul>
<b>Configuration</b>	<ul style="list-style-type: none"> <li>• Configured via docker-compose, with port 1880 exposed for web interface access.</li> <li>• Environment set with TZ=Europe/Amsterdam to align with SCENE’s standard time zone.</li> <li>• Uses an attached volume (./node-red-data) to store data flow configurations, ensuring session persistence.</li> <li>• Operates within SCENE’s secure subnet and manages access through Nginx for HTTPS security.</li> </ul>



<p><b>Role in SCENE</b></p>	<ul style="list-style-type: none"> <li>• Provides a flexible environment for interactive data flow management, particularly useful for API-based ingestion and quick prototyping.</li> <li>• Allows SCENE users to easily create and modify lightweight flows, supporting smaller, interactive workflows that don't require the scale and complexity of NiFi.</li> <li>• Supports T3.1 by enabling custom data routing, aiding in annotation workflows and simpler ontology-related data processes.</li> </ul>
<p><b>Specific Goals and Requirements</b></p>	<ul style="list-style-type: none"> <li>• <b>Ease of Use:</b> Designed to be intuitive, allowing users to quickly create, test, and deploy new data flows with minimal setup.</li> <li>• <b>Rapid Prototyping:</b> Supports fast development of small-scale workflows for testing and integrating APIs relevant to SCENE-O and semantic data.</li> <li>• <b>Interoperability:</b> Operates within SCENE's subnet, easily integrating with other services such as MinIO and MongoDB.</li> <li>• <b>Event-Driven Workflows:</b> Ideal for building workflows that respond to events, such as changes in metadata or ontology tags.</li> </ul>
<p><b>Interactions</b></p>	<ul style="list-style-type: none"> <li>• <b>Nginx:</b> Accessed through Nginx for secure, HTTPS-protected connections.</li> <li>• <b>MinIO:</b> Connects with MinIO to access data storage, particularly for lightweight data retrieval and prototyping tasks.</li> <li>• <b>MongoDB:</b> Might interact with MongoDB for metadata storage and simple data flow tasks related to SCENE-O.</li> <li>• <b>API Integrations:</b> Works alongside external APIs and SCENE services, allowing flexible integration.</li> </ul>
<p><b>Specific Configuration Relevant to T3.1</b></p>	<ul style="list-style-type: none"> <li>• <b>API and Event Handling for Ontology Tasks:</b> Configured to integrate easily with API-based workflows, also supporting interactive ontology-related tasks and annotation tagging.</li> <li>• <b>Lightweight Data Transformation:</b> Optimized for simpler data transformations, making it ideal for tagging and metadata updates</li> <li>• <b>Customizable Flow Storage:</b> Uses persistent volume to store flow configurations, supporting reusable and modifiable workflows that align with SCENE's semantic requirements.</li> </ul>
<p><b>Expected Outcomes</b></p>	<ul style="list-style-type: none"> <li>• <b>Enhanced Flexibility for Lightweight Data Flows:</b> Provides a rapid, user-friendly environment for developing small-scale ingestion and transformation workflows.</li> <li>• <b>Improved Accessibility for Non-Technical Users:</b> Allows even non-technical users to create and manage data flows relevant to SCENE's ontology needs.</li> <li>• <b>Quick Prototyping for Ontology Integration:</b> Supports fast prototyping and testing of new semantic data flows, enabling efficient API integration and workflow automation.</li> <li>• <b>Complementary Tool to NiFi:</b> Adds versatility to SCENE's ingestion layer by handling simpler tasks, freeing NiFi for more complex workflows.</li> </ul>
<p><b>Future Enhancements</b></p>	<ul style="list-style-type: none"> <li>• <b>Expanded Keycloak Integration:</b> It is possible to integrate with Keycloak for role-based access control, ensuring secure user access as the platform scales.</li> <li>• <b>Enhanced API Support:</b> Extend API integration capabilities, allowing Node-RED to interact with more SCENE services and external data sources.</li> <li>• <b>Containerization for Scalability:</b> Explore migration to Kubernetes to support higher availability and load management</li> </ul>



- **Custom Nodes for Ontology-Specific Tasks:** Develop or integrate custom Node-RED nodes for specific ontology-driven tasks, such as semantic tagging or metadata management within SCENE-O.

### 6.1.5. Minio Component Overview (Storage Layer)

The Storage Layer of the SCENE architecture is crucial for managing and preserving the wide range of multimedia and metadata assets central to SCENE’s data lake. MinIO, an object storage service compatible with the S3 API, is the backbone of this layer. It supports SCENE’s extensive storage needs, accommodating structured and unstructured data, including high-volume multimedia files and ontology metadata required for the SCENE-O ontology framework. Configured to operate securely through SSL and managed access protocols, MinIO provides a reliable, scalable, and flexible data repository that meets SCENE’s resilience, interoperability, and compliance requirements. MinIO also plays a pivotal role in supporting T3.1 objectives by ensuring that ontology-linked data and semantic annotations are stored accessibly for seamless integration with other SCENE tools and services.

The table below provides a detailed view of MinIO’s function, configuration, role, interactions within SCENE, and future enhancements designed to support the project’s growth and evolving data requirements.

Table 34. Minio’s implementation details

Component	Minio
Associated Layer	<ul style="list-style-type: none"> <li>• Storage Layer</li> </ul>
Function	<ul style="list-style-type: none"> <li>• Provides object storage for multimedia and metadata files.</li> <li>• Compatible with S3 API for scalable and flexible data storage.</li> <li>• Handles both structured and unstructured data.</li> </ul>
Configuration	<ul style="list-style-type: none"> <li>• Configured via docker-compose file.</li> <li>• Uses a persistent data volume (./minio_data) for storage resilience.</li> <li>• Secured through Nginx with SSL for HTTPS access.</li> <li>• SSL Configuration: Uses wildcard SSL certificates via Nginx to support multiple SCENE domains for secure data access across services.</li> <li>• Admin credentials and test user for initial setup; supports future role-based access as SCENE scales.</li> <li>• Volume Management: Set up to support potential data backup and recovery configurations as project needs grow.</li> </ul>
Role in SCENE	<ul style="list-style-type: none"> <li>• Acts as the primary storage solution for SCENE’s multimedia assets and metadata.</li> <li>• Provides an easily accessible, scalable repository for data lakes.</li> <li>• Supports future integrations with other SCENE components for data processing and analysis (e.g., with Trino for SQL queries).</li> </ul>
Specific Goals and Requirements	<ul style="list-style-type: none"> <li>• <b>Scalability:</b> Should handle large multimedia datasets as SCENE expands.</li> <li>• <b>Interoperability:</b> S3 compatibility allows easy integration with external tools and workflows.</li> <li>• <b>Redundancy and Resilience:</b> Requires a setup that ensures data availability and persistence.</li> <li>• <b>Security:</b> Ensures that stored data is only accessible via secure channels and to authorised users.</li> </ul>



	<ul style="list-style-type: none"> <li>• <b>GDPR Compliance:</b> Must support policies for data deletion and restricted access to personal data to meet GDPR requirements in Europe.</li> </ul>
<p><b>Interactions</b></p>	<ul style="list-style-type: none"> <li>• <b>Nginx:</b> Serves as a proxy for secure access, managing HTTPS connections and forwarding requests.</li> <li>• <b>Trino:</b> Utilizes MinIO for SQL queries, allowing data exploration across SCENE’s data lake.</li> <li>• <b>Keycloak (Future):</b> Planned for access management integration to delegate authentication and access control directly from Keycloak.</li> <li>• <b>WebVOWL and Other Ontology Tools:</b> Supports storage of ontology-related data accessible via MinIO API for analysis and annotation.</li> </ul>
<p><b>Specific Configuration Relevant to T3.1</b></p>	<ul style="list-style-type: none"> <li>• <b>S3-Compatible API:</b> Enables seamless data storage and retrieval for ontology-related multimedia files.</li> <li>• <b>Bucket Structure:</b> Organized with shared buckets for general access and partner-specific buckets for restricted access, facilitating collaborative and secure work environments.</li> <li>• <b>Jupyter Notebook Integration:</b> Preconfigured with examples for S3 API access using Python and JavaScript to enable rapid ontology-related data exploration (e.g., annotation tools, metadata updates).</li> <li>• <b>Metadata Management and Ontology Support:</b> Prepared to store ontology metadata and ontology-related data structures like SCENE-O in a schema-compatible format.</li> <li>• <b>Integration with Annotation Tools:</b> Enables storage and retrieval of semantic annotations, supporting the tagging and mapping functions of T3.1’s ontology tools.</li> </ul>
<p><b>Expected Outcomes</b></p>	<ul style="list-style-type: none"> <li>• <b>Efficient Data Management:</b> A reliable storage solution that ensures SCENE’s data lake can manage large, complex datasets, including multimedia assets.</li> <li>• <b>Interoperable and Expandable Storage:</b> S3 compatibility allows SCENE’s data to integrate with other cloud-native tools and external applications.</li> <li>• <b>Data Accessibility:</b> Ensures that the data required for SCENE-O and ontology-driven processes is consistently available and structured for query and analysis.</li> <li>• <b>Data Lake Reliability:</b> Designed to maintain high availability, allowing SCENE-O and other ontology-driven tasks to function seamlessly even with large data volumes.</li> <li>• <b>Compliance:</b> Configuration will align with data governance and privacy (GDPR), supporting data retention and deletion policies to ensure responsible data management.</li> </ul>
<p><b>Future Enhancements</b></p>	<ul style="list-style-type: none"> <li>• <b>Integration with Keycloak:</b> Provide role-based access control for finer data access permissions.</li> <li>• <b>Kubernetes Deployment:</b> This is for improved load balancing, scalability, and management as data volumes increase.</li> <li>• <b>Enhanced Data Redundancy and Backup Policies:</b> To further ensure data security and resilience, particularly as SCENE’s data needs expand.</li> <li>• <b>Expanded API Functionality:</b> To support advanced metadata queries and provide enriched data access points for other SCENE tools.</li> </ul>



- **Data Cataloging and Governance:** Consider integration with tools like Apache Atlas or CKAN to facilitate data lineage and governance, especially for tracking ontology updates and data evolution.
- **Extended API for Ontology-Specific Queries:** Plan to support enriched query features for metadata and ontology-linked fields, enabling SCENE-O data to be accessed more precisely.

### 6.1.6. Mongo Component Overview (Storage Layer)

MongoDB is a NoSQL database solution used within SCENE to support storing operational data, metadata, and backend requirements that complement SCENE’s data lake. It is particularly useful for storing flexible and schema-free data structures, such as metadata related to SCENE-O and user interaction logs. Configured within the SCENE architecture to operate securely, MongoDB works alongside Mongo Express to provide easy data management and administration, ensuring data accessibility and storage adaptability. Currently, it only supports the Operational Tools, but it can be configured if any other SCENE tools require such a database or even as a way of storing cleaned and analysed data to be exposed to other high-level applications.

The following table summarises MongoDB’s function, configuration, and potential enhancements in SCENE.

*Table 35. MogoDB's implementation details*

Component	MongoDB + Mongo Express
Associated Layer	<ul style="list-style-type: none"> <li>• Storage and backend Layer</li> </ul>
Function	<ul style="list-style-type: none"> <li>• Stores operational data, metadata, and backend information required in SCENE’s Operational Tools component.</li> <li>• Provides a schema-free, flexible storage solution for storing diverse data formats, such as ontology metadata and user logs.</li> <li>• Works with Mongo Express for simplified data management and inspection.</li> </ul>
Configuration	<ul style="list-style-type: none"> <li>• Configured in docker-compose with an attached volume (./ot-mongodata) to ensure data persistence.</li> <li>• Uses admin credentials for secure database access.</li> <li>• Operates within a secure subnet, with Nginx controlling external access when needed.</li> <li>• Can be configured for future role-based access via Keycloak to secure sensitive metadata and operational data.</li> </ul>
Role in SCENE	<ul style="list-style-type: none"> <li>• It serves as the backend storage for operational and metadata-related information required by SCENE’s Operational Tools, but other SCENE tools can re-use it.</li> <li>• Complements the data lake by storing flexible metadata formats that support ontology-driven queries and user interaction data.</li> <li>• Ensures data is accessible for SCENE’s components requiring rapid backend data access, supporting T3.1’s tagging and metadata management needs.</li> </ul>
Specific Goals and Requirements	<ul style="list-style-type: none"> <li>• <b>Flexible Metadata Storage:</b> Provides schema-less storage to accommodate various data formats for SCENE-O and related ontologies, if needed.</li> <li>• <b>Scalability:</b> Supports expanding metadata requirements as SCENE’s dataset grows.</li> </ul>



	<ul style="list-style-type: none"> <li>• <b>Secure Data Handling:</b> Operates within a secure subnet, with plans for integration with Keycloak for role-based access.</li> <li>• <b>High Availability:</b> Ensures data persistence and accessibility for SCENE’s backend operations.</li> </ul>
<p><b>Interactions</b></p>	<ul style="list-style-type: none"> <li>• <b>Mongo Express:</b> Paired with Mongo Express for easy data management and inspection, allowing admins to check and manage backend data securely.</li> <li>• <b>Nginx:</b> Access to Mongo Express is managed through Nginx, providing secure and limited access to backend services.</li> <li>• <b>Keycloak (Future):</b> Planned integration with Keycloak to secure access to sensitive metadata and operational data, enabling role-based control.</li> <li>• <b>Data Lake Components:</b> Works with data lake components for metadata storage related to SCENE-O and other ontology-driven processes.</li> </ul>
<p><b>Specific Configuration Relevant to T3.1</b></p>	<ul style="list-style-type: none"> <li>• <b>Schema-Free Metadata Storage:</b> Configured to store ontology metadata and annotations that may vary in structure, essential for SCENE-O and T3.1’s tagging requirements.</li> <li>• <b>Integration with Annotation Tools:</b> Supports metadata from annotation tools, enabling flexible storage and access.</li> <li>• <b>User Interaction Logs:</b> Stores logs related to models published in the Operational Tools.</li> </ul>
<p><b>Expected Outcomes</b></p>	<ul style="list-style-type: none"> <li>• <b>Efficient Metadata Management:</b> Enables rapid and flexible storage of ontology metadata and annotations, if needed, supporting SCENE-O and semantic workflows.</li> <li>• <b>Adaptable Storage Solution:</b> Provides a flexible backend database that can easily expand to accommodate additional data as SCENE’s needs grow.</li> <li>• <b>Enhanced Backend Functionality:</b> Supports backend tasks such as user logging, metadata retrieval, and annotation storage, contributing to T3.1’s focus on data tagging and management.</li> <li>• <b>Data Integrity and Accessibility:</b> Ensures that essential metadata and operational data remain accessible and secure for SCENE’s data lake and semantic tasks.</li> </ul>
<p><b>Future Enhancements</b></p>	<ul style="list-style-type: none"> <li>• <b>Keycloak Integration for Role-Based Access:</b> Expand secure access control for MongoDB, integrating it with Keycloak for role-based permissions.</li> <li>• <b>Cluster Setup for Scalability:</b> Consider deploying MongoDB in a cluster to support high availability and load distribution as metadata and backend requirements increase.</li> <li>• <b>Advanced Logging and Data Analytics:</b> Implement extended logging and analytics functions to track metadata usage and interaction trends.</li> <li>• <b>Enhanced Data Management Tools:</b> Expand Mongo Express functionality or integrate with other management tools for advanced data inspection and optimisation.</li> </ul>

### 6.1.7. Trino Component Overview (Discovery Layer)

The Discovery Layer in SCENE is responsible for querying and extracting insights from data stored in the data lake. Trino, a distributed SQL query engine, provides SCENE with high-performance SQL-based querying capabilities over large datasets. Configured for secure access and optimised for SCENE's analytical needs, Trino enables complex querying and analytics for exploring and integrating ontology-driven data. This supports T3.1 objectives by facilitating data exploration and analysis workflows directly from the storage



layer, with future enhancements planned for expanding query capabilities and performance as data volumes increase.

The table below provides a comprehensive view of Trino’s function, configuration, role in SCENE, and potential enhancements to meet evolving project needs.

Table 36. Trino's implementation details

Component	Trino
<b>Associated Layer</b>	<ul style="list-style-type: none"> <li>Analysis (and Discovery) Layer</li> </ul>
<b>Function</b>	<ul style="list-style-type: none"> <li>Provides high-performance SQL querying capabilities over SCENE’s data lake.</li> <li>Enables structured exploration and analysis of data within MinIO.</li> <li>Supports complex query operations for large datasets, essential for SCENE’s semantic and analytical tasks.</li> </ul>
<b>Configuration</b>	<ul style="list-style-type: none"> <li>Configured in docker-compose with port 8082 exposed for accessing the Trino UI.</li> <li>Utilizes a dedicated data directory (./trino/data) for query configurations and storage management.</li> <li>Log level set to INFO, with customisable configuration files (<i>config.properties</i>, <i>node.properties</i>, <i>jvm.config</i>) for tailored performance.</li> <li>Integrated through Nginx for secure access via HTTPS, ensuring data security during querying.</li> <li>Authentication is provided internally, not through Nginx</li> </ul>
<b>Role in SCENE</b>	<ul style="list-style-type: none"> <li>Acts as the core querying engine, if needed, allowing structured SQL access to data stored in MinIO.</li> <li>Supports SCENE-O by facilitating data queries related to ontology metadata, aiding in analysing and retrieving semantically relevant data.</li> <li>Enables advanced reporting and data exploration for SCENE’s data-driven objectives.</li> </ul>
<b>Specific Goals and Requirements</b>	<ul style="list-style-type: none"> <li><b>High-Performance Discovery:</b> Provides fast and efficient data processing over large datasets.</li> <li><b>Data Structure Compatibility:</b> Supports SCENE-O with SQL queries to extract metadata for ontology-based retrieval.</li> <li><b>Scalability:</b> Must accommodate growing data volumes, supporting batch and real-time query needs as SCENE scales.</li> <li><b>Security:</b> Operates through Nginx for secure query access and interaction with MinIO.</li> </ul>
<b>Interactions</b>	<ul style="list-style-type: none"> <li><b>MinIO:</b> Accesses and queries data stored in MinIO, retrieving information essential for SCENE’s analytics and semantic exploration tasks.</li> <li><b>Nginx:</b> Uses Nginx as a proxy for secure access, facilitating encrypted connections between Trino and other SCENE components.</li> </ul>
<b>Specific Configuration Relevant to T3.1</b>	<ul style="list-style-type: none"> <li><b>Schema and Data Structure Support:</b> Configured to support schema-based queries aligned with SCENE-O’s ontology structure, enabling effective data retrieval for semantic purposes.</li> <li><b>Custom Query Optimization:</b> Adjusted for analytical workloads relevant to ontology and metadata, ensuring queries meet SCENE’s processing requirements.</li> </ul>



	<ul style="list-style-type: none"> <li>• <b>Integration with Semantic Analysis Workflows:</b> Supports analysis workflows by enabling SQL-based queries that retrieve and structure data for ontology tagging and annotation.</li> </ul>
<p><b>Expected Outcomes</b></p>	<ul style="list-style-type: none"> <li>• <b>Support for Enhanced Data Analytics:</b> Provides robust query capabilities that support SCENE’s analytics needs, enabling SQL-based data exploration for ontology-related insights.</li> <li>• <b>Efficient Ontology-Driven Retrieval:</b> Facilitates data retrieval workflows by enabling queries that align with SCENE-O and semantic annotation requirements.</li> <li>• <b>Flexible Data Access:</b> Supports a variety of SQL operations, making data easily accessible for reporting, analysis, and semantic queries across SCENE’s components.</li> </ul>
<p><b>Future Enhancements</b></p>	<ul style="list-style-type: none"> <li>• <b>Kubernetes Deployment:</b> Plan to deploy Trino within Kubernetes to scale with data demand, improve query load distribution, and ensure high availability.</li> <li>• <b>Integration with SCENE-O Ontology API:</b> Explore integrating Trino queries directly with ontology APIs to enhance real-time data retrieval based on ontology mappings.</li> <li>• <b>Advanced Query Extensions:</b> Develop custom SQL extensions tailored for semantic queries, supporting advanced data retrieval functions relevant to SCENE-O.</li> <li>• <b>Enhanced Logging and Monitoring:</b> Implement detailed logging and monitoring tools to track query performance and optimise based on SCENE’s data retrieval patterns.</li> </ul>

### 6.1.8. WebVowl Component Overview (Discovery Layer)

The Discovery Layer (including visualization) is crucial for SCENE’s user experience, enabling intuitive visualization and interaction with SCENE-O and other relevant ontologies. WebVOWL, an interactive ontology visualization tool, provides a graphical interface adapted for SCENE, making ontology exploration user-friendly and insightful. With SCENE’s look and feel and customised views of SCENE-O, WebVOWL supports users in exploring complex semantic structures, understanding relationships, and accessing metadata details essential for T3.1’s objectives. Configured to display various ontologies like SCENE-O, Schema.org, and others, WebVOWL strengthens SCENE’s semantic search and retrieval capabilities by allowing visual, interactive exploration of ontological data.

The following table summarises WebVOWL’s role, configurations, and future enhancements in SCENE.

*Table 37. WebVowl's implementation details*

Component	Webvowl
<p><b>Associated Layer</b></p>	<ul style="list-style-type: none"> <li>• Discovery Layer (visualisation and interaction)</li> </ul>
<p><b>Function</b></p>	<ul style="list-style-type: none"> <li>• Provides an interactive graphical interface for visualising SCENE-O and other ontologies.</li> <li>• Allows users to explore ontology concepts, relationships, and metadata interactively.</li> <li>• Supports T3.1 by offering a visual tool for ontology-based workflows and data understanding.</li> </ul>



<p><b>Configuration</b></p>	<ul style="list-style-type: none"> <li>• Configured via docker-compose with SCENE-specific adaptations for look and feel.</li> <li>• Accessible through Nginx, using HTTPS to secure access. No authentication is required</li> <li>• Displays SCENE-O and related ontologies like Schema.org, EBUCorePlus, and Dublin Core. Other ontologies (At TTL files) can also be loaded and displayed</li> </ul>
<p><b>Role in SCENE</b></p>	<ul style="list-style-type: none"> <li>• Serves as the primary visualisation tool for exploring SCENE-O and related ontologies, providing insight into data structures and relationships.</li> <li>• Enhances SCENE’s semantic search capabilities by allowing users to interact with ontology structures visually.</li> <li>• Supports ontology-driven workflows by making SCENE-O accessible and navigable for users, aiding in semantic annotation and data retrieval.</li> </ul>
<p><b>Specific Goals and Requirements</b></p>	<ul style="list-style-type: none"> <li>• <b>Interactive Ontology Visualization:</b> Provides a user-friendly interface for exploring ontology relationships, nodes, and metadata.</li> <li>• <b>Scalable Ontology Support:</b> Displays SCENE-O alongside other relevant industry ontologies, adapting to SCENE’s evolving needs.</li> <li>• <b>Intuitive User Experience:</b> Ensures that even non-technical users can navigate complex ontology structures, improving the accessibility of semantic data.</li> <li>• <b>Compliance with Semantic Standards:</b> Configured to support widely accepted ontology standards, ensuring SCENE-O’s compatibility with existing knowledge graphs.</li> </ul>
<p><b>Interactions</b></p>	<ul style="list-style-type: none"> <li>• <b>Nginx:</b> Accessed through Nginx for secure HTTPS connections, ensuring secure visualisation of ontological data.</li> <li>• <b>SCENE-O (Ontology Layer):</b> Directly visualises SCENE-O and other selected ontologies, facilitating semantic understanding.</li> <li>• <b>Annotation Tools:</b> Enhances semantic workflows by displaying annotated data and facilitating the exploration of metadata and ontology tags.</li> </ul>
<p><b>Specific Configuration Relevant to T3.1</b></p>	<ul style="list-style-type: none"> <li>• <b>Customized Ontology Views:</b> Configured to display SCENE-O with custom SCENE-specific visuals, aligning with the project’s look and feel.</li> <li>• <b>Support for Industry Ontologies:</b> Configured to visualise other film-related ontologies like Schema.org, EBUCorePlus, and Dublin Core, broadening the understanding of SCENE-O’s interactions within the industry.</li> <li>• <b>Metadata and Annotation Display:</b> Supports visualisation of metadata and tags as part of the ontology structure, aiding in T3.1’s tagging and semantic annotation objectives.</li> </ul>
<p><b>Expected Outcomes</b></p>	<ul style="list-style-type: none"> <li>• <b>Enhanced Semantic Interaction:</b> Provides an interactive and visual means of exploring SCENE-O, supporting SCENE’s semantic search and retrieval goals.</li> <li>• <b>Improved Ontology Accessibility:</b> Ensures that SCENE-O and other ontologies are accessible in an intuitive, visual format, making it easier for users to navigate semantic relationships.</li> <li>• <b>Informed Data Retrieval:</b> By visually displaying ontology structures, WebVOWL aids in understanding data relationships, improving the accuracy and relevance of semantic searches.</li> <li>• <b>Ontology Consistency and Usability:</b> Facilitates consistent use of SCENE-O by providing an accessible, standardised visualisation tool that promotes proper tagging and annotation.</li> </ul>



<b>Future Enhancements</b>	<ul style="list-style-type: none"> <li>• <b>Dynamic Ontology Updates:</b> Plan to enable WebVOWL to update dynamically as SCENE-O evolves, ensuring the latest ontology version is always accessible.</li> <li>• <b>Extended Ontology Support:</b> Add more film-related or media ontologies as needed, expanding the scope of visualised data within SCENE.</li> <li>• <b>Enhanced User Interaction:</b> Develop additional interactive features, such as custom annotations and relationship highlights, to enhance the visual exploration experience.</li> </ul>
----------------------------	---

### 6.1.9. Jupyter Notebook Component Overview (Visualisation and Interaction Layer)

Jupyter Notebook is a collateral tool within SCENE’s Visualization and Interaction Layer, designed to support basic data exploration and analysis using examples. Configured with access to MinIO’s storage and other SCENE services, Jupyter allows users to interact with SCENE’s data lake programmatically. This setup is particularly useful for T3.1, as it provides a flexible environment for testing and developing scripts for semantic analysis, ontology alignment, and metadata tagging. Jupyter enables SCENE users to write and execute code for data retrieval, manipulation, and annotation through interactive notebooks, supporting SCENE’s semantic workflows and data processing tasks. Most likely data scientist will have their own environment, but this common tool allows fast sharing of examples to learn and test.

The following table summarises Jupyter Notebook’s configuration, role in SCENE, and potential enhancements.

*Table 38. Jupyter Notebook's implementation details*

Component	Jupyter Notebook
<b>Associated Layer</b>	<ul style="list-style-type: none"> <li>• Discovery Layer (visualisation and interaction)</li> </ul>
<b>Function</b>	<ul style="list-style-type: none"> <li>• Provides an interactive, code-driven data exploration, processing, and analysis environment.</li> <li>• Supports data retrieval, manipulation, and annotation workflows essential to T3.1 and SCENE’s semantic goals.</li> <li>• Enables rapid prototyping and testing of ontology-based queries and analysis scripts.</li> </ul>
<b>Configuration</b>	<ul style="list-style-type: none"> <li>• Configured via docker-compose, with port 8888 exposed for notebook access.</li> <li>• Environment set up with JUPYTER_ENABLE_LAB environmental variable for enhanced notebook experience.</li> <li>• Integrated with MinIO through a Python S3 client (boto3).</li> <li>• User token authentication is disabled to simplify local access; it can be integrated with Keycloak for enhanced security in production.</li> </ul>
<b>Role in SCENE</b>	<ul style="list-style-type: none"> <li>• Serves as the primary tool for providing examples of interactive data exploration and analysis with data in the data lake (e.g., Minio)</li> <li>• Provides SCENE users with a flexible data manipulation and visualisation environment, directly accessing MinIO for data needs.</li> <li>• Supports T3.1 by enabling the development of semantic tagging, annotation, and ontology alignment scripts that interact with SCENE’s data lake.</li> </ul>



<p><b>Specific Goals and Requirements</b></p>	<ul style="list-style-type: none"> <li>• <b>Data Analysis and Exploration:</b> Enables users to access and analyse large datasets from MinIO.</li> <li>• <b>Rapid Prototyping:</b> Supports the quick testing and development of scripts for data retrieval and ontology alignment.</li> <li>• <b>Secure Access Control:</b> Operates securely within SCENE’s subnet, with plans to integrate Keycloak for role-based access, if needed.</li> <li>• <b>Support for Scripting and Automation:</b> Facilitates automated data tagging, manipulation, and analysis scripts relevant to SCENE-O.</li> </ul>
<p><b>Interactions</b></p>	<ul style="list-style-type: none"> <li>• <b>MinIO:</b> Connects with MinIO for data retrieval and storage, allowing seamless interaction with SCENE’s data lake.</li> <li>• <b>Nginx:</b> Access managed through Nginx for secure, HTTPS-protected connections.</li> <li>• <b>Keycloak (Future):</b> Planned user authentication and role-based access integration, improving shared environment security.</li> </ul>
<p><b>Specific Configuration Relevant to T3.1</b></p>	<ul style="list-style-type: none"> <li>• <b>S3 API Integration:</b> Configured to access MinIO using S3-compatible API commands, enabling seamless data retrieval.</li> <li>• <b>Notebook Examples for Ontology Tasks:</b> Preconfigured with example notebooks for semantic annotation, ontology alignment, and metadata tagging, helping users quickly prototype T3.1-related scripts.</li> <li>• <b>Flexible Notebook Storage:</b> Volume mappings enable persistent notebooks and related file storage, supporting reproducibility and long-term project tracking.</li> </ul>
<p><b>Expected Outcomes</b></p>	<ul style="list-style-type: none"> <li>• <b>Enhanced Data Exploration:</b> Enables SCENE users to explore and analyse data interactively.</li> <li>• <b>Improved Data Retrieval for SCENE-O:</b> Supports direct access to SCENE-O and other ontologies, enhancing data manipulation and retrieval capabilities for semantic tasks.</li> <li>• <b>Secure and Reproducible Analysis:</b> Provides a secure, persistent environment for running analysis and prototyping, ensuring data consistency across SCENE’s tasks.</li> </ul>
<p><b>Future Enhancements</b></p>	<ul style="list-style-type: none"> <li>• <b>Keycloak Integration for User Access Control:</b> Secure role-based access via Keycloak to ensure only authorised users can interact with sensitive data.</li> <li>• <b>Containerized Execution for Scalability:</b> Consider containerised execution for running intensive data tasks within Kubernetes to scale with data demands.</li> <li>• <b>Additional Pre-Built Notebooks:</b> Add notebooks tailored for T3.1-specific tasks, such as advanced ontology alignment and semantic analysis functions.</li> </ul>

## 6.2. APIs, code and online documentation

### API Overview

This section overviews the APIs for key components used in SCENE’s Data Lake architecture. These APIs facilitate data management, ingestion, and analysis across SCENE’s infrastructure, supporting storage, data flow configuration, and querying functionalities.

- **1. MinIO (Storage Layer).** Main functionalities available through the API:
  - **Bucket Management API:** Endpoints for creating, listing, and deleting buckets within SCENE’s data storage layer.



- **Object Management API:** Allows uploading, downloading, listing, and deleting individual objects, with options to retrieve metadata for each object.
- **Access Control API:** Configures bucket- and object-level access policies, enabling secure access controls for different user roles.
- **Lifecycle Management API:** Manages policies for object lifecycle, allowing automated retention and deletion of obsolete data.
- API Documentation: [MinIO API Documentation](#)
- **2. Apache NiFi (Ingestion Layer).** Main functionalities available through the API:
  - **Flow Management API:** Endpoints for creating, modifying, and monitoring data flow pipelines, allowing administrators to configure complex ingestion flows.
  - **Processor API:** Interfaces to manage individual processors, enabling tasks like starting, stopping, or modifying settings for each step in a data flow.
  - **Remote Process Groups API:** Facilitates connecting and monitoring distributed instances, supporting SCENE's ability to scale ingestion across environments.
  - **Data Provenance API:** Allows retrieval of data lineage information for compliance and auditing, detailing data transformations and routing.
  - **API Documentation:** [Apache NiFi API Documentation](#)
- **3. Node-RED (Ingestion Layer).** Main functionalities available through the API:
  - **Flow Creation and Management API:** Endpoints for creating, modifying, and deleting data flows within the Node-RED environment, allowing users to define ingestion pipelines with minimal coding.
  - **Node Management API:** Enables control over individual nodes within a flow, including parameter adjustments and node status monitoring, providing flexibility for lightweight, event-driven data workflows.
  - **Deployment API:** Supports deployment and versioning of flows, allowing users to iterate on data pipelines and move from prototype to production-ready workflows.
  - **Data Injection and Control API:** Facilitates direct injection of data into flows via API calls, enabling integration with SCENE's other services for data streaming or triggering actions based on external events.
  - **API Documentation:** [Node-RED Documentation](#)
- **4. Trino (Analysis Layer).** Main functionalities available through the API:
  - **Query API:** SQL-based endpoint for executing analytical queries across the data lake, allowing users to retrieve data insights using a unified query language.
  - **Cluster Management API:** Manages query execution nodes, enabling scaling of resources to meet analysis demand across distributed datasets.
  - **Metadata API:** Provides access to schema information, supporting dynamic queries based on data lake structure, which can be useful for advanced analytics within SCENE.
  - **Session Management API:** Allows the configuration of session parameters, optimising queries by adjusting session variables like cache preferences and timeouts.
  - **API Documentation:** [Trino API Documentation](#)
- **5. MongoDB API (Discovery layer).** This is currently used for internal usage of data lake tools, such as the Operational Tools, but can also support metadata management in various ways:
  - **CRUD Operations API:** Standard endpoints for creating, reading, updating, and deleting metadata entries, enabling comprehensive management of media asset metadata in SCENE's data lake.
  - **Aggregation API:** Supports complex querying and aggregation of metadata, facilitating advanced searches and analyses based on criteria like asset type, creation date, or custom tags.



- **Index Management API:** Allows the creation and management of indexes for optimising metadata searches, ensuring efficient query performance even as metadata volumes grow.
- **Change Streams API:** Provides real-time notifications of metadata changes, supporting dynamic updates to SCENE's Media Asset Manager for improved data lineage and version control.

### Code Availability Statement

The project code for this deliverable, including the data lake, ontology, and all relevant configuration files and docker-compose setups, will be hosted at <https://github.com/benmomo/scene-datalake>. This repository can be accessed via the control panel in the SCENE Data Lake. It will be updated as the project evolves and the other SCENE tools of the whole SCENE architecture are implemented, integrated and deployed.

### Online Documentation

Ongoing documentation updates will be maintained to reflect project changes and evolution. An initial implementation is accessible at <https://scene-datalake-doc.readthedocs.io/en/latest/>, and it is also accessible through the Data Lake control panel. It will be updated as the project evolves and the other SCENE tools of the whole SCENE architecture are implemented, integrated and deployed.

## 7. Conclusions

This deliverable, D3.1 "Exploration of Data Lakes & Ontologies," lays the groundwork for the SCENE project's ambition to enhance the filmmaking industry's use of data lakes and ontologies, paving the way for innovative film production and distribution processes. Throughout this document, we have explored the state-of-the-art in data lakes and ontologies, providing a comprehensive review of both technologies within the context of the filmmaking industry. The outcomes of this exploration will serve as a reference for the design and development of the SCENE ontology and the broader SCENE platform.

Key points drawn from this deliverable include:

- **Data Lakes as a Key Enabler for the Filmmaking Industry:** Data lakes provide a robust and scalable solution for managing the diverse and voluminous data associated with filmmaking, from pre-production to distribution. By examining existing implementations, we identified best practices that ensure data lakes can efficiently handle the structured and unstructured data types prevalent in the industry, such as video footage, metadata, and audience data.
- **Ontologies to Bridge the Semantic Gap:** Exploring film-related ontologies highlights the need for a comprehensive and agnostic model to address the sector's varied requirements. The SCENE ontology, being developed under this task, is envisioned to unify disparate data sources through a semantic layer that will support enhanced searchability, discoverability, and interoperability of film assets, ultimately benefiting filmmakers and end-users.
- **Integration with the SCENE Architecture:** The findings and methodologies outlined in this deliverable directly influence other tasks within Work Package 3 (WP3), particularly Task 3.2 (Ontology Integration) and Task 3.3 (Media Asset Manager). The proposed solutions ensure that data lakes and ontologies will be seamlessly integrated into the broader SCENE platform, providing a strong foundation for the project's cognitive systems, AI-based tools, and knowledge graphs.
- **Challenges and Mitigation Strategies:** While data lakes and ontologies offer significant potential for the SCENE project, data governance, GDPR compliance, and metadata management must be carefully addressed. Our approach, which includes dynamic metadata annotation, modular API development, and scalable ontology frameworks, is designed to mitigate these risks and ensure smooth implementation in the project's next phases.



The following key steps will guide future work:

1. **Integration of SCENE Tools from WP3 and WP4:** The data lake and ontology need to be tightly integrated with the other tools developed in WP3, such as the Media Asset Manager (T3.3) and ontology expansion mechanisms (T3.2). The same applies to WP4 tools, including cognitive search modules, recommendation engines, and content licensing. A priority will be placed on ensuring smooth interoperability between these tools, leveraging SCENE-O's semantic annotation and knowledge graph features to maximise their functionality. The system's architecture must be continuously refined to allow these tools to seamlessly interact with the underlying data structures, metadata, and content housed in the data lake.
2. **Pilot Support and Evaluation:** As the project enters the pilot phase, supporting the integration of SCENE tools into real-world pilot environments will be essential. This will ensure that all SCENE modules, including AI-based tools and metadata management systems, are fully operational within the pilot ecosystems. The evaluation of these pilots will provide critical feedback on how well the system performs in various production scenarios, including content ingestion, management, distribution, and audience engagement. The pilot tests will be pivotal in refining the platform's capabilities and identifying areas for improvement.
3. **Optimization of APIs and Documentation for Developers:** With the data lake and ontology in place, providing comprehensive, user-friendly APIs will be a key priority. These APIs must facilitate interaction between the SCENE tools and the core platform, offering flexible data access, semantic search capabilities, and metadata tagging. Clear and detailed documentation will be provided to developers and end-users, ensuring they can easily understand and leverage the platform's capabilities. This will include didactic examples of utilising the APIs effectively, contributing to a smoother integration process during the pilot phase.
4. **Continuous Ontology Enhancement:** Although SCENE-O has been implemented, further refinement will be required as the pilots reveal new insights and requirements. The ontology must be continuously updated to reflect evolving user needs, emerging industry standards, and new data sources encountered during the pilots. These enhancements will ensure that SCENE-O remains a scalable and flexible framework supporting diverse filmmaking workflows, from pre-production to distribution.
5. **Cross-Pilot Data and Feedback Consolidation:** As the pilots progress, gathering and consolidating feedback from various use cases will be crucial. This feedback will inform the continuous improvement of both the data lake and the ontology, helping to ensure that they meet the practical needs of filmmakers, content managers, and other stakeholders. This iterative process will involve close collaboration with WP5, particularly in pilot evaluation and platform integration tasks.
6. **Ensuring GDPR Compliance and Data Privacy:** A focus will be placed on ensuring that all data handling processes comply with GDPR and other relevant data privacy regulations. This includes monitoring data flows within the data lake and ensuring that personal information is appropriately managed, especially as data from pilots is ingested and analysed.

The project will move towards a fully operational platform by prioritising the integration and optimisation of SCENE tools, supporting the pilot environments, and refining the ontology and data lake based on feedback. This will ensure that the SCENE ecosystem delivers its promise to revolutionise film production and distribution while preserving European cultural heritage.

## References

- [1] Grant Agreement No 101095303 – SCENE - Annex 1: Description of Innovation Action
- [2] HPC Systems, “Data Lake Curation and Governance with Tombolo”, white paper, 2022, Available: [https://cdn.hpcsystems.com/whitepapers/wp\\_data\\_lake\\_curation\\_and\\_governance\\_with\\_tombolo.pdf](https://cdn.hpcsystems.com/whitepapers/wp_data_lake_curation_and_governance_with_tombolo.pdf)
- [3] Unified Data Science, “Data lake design patterns on Google (GCP) cloud”, 2020, Available: <https://www.unifieddatascience.com/data-lake-design-patterns-on-google-cloud>
- [4] Data Lakes: GCP Solutions [Online]. Available: <https://flowygo.com/en/blog/data-lakes-gcp-solutions/> [Accessed Nov 2024]
- [5] Google Cloud, “Databases on Google Cloud part 2 - Options at a glance”, May 4 2022 [Online]. Available: <https://cloud.google.com/blog/topics/developers-practitioners/databases-google-cloud-part-2-options-glance/> [Accessed Nov 2024]
- [6] IBM, “Cloud data lake”, Available: <https://www.ibm.com/cloud/architecture/architectures/cloud-data-lake/> [Accessed Nov 2024]
- [7] IBM, “IBM watsonx.data on AWS”, Available: <https://developer.ibm.com/articles/awb-ibm-watsonx-data-aws/>
- [8] AWS Big Data Blog, “Enhancing customer safety by leveraging the scalable, secure, and cost-optimized Toyota Connected Data Lake”, 2020, Available: <https://aws.amazon.com/blogs/big-data/enhancing-customer-safety-by-leveraging-the-scalable-secure-and-cost-optimized-toyota-connected-data-lake/>
- [9] Amazon, “Data Lake on AWS”, Available: [https://aws.amazon.com/solutions/implementations/data-lake-solution/?nc1=h\\_ls](https://aws.amazon.com/solutions/implementations/data-lake-solution/?nc1=h_ls)
- [10] Microsoft Learn, “Data management across Azure Data Lake with Microsoft Purview”, Available: <https://learn.microsoft.com/en-us/azure/architecture/solution-ideas/articles/azure-purview-data-lake-estate-architecture>
- [11] Microsoft Learn, “Near real-time lakehouse data processing”, Available: <https://learn.microsoft.com/en-us/azure/architecture/example-scenario/data/real-time-lakehouse-data-processing>
- [12] Prasanna Kumar Illa, “Modern Unified Data Architecture”, 2020, Available: <https://towardsdatascience.com/modern-unified-data-architecture-38182304afcc>
- [13] Hoffart, J., Suchanek, F. M., Berberich, K., & Weikum, G. (2013). YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial intelligence*, 194, 28-61.
- [14] Sejwal, V. K., & Abulaish, M. (2021). CAMO: A context-aware movie ontology generated from LOD and movie databases. *Multimedia Tools and Applications*, 80(5), 7247-7269.
- [15] Cuenca Grau, B., & Jimenez-Ruiz, E. (2009). *Ontology Integration Using Mappings: Towards Getting the Right Logical Consequences*.
- [16] Shvaiko, P., & Euzenat, J. (2011). *Ontology matching: state of the art and future challenges*. *IEEE Transactions on knowledge and data engineering*, 25(1), 158-176.
- [17] Overton, J. A., Dietze, H., Essaid, S., Osumi-Sutherland, D., & Mungall, C. J. (2015, July). *ROBOT: A command-line tool for ontology development*. In *ICBO*



- [18] Schober, D., Smith, B., Lewis, S. E., Kusnierczyk, W., Lomax, J., Mungall, C., ... & Sansone, S. A. (2009). Survey-based naming conventions for use in OBO Foundry ontology development. *BMC bioinformatics*, 10, 1-9.,
- [19] Hu, W., & Qu, Y. (2008). Falcon-AO: A practical ontology matching system. *Journal of web semantics*, 6(3), 237-239.
- [20] Jiménez-Ruiz, E., & Cuenca Grau, B. (2011). Logmap: Logic-based and scalable ontology matching. In *The Semantic Web–ISWC 2011: 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I 10* (pp. 273-288). Springer Berlin Heidelberg.
- [21] Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I. F., & Couto, F. M. (2013). The agreementmakerlight ontology matching system. In *On the Move to Meaningful Internet Systems: OTM 2013 Conferences: Confederated International Conferences: CoopIS, DOA-TrusteD Cloud, and ODBASE 2013, Graz, Austria, September 9-13, 2013. Proceedings* (pp. 527-541). Springer Berlin Heidelberg.
- [22] Faria, D., Pesquita, C., Santos, E., Cruz, I. F., & Couto, F. M. (2014, July). AgreementMakerLight: A scalable automated ontology matching system. In *Proceedings of the 10th International Conference on Data Integration in the Life Sciences (DILS 2014)* (pp. 29-32).
- [23] Tang, X., Zhang, J., Chen, B., Yang, Y., Chen, H., & Li, C. (2020). BERT-INT: A BERT-based interaction model for knowledge graph alignment. *interactions*, 100, e1.
- [24] Qi Zhu, Hao Wei, Bunyamin Sisman, Da Zheng, Christos Faloutsos, Xin Luna Dong, and Jiawei Han. 2020. Collective Multi-type Entity Alignment Between Knowledge Graphs. In *WWW. ACM/IW3C2, Taipei, Taiwan, 2241–2252*
- [25] A. Joulin, “Bag of Tricks for Efficient Text Classification,” 2015
- [26] Chen, J., & Yan, R. (2024, May). EMGCN: Enhancement Graph and Multi-head Attention Graph Convolutional Networks for Aspect-based Sentiment Analysis. In *2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (pp. 1591-1596). IEEE.
- [27] Zhang, Q., Sun, Z., Hu, W., Chen, M., Guo, L., & Qu, Y. (2019). Multi-view knowledge graph embedding for entity alignment. *arXiv preprint arXiv:1906.02390*.
- [28] Huang, J., Sun, Z., Chen, Q., Xu, X., Ren, W., & Hu, W. (2023). Deep active alignment of knowledge graph entities and schemata. *Proceedings of the ACM on Management of Data*, 1(2), 1-26.
- [29] Ngo, D., & Bellahsene, Z. (2012). YAM++: A multi-strategy-based approach for ontology matching task. In *Knowledge Engineering and Knowledge Management: 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings 18* (pp. 421-425). Springer Berlin Heidelberg.
- [30] Hu, W., & Qu, Y. (2008). Falcon-AO: A practical ontology matching system. *Journal of web semantics*, 6(3), 237-239
- [31] Gómez-Pérez, A., Fernández-López, M., & Corcho, O. (2004). *Ontological engineering: with examples from the areas of knowledge management, e-commerce and the semantic web*. Springer Science & Business Media.
- [32] Sure, Y., Staab, S., & Studer, R. (2002). On-To-Knowledge methodology (OTKM). In *Handbook on Ontologies* (pp. 117–132). Springer.
- [33] Uschold, M., & King, M. (1995). Towards a methodology for building ontologies. *IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing*



- [34] Suárez-Figueroa, M. C., Gómez-Pérez, A., & Villazón-Terrazas, B. (2012). The NeOn Methodology for ontology engineering. In *Ontology engineering in a networked world* (pp. 9–34)
- [35] Gangemi, A. (2005). Ontology design patterns for semantic web content. *Proceedings of the 4th International Semantic Web Conference*
- [36] Noy, N., & Klein, M. (2004). Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, 6(4), 428–440
- [37] Klein, M., & Fensel, D. (2001). Ontology versioning on the Semantic Web. *Proceedings of the International Semantic Web Working Symposium*



## Annexes

### A. Data Management Tools and Methods

There are a bunch of different concepts related to the data management domains, which can be separated into two different domains:

- Tools
  - Data Lake
  - Data Warehouse
  - Data Lakehouse
- Methods
  - Data fabric
  - Data mesh

#### i. Tools

Considering the previous definitions (descriptions), data lakes and data warehouses are probably the most confusing terms due to potential conceptual overlap. A comparison table is provided below.

Table 39. Data lake vs data warehouse

Data lake (staging & preparation)	Aspect	Data warehouse (Serving, security & compliance)
Structured, semi-structured, unstructured, raw/binary, Batch processing, Data refinement/cleaning	<b>Data</b>	Structured, processed Low latency Complex joins
Schema-on-read	<b>Processing</b>	Schema-on-write
Designed for Low-cost storage (store older/backup data)	<b>Storage</b>	Expensive for large data volumes
Highly agile, reconfiguration feasible	<b>Agility</b>	Less agile, reconfiguration is typically not feasible (fixed)
Maturing (relatively new tech)	<b>Security</b>	Mature technology
Data scientists (sandbox for data exploration)	<b>Users</b>	Business professionals (Dashboards)

Data lakes can be helpful under different generic scenarios, as presented in Table 40

Table 40. General use cases for data lakes

Scenario	Description
<b>Support for traditional and emerging data sources</b>	Analyse in real-time information (massive data) from various origins: customer, demographics, employee, social data, external data, etc.
<b>Data exploration sandbox</b>	Capacity to retrieve specific datasets and attributes from the whole data store through filtering techniques, to be later used in analytic tools
<b>Extended customer analysis</b>	Analysis of user behaviour through all their possible interactions (coming from various sources)

<b>Centralised Discovery</b>	Single centralised discovery engine for discovering and managing published datasets
------------------------------	---

### ii. Methods or Models

Considering the models (methods), **data mesh** and **data fabric** are two related concepts, but they serve slightly different purposes and have distinct characteristics. A data mesh can be seen as an approach to implementing certain aspects of a data fabric. In practice, organisations can **adopt both concepts** in conjunction. A data fabric can provide the technical foundation for data integration, while a data mesh can define the organisational structure and responsibilities for managing and delivering data products. Another comparison table is provided below.

Table 41. Data mesh vs data fabric

Data mesh	Aspect	Data Fabric
Promotes a culture of data ownership and collaboration	<b>Culture</b>	Focuses on unified data management and integration
Domain teams own and manage their data as a product	<b>Data Ownership</b>	Centralised ownership and management of data
Decentralised integration, standardised APIs	<b>Data integration</b>	Centralised integration, abstraction layer
Often includes a data catalogue for data discovery	<b>Data Catalogue</b>	May include a data catalogue for unified data access
Distributed governance, domain-specific policies	<b>Governance</b>	Centralised governance policies and enforcement
Data Products owning the domain and applying security	<b>Security</b>	Focuses on a comprehensive, unified security model across the entire data ecosystem
Relies on domain teams to maintain data consistency	<b>Consistency</b>	Centralised mechanisms ensure data consistency
Complex, even a small implementation, as it requires understanding and segregating domain data	<b>Implementation</b>	Simpler, due to the inherent use of data virtualisation, metadata and knowledge graphs (centralised management can simplify certain aspects)

### iii. Data Lakes and its evolution towards data mesh

Another aspect to reflect on is the relationship between **data lakes** and **data mesh**. Initially, when **Big Data** started to be hyped, there was a general market trend to store all sorts of data in a common **centralised repository** (e.g., data lake) and later check their **potential** through analytics (see Figure 20).

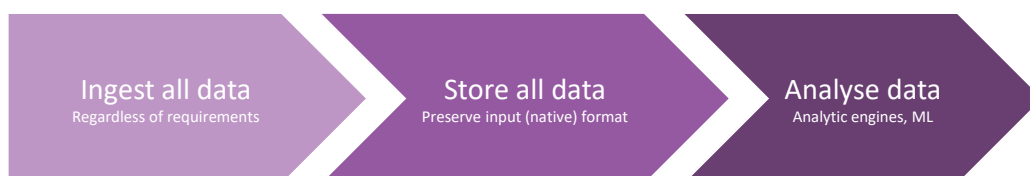


Figure 20. General data flow for data lakes

Though it seemed a good idea, there were some drawbacks. As data volumes grew, so did the complexity of managing these lakes, leading to **data silos**, inconsistent **data quality**, and difficulties in deriving **meaningful insights**.



From a **data mesh** perspective, it is a paradigmatic shift that recognises the need for a more **decentralised, domain-oriented** approach. Unlike the centralised model of data lakes, data mesh decentralises data ownership, empowering individual domain teams to treat data as a product. This transformation is driven by the recognition that domain experts **understand their data best** and can **manage** it effectively. By fostering a **culture of ownership, accountability, and collaboration**, data mesh not only resolves the challenges of data lakes but also unleashes the potential of diverse, high-quality data. It allows organisations to **adapt swiftly** to changing business needs, democratise data access, and foster innovation by enabling cross-functional teams to collaborate, seamlessly transforming raw data into valuable insights. This shift is not just technological but also cultural, acknowledging that the future of data lies in the hands of those who understand it most intimately – the experts within the domains.

Data mesh can be seen as an **evolution** of data lakes, not a replacement. This means that data lakes already have their meaning and usefulness as a first phase, **facilitating** the way towards the data mesh, which should be considered a medium or long-term step. The **SCENE project** is focussed on data lakes and this first phase, considering the immature (technological) status of the film-making industry, at least for most of the main stakeholders with limited IT knowledge and/or resources. Additionally, **data lakes pave the way towards data mesh** by identifying the different domains and subdomains for data treatment and their interactions across the whole film pipeline (pre-production, production, post-proc, distribution). Moreover, considering the **evolution** of data approaches in the last 40 years (see Figure 21), it seems more sensible to focus on a **data lakehouse** as an **extension** of a data lake, still centralised and conceptually more similar to a data mesh.

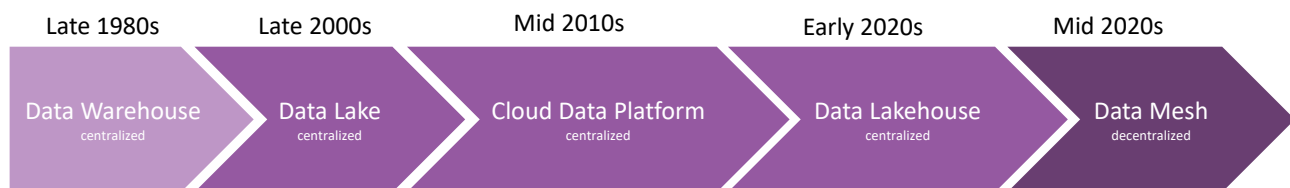


Figure 21. Evolution from centralised to decentralised data management

## B. Other data architectures

### i. Data warehouse

A general overview of a data warehouse is depicted in **Figure 22**. In large companies and organisations, the term **Enterprise Data Warehouse (EDW)** is also used. All input data sources undergo an **ETL (Extract-Transform-Load) process** to ensure that the data is adequately reformatted (here adequately means that the destination format is fixed and the input data might need adaptation, whether by changing the format or merging various input sources into another format for storage). The communication with the core of the data warehouse is typically through an **API**, both for the input data sources and the target users. Once data is inserted correctly in the EDW **storage layer** via the **ETL tools**, they can be employed by different users in different ways:

- **Built-in analytics**
- **Business intelligence (BI) tools**
- **Predictive Analytics and Machine Learning (PAML) tools**

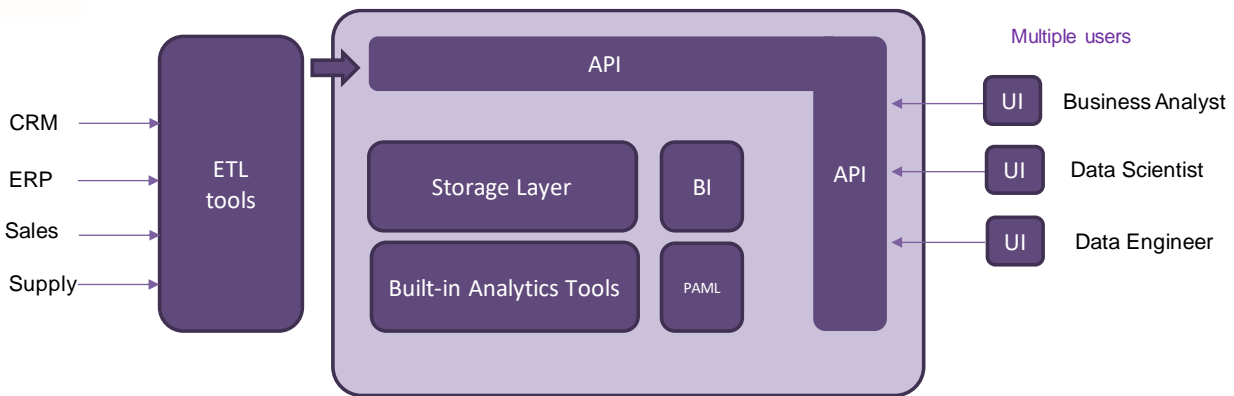


Figure 22. Data warehouse architecture

EDWs can be deployed in various ways:

- *On-premises: via either commodity hardware or appliance from a single manufacturer.*
  - **Pros:** complete control, leverage network speed and high availability, strict governance.
  - **Cons:** upfront investment, complex maintenance
- *Cloud-based (SaaS):*
  - **Pros:** resource-free, focus on analytics, more effortless scalability, automatic upgrades.
  - **Cons:** cost
- *Hybrid: best of both approaches and quite a common approach.*
  - **Pros:** allows to study new use cases, Disaster Recovery (DR) and backup scenarios
  - **Cons:** requires a clear strategy for data (on-premises vs cloud)

## ii. Data lakehouse (Delta Lake)

A [delta lake](#) is an open-source storage framework that sits on top of a data lake and integrates with many tools (see Figure 23). As a reminder, a data lakehouse combines the best of data lakes and data warehouses, supporting Business Intelligence (BI) and Machine Learning (ML) on all data.

This open-source **ACID (Atomicity, Consistency, Isolation and Durability) table layer** of the delta lake over cloud object stores was initially developed at Databricks (a relevant company founded by the creators of Apache Spark) with the primary aim of providing **efficiency** and **reliability** to existing machine learning operations using data lakes and other data science use cases. The main features of delta lakes are:

- *ACID transactions:* Data concurrency control, data integrity protection
- *Metadata scalability:* usage of Spark for “big data” metadata
- *Time travel:* data versioning, enables rollbacks, audit trail
- *Open format:* Usage of Apache Parquet format
- *Streaming and batch unification:* delta table can be batch table, source or target stream
- *Schema enforcement and evolution:* ensuring data type correctness
- *Updates, inserts and deletes (MERGE):* supports Scala / Java / Python APIs
- Performance improvement

One drawback of this implementation (delta lake) of Databricks (provider) is the license and installation aspect, especially if you want to deploy it (completely) in your own infrastructure. Databricks perspective is cloud-oriented and pushes organisations to migrate to the cloud by providing packages for the major cloud providers (Microsoft Azure, Amazon AWS and Google Cloud). The technological approach is sometimes called hybrid SaaS, where the vendor manages the control plane, whereas the data plane is in the customer’s environment as an IaaS.

In other words, whereas the open-source version of Databricks offers only two principal components (Apache Spark and Delta Lake Core), the hybrid SaaS solution offers the whole set by adding:

- Photon engine, a query-performant native engine
- Unity catalogue as a unified governance solution component
- Access control component supporting ACLs to configure different permission types

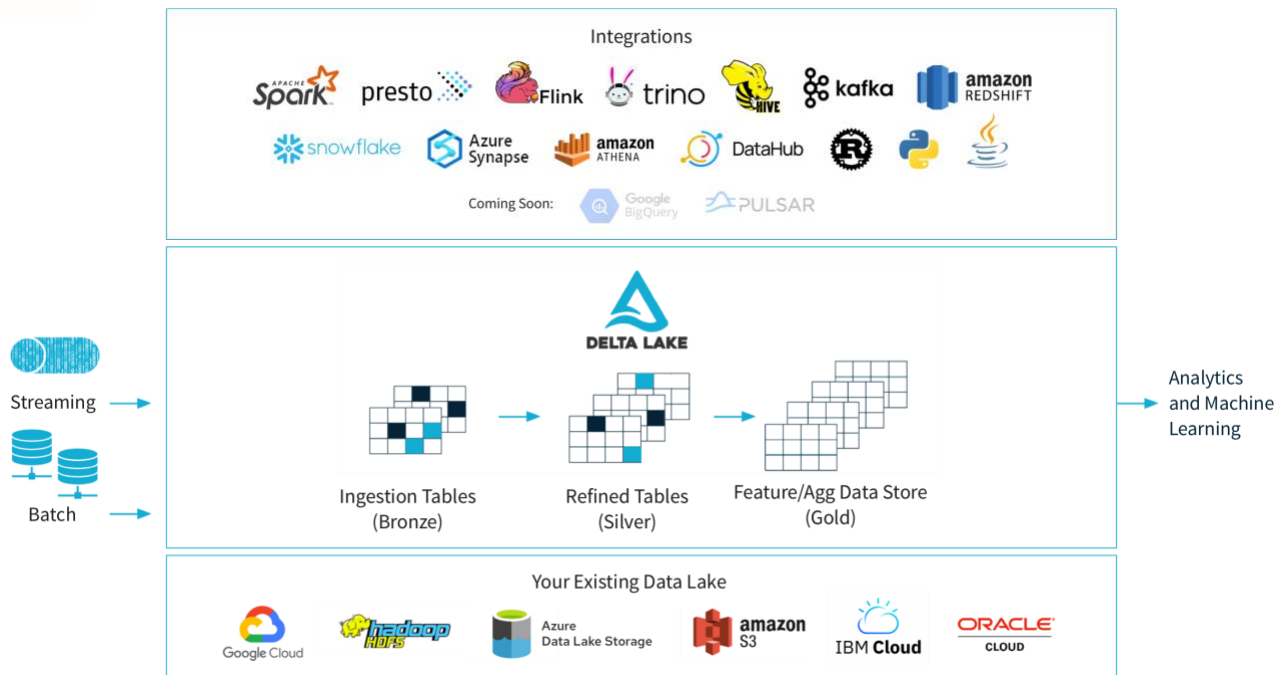


Figure 23. Delta lake<sup>2</sup>.

### iii. Data mesh

Zhamak Dehghani<sup>3</sup> coined the term [data mesh](https://delta.io/), focusing more on organisational changes and decentralising the **data team** so it does not become a significant bottleneck (domain knowledge is decentralised). Though there is no reference standard for this approach, a helpful architecture overview that is easy to understand is provided by [data-mesh-architecture.com](https://data-mesh-architecture.com). The data team, expert on a specific domain, is responsible for analysing operational data and building data products that can be used internally or **published** to be consumed by **other domains**. The **governance layer** allows all data teams to agree on globally interoperable standards, security and privacy policies, and documentation standards in a federated way so that (data) products are easily discovered, understood through complete documentation, and used. The **self-serve data platform** provides tools to build the data products effectively. Additionally, an **enabling team** might help in the modelling and analysis procedure.

<sup>2</sup> Source: <https://delta.io/>

<sup>3</sup> “Data Mesh: Delivering Data-Driven Value at Scale”, Zhamak Dehghani, 2022, O’reilly

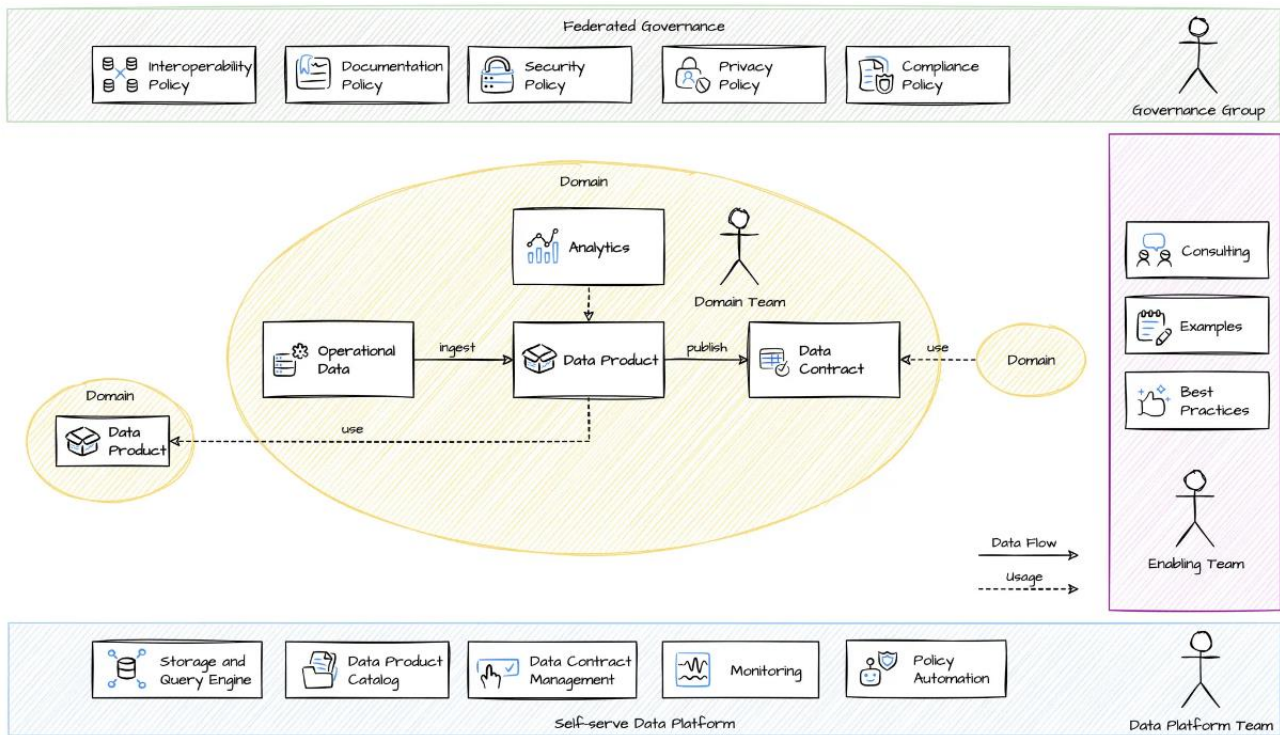


Figure 24. Data mesh architecture<sup>4</sup>.

Regarding task T3.1, the two most relevant aspects are **data products** and **data contracts**. A **data product** is similar to a **microservice** with its API, but for analytical data (it allows the gathering of all data needed for a particular purpose, connecting to sources and outputting the data in a specific format). Data products can be designed with a [Data Product Canvas](#). A data contract is a **document** establishing structure, format, **semantics**, quality (e.g., freshness) and terms of use. Through the semantics, **data attributes (metadata)** can be included to facilitate discoverability and interoperability. A [Data Contract Specification](#) can be used in YAML format.

#### iv. Data fabric

A data fabric is an **architectural approach** (not a tool or product) consisting of a set of technologies aiming at breaking down data silos and **getting data into the hands of data users**. It encompasses accessing, ingesting, integrating and sharing data in a governed way regardless of location (on-premises, cloud-based or hybrid). Simply put, it **connects data producers with data consumers** within the organisation. A data fabric is grounded on five fundamental pillars:

- **Self-serve**: simple data access mechanisms for a large variety of tools (from Excel to PowerBI and PAML)
- **Augmented knowledge**: based on **smart catalogues** (information, policies, security).
- **Smart integration**: helping integrate data for analytics.
- **Governance and security**: a framework for applying rules and policies to protect the data
- **Unified Lifecycle**: from data ingestion until data exposure, simply and securely.

The main building blocks of a data fabric are depicted in **Figure 25**, connecting data producers and consumers. Data can come from multiple sources, such as Enterprise Data Warehouses (EDW), data lakes, specific databases, various applications, ETL tasks and even data meshes. An abstracted (virtualised) layer is typically built to homogenise access and avoid duplicating data, which can use intelligent and efficient caching. As part of the **Lifecycle management**, the **governance layer** covers various different topics, such as:

- Data discovery (newly added datasets should be easily discovered)

<sup>4</sup> Source: [Data Mesh architecture](#). Data Mesh from an Engineering perspective

- Data quality (ensuring correctness through the pipeline)
- Data privacy (e.g., implementing Role Based Access Control, RBAC)
- Use of metadata and business vocabularies/glossaries (so that business rules might apply up to the data registry level).
- Data lineage (tracking the origin of the data when building new AI models)
- Master Data Management (MDM, for products, services and contracts)
- Compliance with both legal and industrial policies (e.g., GDPR for personal data,

All this information is incorporated in an enterprise catalogue or **knowledge catalogue**, placed at the **heart** of the data fabric. The **egress layer** (data exposure) exposes different **APIs** (e.g., REST API) to access the catalogued data. Still, it should also provide additional integration with other **open-source** (OS) tools already in place, such as MongoDB, Spark, etc. Additional **trustworthy AI integration** is also provided to support MLOps, fairness and explainability purposes. Through all this exposure layer, data consumers (business analysts, data scientists and applications developers) can work and exploit the data in an easy, unified and agile way.

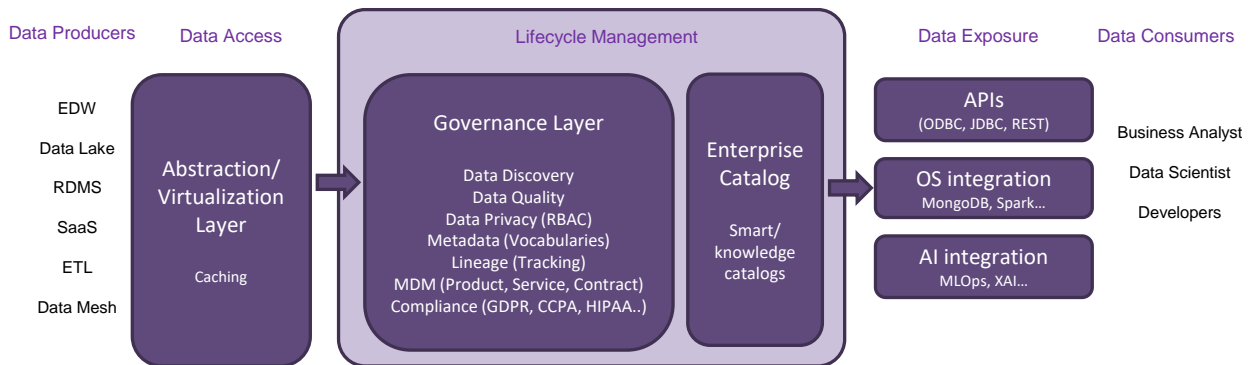


Figure 25. Data fabric architecture

## C. Introduction to semantic concepts

### i. Ontology Overview

Ontology, originating from philosophy, concerns the study of being and existence. It was integrated into computer science by AI researchers in the 1980s for building computational models and enabling automated reasoning. In the 1990s, ontologies became standard components for knowledge systems to address **interoperability issues** among emerging technologies.

In computer science, an ontology specifies a **shared conceptualisation**, defining **concepts** and **relationships** within a **domain**. This involves a set of classes (concepts) and properties (attributes of classes). The ontology spectrum ranges from less structured models like taxonomies and database schemas to highly structured models like logical theories.

Ontologies are created by teams consisting of **domain experts**, who provide domain-specific knowledge, and **modellers/ontologists**, who build the semantic structures and integrate them with other domains.

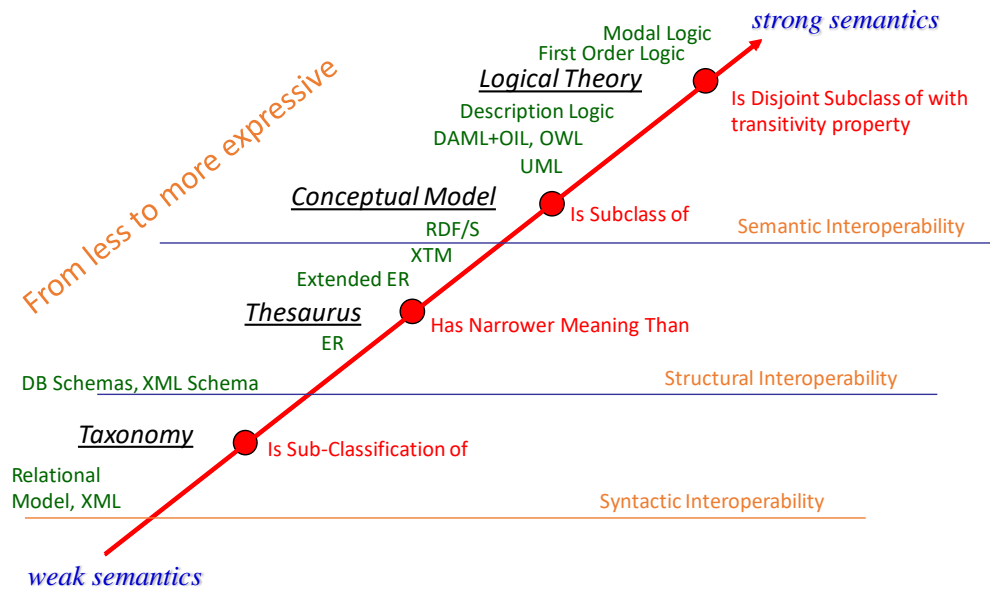


Figure 26. Ontology spectrum<sup>5</sup>. One view.

The primary advantage of ontologies is that they **enhance intelligent searches**, allowing **software agents** to understand and interpret data meaningfully. This capability supports **advanced data search**, integration, and decision-making processes.

For machine interpretability, logical theories (strong ontologies) are employed beyond human-understood conceptual models. These can be **frame-based**, focusing on entities and their properties, or **axiomatic**, focusing on rules and axioms about entities.

## ii. RDF as the basis for the semantic web

The **Resource Description Framework (RDF)** forms the foundation of the semantic web, working alongside **RDFS** and **OWL** to manage distributed data. RDF structures data in triples, each consisting of a **subject**, **predicate**, and **object**, which can form complex graphs linked by **URIs** (Uniform Resource Identifiers).

Table 42. Triple examples using Compact URIs (CURIEs)

Subject	Predicate	Object
geo:Valencia	geo:partOf	geo: Spain
eu-prj: SCENE	eu-prj:acceptedProjectFromTopic	eu-prj: HORIZON-CL2-2022-HERITAGE-01-06
eu-prj: SCENE	eu-prj:hasPartner	edu:UPV
eu-prj: SCENE	eu-prj:hasPilot	scene: AthensPilot

The W3C has defined several **standard namespaces** for use with Web technologies, and so does the semantic web. Some examples are listed in Table 43.

<sup>5</sup> Source: D. L. Obrst, "[The Ontology Spectrum & Semantic models](#)," MITE Corporation, 2006

Table 43. Common W3C namespaces

Namespace	Description
rdf	RDF Identifiers The global URI for the rdf namespace is <a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a> . <b>Examples:</b> <code>rdf:type</code> , <code>rdf:Property</code>
rdfs	RDFS Identifiers The global URI for the rdfs namespace is <a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a> . <b>Examples:</b> <code>rdfs:Class</code> , <code>rdfs:SubClassOf</code> , <code>rdfs:subPropertyOf</code> , <code>rdfs:domain</code> , <code>rdfs:range</code> , <code>rdfs:label</code>
skos	Simple Knowledge Organization System (SKOS) Identifiers. It is used for the management of vocabularies on the Web. The global URI for SKOS is <a href="http://www.w3.org/2004/02/skos/core#">http://www.w3.org/2004/02/skos/core#</a> <b>Examples:</b> <code>skos:prefLabel</code> , <code>skos:altLabel</code> , <code>skos:hiddenLabel</code> , <code>skos:borader</code> , <code>skos:narrower</code> , <code>skos:related</code> , <code>skos:narrowerTransitive</code> , <code>skos:broaderTransitive</code> , <code>skos:exactMatch</code> , <code>skos:narrowMatch</code> , <code>skos:boradMatch</code> , <code>skos:closeMatch</code>
owl	OWL Identifiers The global URI for the OWL namespace is <a href="http://www.w3.org/2002/07/owl#">http://www.w3.org/2002/07/owl#</a> <b>Examples (RDFS-Plus):</b> <code>owl:inverseOf</code> , <code>owl:SymmetricProperty</code> , <code>owl:TransitivityProperty</code> , <code>owl:equivalentClass</code> , <code>owl:equivalentProperty</code> , <code>owl:sameAs</code> , <code>owl:FunctionalProperty</code> , <code>owl:InverseFunctionalProperty</code> , <code>owl:DatatypeProperty</code> , <code>owl:ObjectProperty</code> , <code>owl:Class</code>

RDF data can be displayed in various formats:

- **N-Triples:** raw RDF triples using their fully (not abbreviated) URI. For names graphs, there are *N-Quads*.
- **Turtle/N3:** compact serialisation of RDF by defining prefixes in the preamble (turtle uses CURIEs). It improves (human) readability and includes additional abbreviations (e.g., `rdf:type` changes to `'a'`). For named graphs, *TriG* is an extension of *Turtle*.
- **RDF/XML:** Intended for web infrastructures using HTML and XML to represent information
- **JSON-LD (JSON for Linked Data):** The W3C recommended this format to make linked data in RDF more available to applications that use JSON (direct mapping between JSON-LD and RDF triples). Thus, developers can build applications purely in JSON.

RDF can be extended for enhanced modelling capabilities, such as **inference** (drawing new data from existing data) using **RDFS**, **RDFS-Plus**, and **OWL**, and expectation (predicting unseen data) using **SHACL** (Shapes Constraint Language).

### iii. Main Building Blocks in Semantic Applications

Typically, a semantic application includes:

- **RDF Parser:** Reads and interprets RDF formats.
- **RDF Store:** Stores and retrieves data in triple form.
- **RDF Query Engine:** Retrieves RDF data via structured queries (SPARQL).
- **Application:** Processes RDF data.
- **Reasoning Engine:** Infers logical consequences from RDF data.



Besides the format describing how data is embedded, working with some shared vocabulary is essential. **Schema.org** is a collaborative initiative that provides a standardised vocabulary for structuring data on websites, allowing search engines and applications to understand the content better. It is linked to semantic web applications by enabling more precise data interpretation, facilitating interoperability between systems, and improving the visibility and relevance of web content in search results.

#### iv. Linked Data

Linked data refers to connecting datasets across the web, creating an interconnected web of data. Unlike static data on the web, such as a PDF file, linked data utilises RDF to describe data, including links to other data descriptions.

Given the vast number of possible schemas on the web, tools like **Linked Open Vocabularies (LOV)** provide directories and search functionalities to help discover these schemas. LOV supports the discovery of various vocabularies that facilitate the integration and linking of data.

In line with REST (Representational State Transfer) principles for web services, the W3C launched the **Linked Data Platform (LDP)** in 2015. LDP defines the guidelines for web applications to create, modify, and remove data resources via the HTTP protocol. It differentiates between RDF and non-RDF resources and includes containers (basic, direct, indirect) to effectively organise and manage these resources.

#### v. Simple Knowledge Organization System (SKOS)

**SKOS** is a **W3C Recommendation** for sharing and linking knowledge systems on the web, enhancing thesauri and taxonomies with the distributed nature of the semantic web.

#### vi. Web Ontology Language (OWL)

**OWL** is a language for the semantic web, providing classes, properties, individuals, and data values, supporting complex relationships and restrictions. OWL 2.0 includes profiles like OWL 2 EL, OWL 2 QL, and OWL 2 RL for various technological compatibilities.

#### vii. Common Practices for Semantic Modelling

Generally speaking, there are three ways to proceed when modelling:

- *Reuse existing (semantic) models available on the web.* There are hundreds of models on the web that might be helpful or at least inspiring.
- *Identify the information assets that have real value for your company, project or application.* In the case of a consolidated company, there are typical procedures to be followed that probably share a controlled vocabulary.
- *Engineer a model from scratch.* This is typically the case where your domain is too specific or the available related semantic models are either non-existent or too complex.

Modelling in the field of the Semantic Web goes beyond the usual design of an engineered component with system requirements. A model is expected to be merged with other models in the former. This implies targeting not only known requirements but also anticipating (to some extent) future usage potentials. Current and future requirements can be identified using motivating scenarios and competency questions.

The motivating scenarios and competency questions are valuable tools to shape the ontology in terms of the following:

- *Completeness* (to which extent does the ontology cover the scenario?)
- *Specificity* (how many specific aspects of the scenario are captured?)
- *Granularity* (how precisely are the terms in the scenario defined?)
- *Formality* (to which extent are definitions formalised?), and



- *Reusability* (how can the given ontology be used in other related scopes?)

While developing the ontology, it is recommended to follow naming conventions used in RDFS and followed by the W3C.

### viii. Semantic benefits

Contrary to syntactic searches, **semantic search engines aim to understand what users mean** rather than what they type. This meaning refers to determining the **intent** and **contextual meaning** of the words a person uses for searching. Needless to say, by using ontologies and software, agents can directly interact with a semantic search engine without any user intervention and make their own decisions. In general, there are various theoretical/potential benefits of using semantic searches:

- **Results of better accuracy:** contrary to the syntactic world, where the user is limited to the data model behind the system and can only apply filters that somehow match specific data model fields, a semantic model covers a wider range of possibilities. By introducing meaning, it is possible to describe an entity better and define what it does, what it does not do, how it links to other entities, how it can be a member of various categories, etc. In practical terms, semantic filters are more powerful than a syntactic search engine.
- **Improved user experience when combined with NLP:** NLP allows the user to express himself naturally. In this regard, the system adapts to the user, not the other way (traditional syntactic way). The NLP must be combined with semantics (taxonomy and/or ontology) to exploit its full potential. For example, ambiguous terms can be resolved via a Knowledge Graph. Different knowledge bases (e.g., *WikiData*, *GeoNames*, *DBPedia*) can be linked to enrich the base vocabulary.
- **Capacity to model any specific area of interest:** Using ontologies, it is possible to model untargeted areas and give meaning only to that particular scope. They can be created in a modular way and can even be complementary (they are not exclusive) to existing syntactic engines.

To clarify the semantic benefits, we will provide a basic example. Suppose a film producer is looking for a specific location in Athens with a wooden house, garden, and forest nearby (location scouting activity). This is precisely what they may insert in a search box:

---

I'm looking for a wooden house with a garden and a forest nearby in Athens

---

An NLP module will analyse the sentence and, broadly speaking, it will come up with 3 aspects:

**The user is looking for a house (action):** the user wants to find a particular item or object. Several synonyms might have also been applied here (e.g., search, need).

**The searched object relates to a house:** the services or datasets provided should refer to residential houses (not hospitals, universities, etc) with particular attributes (made of wood, with a garden, near a forest).

**The data is georeferenced to Athens:** by linking this value with GeoNames it is possible to determine the target bounding box.

## D. Semantics and Ontologies in the Film-making Industry

### i. Schema.org

Below is a table with potential concepts from schema.org that directly relate to the film-making industry and could be re-used.

Table 44. Potential re-usable entities and properties from schema.org

Entity	Definition and Key Properties
<a href="#">Movie</a>	<p><b>Definition:</b> A work of visual storytelling, usually produced for cinema, TV, or streaming platforms</p> <ul style="list-style-type: none"> <li>• <b>name:</b> The movie's title.</li> <li>• <b>director:</b> The director of the movie (Person entity).</li> <li>• <b>actor:</b> Actors who appeared in the movie (Person entity).</li> <li>• <b>genre:</b> The genre of the movie (e.g., "Action", "Comedy").</li> <li>• <b>datePublished:</b> The release date of the movie.</li> <li>• <b>duration:</b> The movie's duration, typically in ISO 8601 format.</li> <li>• <b>productionCompany:</b> The production company responsible for the film.</li> <li>• <b>trailer:</b> A trailer or teaser for the movie (VideoObject entity).</li> <li>• <b>aggregateRating:</b> Aggregated user ratings for the movie (AggregateRating entity).</li> </ul>
<a href="#">TVSeries</a>	<p><b>Definition:</b> A serialised work of fiction or non-fiction broadcasted over television or streaming services</p> <ul style="list-style-type: none"> <li>• <b>name:</b> The name of the TV series.</li> <li>• <b>creator:</b> The person or organisation responsible for creating the series.</li> <li>• <b>actor:</b> Main actors who appeared in the series.</li> <li>• <b>numberOfEpisodes:</b> Total number of episodes in the series.</li> <li>• <b>startDate</b> and <b>endDate:</b> The date range during which the series was aired.</li> <li>• <b>trailer:</b> A teaser or trailer for the series.</li> </ul>
<a href="#">Person</a>	<p><b>Definition:</b> Individuals involved in film production or acting, such as directors, actors, producers, etc.</p> <ul style="list-style-type: none"> <li>• <b>name:</b> The name of the person.</li> <li>• <b>birthDate:</b> The birth date of the individual.</li> <li>• <b>jobTitle:</b> Role or occupation, e.g., "Director", "Actor", "Producer".</li> <li>• <b>worksFor:</b> The organisation or company they work for.</li> <li>• <b>sameAs:</b> Links to the person's profiles on external websites like IMDb, Wikipedia, or social media.</li> </ul>
<a href="#">Organization</a>	<p><b>Definition:</b> A legal entity representing a production company, distributor, or studio</p> <ul style="list-style-type: none"> <li>• <b>name:</b> The organisation's name.</li> <li>• <b>location:</b> The geographical location of the organisation.</li> <li>• <b>foundingDate:</b> The date the organisation was founded.</li> <li>• <b>sameAs:</b> Links to official websites or external sources (e.g., IMDb or Wikipedia).</li> <li>• <b>logo:</b> A logo image for the organisation.</li> </ul>
<a href="#">Event</a>	<p><b>Definition:</b> An event related to film, such as premieres, film festivals, or award shows</p> <ul style="list-style-type: none"> <li>• <b>name:</b> The name of the event (e.g., "Academy Awards").</li> <li>• <b>startDate</b> and <b>endDate:</b> The date of the event.</li> <li>• <b>location:</b> The place where the event is happening.</li> <li>• <b>performer:</b> Any performances or actors present during the event.</li> <li>• <b>organizer:</b> The organisation responsible for the event.</li> </ul>
<a href="#">CreativeWork</a>	<p><b>Definition:</b> A general class for any work that involves creative effort, used as a superclass for movies, TV shows, scripts, etc.</p> <ul style="list-style-type: none"> <li>• <b>name:</b> The name of the creative work.</li> <li>• <b>creator:</b> The person or organisation that created the work.</li> <li>• <b>description:</b> A short description or synopsis of the work.</li> <li>• <b>license:</b> Information on the rights or licensing of the work.</li> <li>• <b>isBasedOn:</b> Other works or sources the current work is based on.</li> </ul>



Schema.org fully supports **JSON-LD** as one of the preferred serialisation formats for structured data. With JSON-LD, entities in Schema.org can be linked to external datasets, enriching the data with additional context. By embedding JSON-LD into web pages or systems, data is accessible in a structured format, enabling seamless data integration across different platforms and ontologies.

In the example below, we integrate multiple entities—*Movie*, *Person*, *Organization*, and *Event*—to showcase how JSON-LD can represent film-related data comprehensively.

```
{
  "@context": "https://schema.org",
  "@type": "Movie",
  "name": "Inception",
  "director": {
    "@type": "Person",
    "name": "Christopher Nolan",
    "birthDate": "1970-07-30",
    "jobTitle": "Director",
    "sameAs": "https://www.imdb.com/name/nm0634240/"
  },
  "actor": [
    {
      "@type": "Person",
      "name": "Leonardo DiCaprio",
      "birthDate": "1974-11-11",
      "jobTitle": "Actor",
      "sameAs": "https://www.imdb.com/name/nm0000138/"
    },
    {
      "@type": "Person",
      "name": "Joseph Gordon-Levitt",
      "birthDate": "1981-02-17",
      "jobTitle": "Actor",
      "sameAs": "https://www.imdb.com/name/nm0330687/"
    }
  ],
  "genre": ["Science Fiction", "Action"],
  "datePublished": "2010-07-16",
  "duration": "PT148M",
  "productionCompany": {
    "@type": "Organization",
    "name": "Warner Bros.",
    "location": "Burbank, California",
    "foundingDate": "1923-04-04",
    "sameAs": "https://www.imdb.com/company/co0002663/",
    "logo": "https://example.com/images/logo.png"
  },
  "trailer": {
    "@type": "VideoObject",
    "name": "Inception Trailer",
    "description": "Official trailer for Inception.",
    "thumbnailUrl": "https://example.com/thumbnail.jpg",
  }
}
```



```
"uploadDate": "2010-06-08",
"contentUrl": "https://example.com/trailer.mp4"
},
"event": {
"@type": "Event",
"name": "Inception Premiere",
"startDate": "2010-07-13",
"location": {
"@type": "Place",
"name": "Grauman's Chinese Theatre",
"address": "6925 Hollywood Blvd, Los Angeles, CA"
},
"performer": {
"@type": "Person",
"name": "Leonardo DiCaprio"
},
"organizer": {
"@type": "Organization",
"name": "Warner Bros."
}
},
"aggregateRating": {
"@type": "AggregateRating",
"ratingValue": "8.8",
"reviewCount": "2000000"
}
}
```

## ii. EBUCorePlus

EBUCorePlus is entirely semantic, designed to eliminate the ambiguities that arose when combining EBUCore and CCDM classes within a single graph. It introduces a new namespace, making it incompatible with previous versions, though mappings to its predecessors are possible. Comprehensive documentation is available for all entities in English, French, and German, with English as the authoritative language.

Figure 27 shows a schema of the main aspects considered in EBUCorePlus, just to give you an initial idea. Unfortunately, the documentation is not extensive at all currently, but it is expected to grow as an open-source project.

There is already an [ontology](#) for EBUCorePlus, but as the documentation is missing, it is unclear if it fully covers the existing specification. A [navigation web page](#) is available to browse the different classes, data properties and object properties.

The ontology viewer used in SCENE can depict the main classes of this ontology (see Figure 28 with an extract of it to read the class names properly).



Figure 27. EBUCorePlus<sup>6</sup> Overview

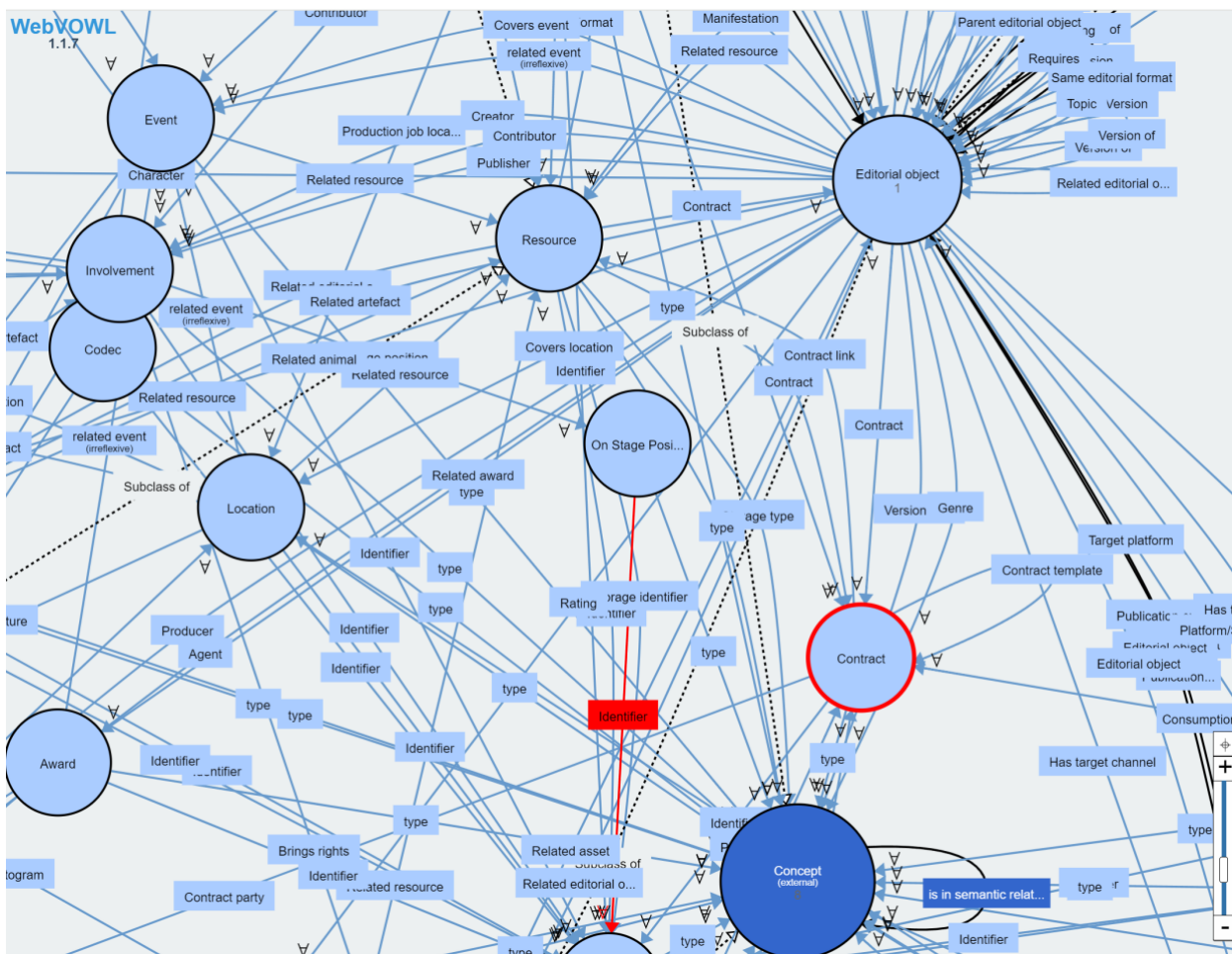


Figure 28. EBUCorePlus displayed in the ontology viewer

Below is a table with potential concepts that directly relate to the film-making industry and could be re-used. Extracting these properties directly from the current OWL file is not straightforward due to the form it is implemented (through OWL restrictions). For simplicity, we have listed some classes from EBUCore.

<sup>6</sup> Source: [EBUCorePlus GitHub](#)

Table 45. Potential re-usable entities and properties from EBUCorePlus

Entity	Definition and Key Properties
<b>MediaResource</b> (subClass of ec:Resource)	<p><b>Definition:</b> Represents audiovisual content, such as movies, TV shows, audio, or multimedia files.</p> <ul style="list-style-type: none"> <li>• <b>title:</b> The title of the media.</li> <li>• <b>identifier:</b> A unique identifier (e.g., an EIDR ID or an internal media ID).</li> <li>• <b>description:</b> A brief summary or synopsis of the content.</li> <li>• <b>creator:</b> The person, company, or organisation responsible for creating the media (linked to Person or Organization).</li> <li>• <b>duration:</b> The total media duration in a standardised time format.</li> <li>• <b>format:</b> Technical details about the media format (e.g., resolution, frame rate, codec).</li> <li>• <b>publicationDate:</b> The date the media resource was published or first broadcast.</li> <li>• <b>hasPart:</b> If applicable, refers to subcomponents of the media (e.g., individual episodes in a TV series).</li> </ul>
<b>Person</b> (subClass of ec:Agent)	<p><b>Definition:</b> Represents individuals involved in the production or creative process of the media, such as actors, directors, writers, or producers</p> <ul style="list-style-type: none"> <li>• <b>firstName</b> and <b>lastName:</b> The given name and family name of the individual.</li> <li>• <b>role:</b> The role or occupation of the person (e.g., actor, director, producer).</li> <li>• <b>birthDate:</b> The birthdate of the individual.</li> <li>• <b>identifier:</b> A unique identifier, potentially linking to an external database (IMDb, LinkedIn, etc.).</li> <li>• <b>contributorTo:</b> Media resources that the person has contributed to.</li> </ul>
<b>Organisation</b> (subClass of ec:Agent)	<p><b>Definition:</b> Entities such as production companies, broadcasters, or film studios</p> <ul style="list-style-type: none"> <li>• <b>name:</b> The organisation's name.</li> <li>• <b>role:</b> The function of the organisation in the media value chain (e.g., production, distribution, or broadcasting).</li> <li>• <b>identifier:</b> A unique identifier potentially linked to a registration number or external entity.</li> <li>• <b>address:</b> The organisation's physical or digital address.</li> </ul>
<b>PublicationEvent</b>	<p><b>Definition:</b> Represents the distribution or broadcast event for the media, such as a film's theatrical release or a TV series episode's first airing.</p> <ul style="list-style-type: none"> <li>• <b>date:</b> The date of the publication or broadcast event.</li> <li>• <b>location:</b> The geographical or virtual location where the media was broadcasted or distributed.</li> <li>• <b>publisher:</b> The organisation responsible for the publication or broadcast.</li> <li>• <b>channel:</b> If applicable, the channel or service (e.g., Netflix, BBC) where the media was distributed or aired.</li> </ul>
<b>TechnicalAttributes</b>	<p><b>Definition:</b> Metadata related to the technical properties of the media resource.</p> <ul style="list-style-type: none"> <li>• <b>format:</b> The file format or encoding (e.g., H.264, MP4).</li> <li>• <b>bitRate:</b> The bit rate of the media, especially relevant for streaming.</li> <li>• <b>resolution:</b> The video resolution (e.g., 1920x1080).</li> <li>• <b>frameRate:</b> The number of frames per second.</li> <li>• <b>fileSize:</b> The total size of the file</li> </ul>

EBUCore is designed to work seamlessly with **Linked Data** and supports **JSON-LD** as one of its serialisation formats. By using JSON-LD, EBUCore enables the structured representation of media metadata that can be linked to external datasets and ontologies. EBUCore's use of JSON-LD also facilitates its integration with other



linked data models, enabling interoperability with other ontologies such as Schema.org, and improving the discoverability of media content

In the example below, we integrate multiple entities - *MediaResource*, *Person*, *Organization*, and *PublicationEvent* -, representing a film and its related metadata.

```
{
  "@context": "https://www.ebu.ch/metadata/ontologies/ebucoreplus/",
  "@type": "MediaResource",
  "title": "Inception",
  "identifier": "urn:movie:tt1375666",
  "description": "A skilled thief, who enters the dreams of others to steal secrets, is given a chance to have his criminal history erased if he can implant an idea in a target's subconscious.",
  "creator": {
    "@type": "Person",
    "firstName": "Christopher",
    "lastName": "Nolan",
    "role": "Director",
    "birthDate": "1970-07-30"
  },
  "contributor": [
    {
      "@type": "Person",
      "firstName": "Leonardo",
      "lastName": "DiCaprio",
      "role": "Actor",
      "birthDate": "1974-11-11"
    },
    {
      "@type": "Person",
      "firstName": "Joseph",
      "lastName": "Gordon-Levitt",
      "role": "Actor",
      "birthDate": "1981-02-17"
    }
  ],
  "publisher": {
    "@type": "Organization",
    "name": "Warner Bros.",
    "identifier": "urn:warnerbros:studio",
    "role": "Production Company"
  },
  "publicationEvent": {
    "@type": "PublicationEvent",
    "date": "2010-07-16",
    "location": {
      "@type": "Place",
      "name": "United States"
    },
    "publisher": {
      "@type": "Organization",
      "name": "Warner Bros."
    }
  }
}
```



```

"channel": "Theatrical Release"
},
"technicalAttributes": {
"@type": "TechnicalAttributes",
"format": "H.264",
"bitRate": "5000kbps",
"resolution": "1920x1080",
"frameRate": "24fps",
"fileSize": "15GB",
"aspectRatio": "16:9"
},
"duration": "PT148M"
}

```

EBUCore was successfully integrated into two significant European initiatives, **Europeana** and **EUScreen**, both aimed at preserving and promoting access to European cultural heritage.

- [Europeana](#) is an EU-funded initiative that is a digital gateway to millions of cultural heritage items from European museums, libraries, archives, and galleries. EBUCore plays a critical role in enhancing the descriptive metadata for audiovisual materials within the platform, ensuring that media content such as films, broadcasts, and audio recordings can be catalogued, searched, and retrieved with high precision. By leveraging EBUCore’s detailed technical and descriptive metadata framework, Europeana enables better integration and interoperability for audiovisual content, ensuring that cultural heritage is represented accurately across various formats and mediums. The Europeana Data Model (EDM) reuses many properties from EBUCore RDF.
- [EUScreen](#) was a project (ended in 2016) that provided online access to European television and audiovisual archives, aimed at enhancing public access to historical and contemporary broadcast material. EBUCore was used within EUScreen to standardise the metadata for these audiovisual archives, ensuring that content from different countries, languages, and formats could be consistently described and made searchable. The integration of EBUCore enabled EUScreen to offer detailed metadata for media resources, making it easier for users to discover and explore European broadcast history in an organised and user-friendly manner.

### iii. Dublin Core

Dublin Core provides a set of 15 flexible core elements that can be reused to describe film-related entities. Below is a table with potential concepts from Dublin Core that might directly relate to the film-making industry and could be re-used.

*Table 46. Potential re-usable terms from Dublin Core*

Term	Definition
<b>title</b>	The name or title of the resource (e.g., the title of a movie)
<b>creator</b>	The person or organisation responsible for the creation of the content (e.g., the director of a movie)
<b>subject</b>	The topic or genre of the resource (e.g., "Science Fiction", "Action", "Drama")



<b>description</b>	A summary or description of the content (synopsis)
<b>publisher</b>	The entity responsible for making the resource available, typically the production company or studio
<b>contributor</b>	Any individual or entity contributing to the resource (e.g., actors, producers, or other contributors)
<b>date</b>	A point in time related to the creation or publication of the resource (e.g., release date, broadcast date, or any other relevant time-related information)
<b>type</b>	The nature or genre of the resource, often corresponding to film, TV show, or other media types (e.g., "Film", "TVSeries", "Documentary")
<b>format</b>	The file format, physical medium, or dimensions of the resource (e.g., "MP4", "1080p", "H.264")
<b>identifier</b>	A unique reference to the resource, such as an ISBN, DOI, or EIDR for films (e.g., "urn:movie:tt1375666" for IMDb)
<b>source</b>	The resource from which the current resource is derived or related
<b>language</b>	The language of the resource content (e.g., "English", "Spanish")
<b>relation</b>	A reference to a related resource, such as sequels, prequels, adaptations, or spin-offs
<b>coverage</b>	The spatial or temporal scope of the resource (e.g., "United States", "20th Century")
<b>rights</b>	Information about rights held in and over the resource

Dublin Core supports **JSON-LD** for structuring metadata in a machine-readable, linked-data format. Dublin Core elements can be serialised in JSON-LD using their namespaces to connect to external resources like creators, contributors, or related media entities, promoting interoperability and easy data integration across different domains.

Here's an example of how to represent a film (Inception) using Dublin Core elements in JSON-LD:

```
{
  "@context": "http://purl.org/dc/elements/1.1/",
  "@type": "Movie",
  "title": "Inception",
  "creator": {
    "@type": "Person",
    "name": "Christopher Nolan"
  },
  "subject": "Science Fiction",
  "description": "A skilled thief is given a chance to erase his past by planting an idea into the mind of a corporate target.",
  "publisher": {
    "@type": "Organization",
    "name": "Warner Bros."
  },
  "contributor": [
    {
      "@type": "Person",
      "name": "Leonardo DiCaprio"
    }
  ]
}
```



```

{
  "@type": "Person",
  "name": "Joseph Gordon-Levitt"
}
],
"date": "2010-07-16",
"type": "Film",
"format": "H.264",
"identifier": "urn:movie:tt1375666",
"language": "English",
"relation": {
  "@type": "Movie",
  "name": "The Dark Knight"
},
"coverage": "United States",
"rights": "All rights reserved by Warner Bros."
}

```

#### iv. Ontology for Media Creation (OMC) from MovieLabs

The **OMC** ontology includes entities representing various stages and components of media creation, which help design complex workflows. Main building blocks are represented in Figure 29.

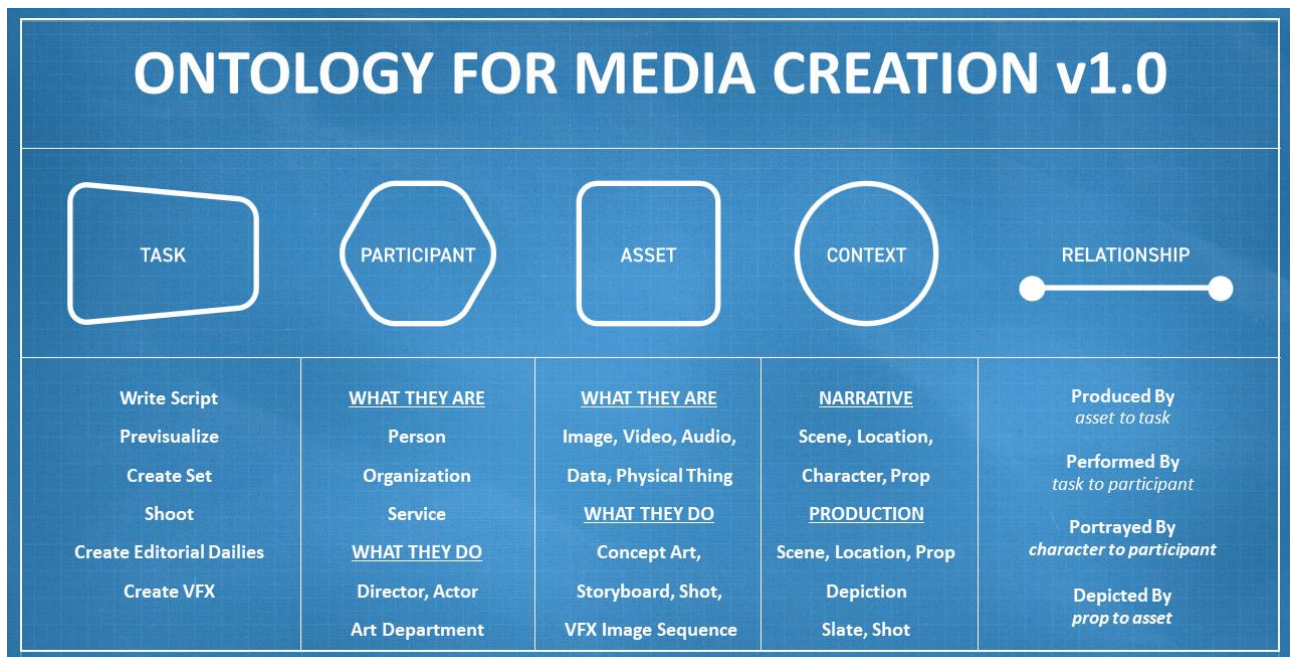


Figure 29. OMC building blocks<sup>7</sup>

There is [online documentation](#) where one can [browse](#) through the classes, data properties and object properties, among others in a textual way. For a more graphical interface, we used the Ontology viewer (see Figure 30) There is a [JSON schema](#) for the ontology as well as a [TTL file](#) with the OMC ontology itself.

<sup>7</sup> Source: [MovieLabs](#)

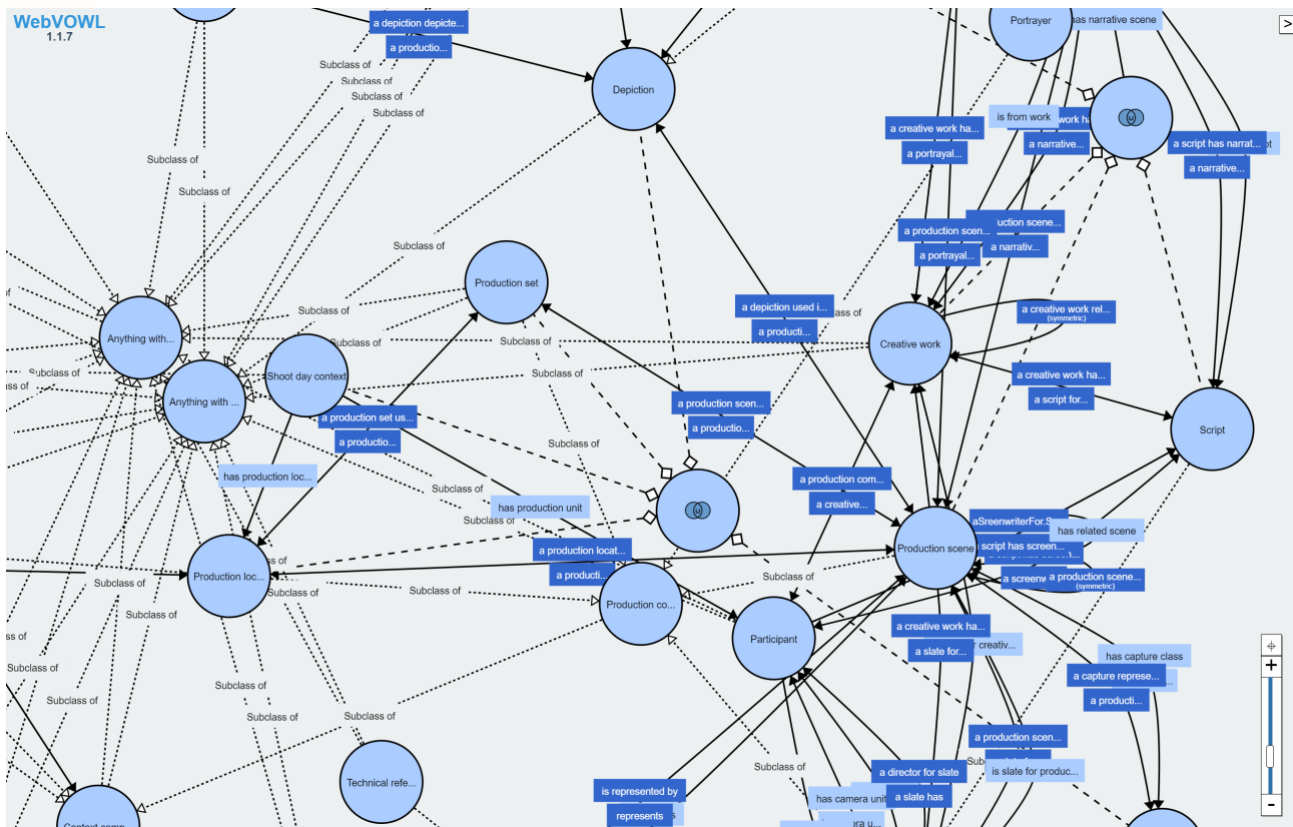


Figure 30. OMC displayed in the ontology viewer

Below is a table with potential concepts from MovieLabs’ OMC that directly relate to the film-making industry and could be re-used.

Table 47. Potential re-usable entities and properties from OMC

Entity	Definition and Key Properties
Asset	<p><b>Definition:</b> Represents a physical or digital object specific to the creation of the creative work (e.g., video files, images)</p> <ul style="list-style-type: none"> <li>• <b>identifier:</b> Unique identifier for the asset</li> <li>• <b>name:</b> Human-readable name for the asset</li> <li>• <b>description:</b> A human-readable description of the asset</li> <li>• <b>version:</b> Information about the version of the asset</li> <li>• <b>provenance:</b> Tracks the origin or history of the asset</li> </ul>
Character	<p><b>Definition:</b> Represents a sentient entity in the script whose specific identity is important to the narrative (e.g., actors)</p> <ul style="list-style-type: none"> <li>• <b>identifier:</b> Unique identifier for the character</li> <li>• <b>name:</b> Primary name used for the character</li> <li>• <b>description:</b> A brief description of the character</li> <li>• <b>profile:</b> Physical and background characteristics of the character</li> </ul>
MediaCreationContext	<p><b>Definition:</b> Represents a context within which a media project is created</p> <ul style="list-style-type: none"> <li>• <b>identifier:</b> Unique identifier for the context</li> <li>• <b>name:</b> Human-readable name for the context</li> <li>• <b>description:</b> A brief description of the context</li> <li>• <b>Context:</b> Nested contexts to define detailed scopes</li> </ul>
CreativeWork	<p><b>Definition:</b> Represents a unique production or creative work (e.g., a film or TV series)</p> <ul style="list-style-type: none"> <li>• <b>identifier:</b> Unique identifier for the creative work.</li> </ul>



- **title:** Includes the official and working title of the work.
- **description:** A brief description of the creative work.
- **originalLanguage:** The language(s) of the creative work.

Although the OMC schema is structured in **JSON**, the data can easily be adapted into **JSON-LD** due to its hierarchical structure and use of identifiers. The OMC JSON schema allows linking entities (such as assets, characters, and creative works) using references, making it suitable for **Linked Data** environments.

Here is an example based on the OMC schema to represent an **Asset** and **CreativeWork**:

```
{
  "@context": "https://movielabs.com/omc/json/schema/v2.0",
  "@type": "CreativeWork",
  "identifier": "urn:creativework:inception123",
  "title": {
    "officialTitle": "Inception",
    "workingTitle": "Dream Project"
  },
  "description": "A skilled thief is given a chance to erase his past by planting an idea into the mind of a corporate target.",
  "originalLanguage": ["English"],
  "hasAsset": {
    "@type": "Asset",
    "identifier": "urn:asset:scene1",
    "name": "Inception Hallway Fight",
    "description": "The famous zero-gravity fight scene",
    "version": {
      "versionNumber": "1.0",
      "description": "Original Cut"
    }
  },
  "provenance": "Created during principal photography."
}
```

## v. VideoMetadataHub from IPTC

As extracted from its summary brochure<sup>8</sup>, The **IPTC (International Press Telecommunications Council) Video Metadata Hub** is a universal schema designed to manage and exchange video metadata consistently and reliably, providing a common ground across different systems and standards. It combines metadata properties from various formats, such as QuickTime, XMP, MPEG4, MPEG7, EBUCore, and PBCore, as well as manufacturer-specific formats like Sony XDCAM and Panasonic SMPTE P2. This standard enables users to describe the content, rights, administrative details, and technical characteristics of videos in a system-independent manner, addressing challenges in transferring metadata between systems.

The summary brochure also states that the hub provides **23 properties** (currently 26) for content description, **14 for rights** (currently 15), **22 for administrative information** (currently 30), and **26 for technical details**. Each property is defined with clear semantics, a basic data type, and cardinality. These properties can be used for entire videos or specific clips and expressed in standards like **EBUCore**, **XMP**, and **JSON**. The hub

<sup>8</sup> [Video Metadata Hub one-pager PDF](#)



also supports mappings to formats such as **Apple QuickTime**, **MPEG-7**, and **Schema.org**, promoting easier interoperability across different platforms and workflows

There is [online documentation](#) about the latest recommendation on 4 October 2023. The specifications for Video Metadata Hub are separated into two parts: [Properties](#) and [Mappings](#).

There is also a [user guide](#) on how to use and implement IPTC Video MetadataHub. Some property examples have been extracted from the previous link and are listed below in a summary table.

Property name (and type)	Definition and additional information
<b>Date Created (administrative)</b>	<p><b>Definition:</b> Designates the date and, optionally, the time the content of the video was created (rather than the date of the creation of the digital representation)</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “photoshop:DateCreated (Date)”</li> <li>• <b>JSON ID and data type:</b> “dateCreated (string/date-time/)”</li> </ul>
<b>Episode (administrative)</b>	<p><b>Definition:</b> Episode in a specific season of a TV or video series this video is a member of.</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “Iptc4xmpExt:Episode (Episode structure)”</li> <li>• <b>JSON ID and data type:</b> “episode (EpisodeSeason)”</li> </ul>
<b>External Metadata URL (administrative)</b>	<p><b>Definition:</b> Link(s) to an external web resource for retrieval of further metadata about this video</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “Iptc4xmpExt:ExternalMetadataLink (Bag of URL)”</li> <li>• <b>JSON ID and data type:</b> “externalMetadataLinks (string/uri/array)”</li> </ul>
<b>Metadata Authority (administrative)</b>	<p><b>Definition:</b> Party responsible for the accuracy of the metadata values.</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “Iptc4xmpExt:metadataAuthority (Entity structure)”</li> <li>• <b>JSON ID and data type:</b> “metadataAuthority (Entity//)”</li> </ul>
<b>Rating (administrative)</b>	<p><b>Definition:</b> How the video is classified by a public authority such as MPA</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “Iptc4xmpExt:Rating (Bag Rating structure)”</li> <li>• <b>JSON ID and data type:</b> “ratings (Rating//array)”</li> </ul>
<b>CV Term About the Content (content description)</b>	<p><b>Definition:</b> What the video is about expressed by term(s) selected from taxonomies or controlled vocabularies</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “Iptc4xmpExt:AboutCvTerm (Bag CV Term structure)”</li> <li>• <b>JSON ID and data type:</b> “aboutCvTerms (CvTerm//array)”</li> </ul>
<b>Description (content description)</b>	<p><b>Definition:</b> Textual description of the content of the video</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> dc:description (Lang Alt)</li> <li>• <b>JSON ID and data type:</b> description (AltLang)</li> </ul>
<b>Extended Description (Accessibility) (content description)</b>	<p><b>Definition:</b> A more detailed textual description of the purpose and meaning of a video that elaborates on the information provided by the Alt Text (Accessibility) property. This property has no character limitation and is not required if the Alt Text (Accessibility) field sufficiently describes the video.</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “Iptc4xmpCore:ExtDescrAccessibility (Lang Alt)”</li> <li>• <b>JSON ID and data type:</b> “extDescrAccessibility (AltLang)”</li> </ul>
<b>Genre (content description)</b>	<p><b>Definition:</b> Artistic, style, journalistic, product or other genre(s) of the video.</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “Iptc4xmpExt:Genre (Bag CV Term structure)”</li> <li>• <b>JSON ID and data type:</b> “genres (CvTerm//array)”</li> </ul>
<b>Keywords (content description)</b>	<p><b>Definition:</b> What the video is about expressed by a free choice of descriptive phrases or keywords</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “dc:subject (Bag Lang Alt)”</li> </ul>



	<ul style="list-style-type: none"> <li>• <b>JSON ID and data type:</b> “keywords (AltLang//array)”</li> </ul>
<b>Title</b> (content description)	<p><b>Definition:</b> Title of the video, should be a shorthand reference</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “dc:title (Lang Alt)”</li> <li>• <b>JSON ID and data type:</b> “title (AltLang)”</li> </ul>
<b>Contributor</b> (rights)	<p><b>Definition:</b> Party or parties (person or organisation) which contributed to the video, refinement by the role attribute.</p> <p><b>Notes:</b> A distinction between contributor and creator should follow rights laws, contracts or common business rules. Vocabulary of roles of persons contributing to a video should be based on the use by a video or movie producer association.</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “lptc4xmpExt:Contributor (Bag Entity with Role structure)”</li> <li>• <b>JSON ID and data type:</b> “contributors (EntityWRole//array)”</li> </ul>
<b>Creator</b> (rights)	<p><b>Definition:</b> Party or parties (person or organisation) which created the video, refinement by the role attribute.</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “lptc4xmpExt:Contributor (Bag Entity with Role structure)”</li> <li>• <b>JSON ID and data type:</b> “contributors (EntityWRole//array)”</li> </ul>
<b>Copyright Notice</b> (rights)	<p><b>Definition:</b> Any textual notice necessary by legal needs or common use to indicate the current owner of the copyright of this media resource.</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “dc:rights (Lang Alt)”</li> <li>• <b>JSON ID and data type:</b> “copyrightNotice (AltLang)”</li> </ul>
<b>Data Mining</b> (rights)	<p><b>Definition:</b> Data mining prohibition or permission, optionally with constraints.</p> <p><b>Notes:</b> This is a <a href="#">structured PLUS version 2.0 property</a>. The value must be one of the URIs listed in the Controlled Vocabulary section of the PLUS specification for the Data Mining property.</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “plus:DataMining (see PLUS spec)”</li> <li>• <b>JSON ID and data type:</b> “dataMining (string/uri)”</li> </ul>
<b>Audio Bitrate</b> (technical)	<p><b>Definition:</b> Bit rate of the audio data depending on the Video Bit Rate Type: if fixed, the fixed rate; if variable, the maximum rate. The unit is bits per second.</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “lptc4xmpExt:audioBitRate (Integer)”</li> <li>• <b>JSON ID and data type:</b> “audioBitRate (number//)”</li> </ul>
<b>DisplayAspectRatio</b> (technical)	<p><b>Definition:</b> Ratio of width and height of the displayed image. (Width and height do not require to be in pixels).</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “lptc4xmpExt:videoDisplayAspectRatio (Rational)”</li> <li>• <b>JSON ID and data type:</b> “videoDisplayAspectRatio (string//)”</li> </ul>
<b>File Duration</b> (technical)	<p><b>Definition:</b> Duration of the overall video (from the first to the last frame).</p> <ul style="list-style-type: none"> <li>• <b>XMP ID and data type:</b> “xmpDM:duration (Time)”</li> <li>• <b>JSON ID and data type:</b> “fileDuration (VideoTime)”</li> </ul>

There are also online [examples of JSON files](#) available. Additionally, the IPTC’s interactive [IPTC Video Metadata Hub Generator tool](#) can be used to create Video Metadata Hub metadata in various formats for testing and prototyping purposes. The available [GitHub repository](#) might also be useful for checking other examples and for developers.

While **JSON** is supported, the official resources do not explicitly mention JSON-LD. However, given the structured format of the properties, it is possible to adapt them into a **JSON-LD** representation by adding contexts and types in the linked data format.



Here's a possible JSON-LD representation based on the JSON structure provided by IPTC, combined with schema.org:

```
{
  "@context": {
    "iptc": "https://iptc.org/std/videometadatahub/",
    "schema": "https://schema.org/"
  },
  "@type": "schema:VideoObject",
  "schema:identifier": "urn:video:example123",
  "schema:name": "Example Video",
  "schema:description": "A short example video.",
  "schema:duration": "PT2M30S",
  "schema:contributor": {
    "@type": "schema:Person",
    "schema:name": "John Doe",
    "schema:role": "Director"
  },
  "schema:publication": {
    "@type": "schema:BroadcastEvent",
    "schema:startDate": "2024-07-16",
    "schema:location": "Online Streaming"
  },
  "schema:encodingFormat": "MP4",
  "schema:contentUrl": "https://example.com/video.mp4",
  "schema:license": "Creative Commons",
  "schema:copyrightHolder": "Example Rights Holder",
  "iptc:technicalData": {
    "iptc:format": "MP4",
    "iptc:bitRate": "4000kbps",
    "iptc:frameRate": "30fps",
    "iptc:resolution": "1920x1080"
  },
  "iptc:rights": {
    "iptc:license": "Creative Commons",
    "iptc:rightsHolder": "Example Rights Holder"
  }
}
```

## E. Requirements Questionnaire (first iteration)

### Introduction

Collaboration is vital as we delve into SCENE's T3.1 task, focusing on film-related data lakes and creating the SCENE-O ontology. Each partner, including you, brings a unique tool to the table. To ensure synergy and a unified approach, we'd love to understand more about the data specifics of your tool. Please take a moment to share these details below.

**Important:** *We realise that it might be difficult to answer many of the questions since the tools are not yet mature and are still in the design phase. Therefore, if you are unsure what to answer do not provide an answer. We will come back again at a later stage of the project.*



**Part A: General Information**

1) Which tool are you developing and for which task?

Please Specify:

2) At which stage of the film production process is your tool used (more than one, if applicable)?

- Pre-production
- Production
- Post-production
- Distribution
- All the phases

**Part B: Data Characteristics**

3) What type of data does your tool consume? Please specify common formats for each data type you use

- Images
- Videos
- Audio
- Metadata
- Text
- 3D models
- Other  Please specify .....

4) Is the data you consume primarily owned by you, or are you using external data sources? Or maybe a combination of both?

5) How do you search for the data that you consume? Are you using any search tools? Please specify

6) What type of data does your tool produce?

- Images
- Videos
- Audio
- Metadata



- Text
- 3D models
- Other Please specify .....

- 7) If your tool produces data, please specify the common type formats, codes (if applicable), and resolutions (if applicable).
- 8) What software is often used to view or modify your data?

**Part C: Data Storage**

- 9) How do you currently store the data your tool produces? (e.g., flat files, databases, in-memory, cloud storage)
- 10) If using a database:
  - a. What Type of database is it? (e.g., SQL, NoSQL, Time Series)
  - b. Can you provide a sample schema or an overview of the data structure?

**Part D: Data Volume**

- 11) If you use owned data as input for your tool, can you estimate its size (GB/TB)? Even if this might change over time, can you estimate an average size representing 80% of the common input data types typically needed by your tool?
- 12) Can you estimate how much data your tool produces (specify GBs/day or GBs/week)?
- 13) How frequently is new data produced? (e.g., real-time, hourly, daily, on request)

**Part E: Data Access**

- 14) Who are the primary users or systems that need access to this data?
- 15) What authentication and authorisation mechanisms are in place or required?

**Part F: Data Security, Privacy & Sharing**

- 16) Does your tool handle any sensitive or personal data elements? If so, please describe.
- 17) What Encryption methods are used for data at rest and in transit?
- 18) Are there any data anonymisation or pseudonymisation requirements?
- 19) Do you anticipate sharing this data with other systems or partners?
- 20) Do you foresee that there will be any data licensing or copyright considerations?



Note: for the last two questions, do you foresee that your input and/or output data could be stored and managed in an independent repository (e.g., within the SCENE platform), or is there any technical or legal constraint?

### Part G: Ontology & Metadata Standards

Note: If you are unaware of ontologies, it is a procedure to model knowledge. Most of the time, you might identify them as a set of vocabularies that appear as metadata or taxonomies (classifications). More formally, ontologies appear as OWL files and have their own semantic API (SPARQL).

- 21) Does your tool use semantics (ontologies) to describe or annotate the data?
  - a. If yes, please specify the semantics/ontology(ies) and provide any relevant documentation or links.
- 22) How is the ontology integrated into the data? (e.g., embedded, separate file, linked data)
- 23) Does your tool produce metadata alongside the primary data? If so, what metadata standards or schemas do you follow?
- 24) Can you provide a sample of the metadata produced or a template?
- 25) How crucial is this metadata for understanding or processing the primary data?
- 26) Are semantic relationships or hierarchies within or between the data and other datasets?
- 27) How are these relationships encoded or represented?
- 28) Do you experience or see any limitation in the current treatment of your data that could be better managed by using (or enriching with) semantics?

### Part H: Communication with other tools

- 1) Does your tool require any input to be consumed from another task?
- 2) Does your tool provide outputs consumed by other tasks?
- 3) How would your tool provide their output? Via APIs or through the storage in a database (e.g., the Data Lake)?

Thank you for providing this information. It's crucial for ensuring seamless integration and optimal data lake use. If there are any further details or updates, please let us know.

## F. Requirements Questionnaire (second iteration)

For the following tables, the following score legend applies:

<b>R</b>	greatly relevant
<b>r</b>	less relevant
<b>n</b>	not relevant at all



**i. Data Lakes**

*Table 48. Requirements summary for the data lake (feedback from 4 partners)*

FEATURE (see D3.1, section 2.1.4.1)	CERTH	MOG	AUTH	DTT
Content storage (store massive amounts of content)	R	R (Media Asset Manager needs to store the generated video assets in the data lake)	r (For the Audio Simulation module, some audio files and models need to be stored in order to provide the acoustic simulation of a room.)	r
Data ingestion (ingestion from data from different sources)	n	R (Media Asset Manager should be able to ingest video content to the data lake)	n	R
Metadata Management (store metadata from primary media assets)	r	R (Videos ingested have associated metadata. The media asset manager uses the concept of collections (list of videos) and products (list of collections/videos) to organize content. Videos can be licensed, this information will be stored in a blockchain, but might also make sense to store it in the data lake (even if only as a reference to the blockchain))	n	r
Data Analysis (e.g. audience preferences, market trends, etc)	n	n	n	n
Collaboration (common repo for various teams/individuals - directors, producers, etc.)	n	n	n	n

sharing/exploiting similar info				
Cost management	n	n	n	n
Archiving & preservation	n	r (Archiving can be used to store the original asset, before transcoding)	n	R
Personalization	n	n	r (For the Audio Simulation module, based on the task description, users are expected to provide their own audio files in order to listen to how they would sound in the target space.)	n
Security & compliance	R	n	n	r
Streaming service	n	R (Distribution engine requires a public facing url for video playout)	n	n
Other functionality ...<please define>	n	n	n	n

Table 49. User stories summary for the data lake (feedback from 4 partners)

	As a <name the user profile/consumer of your tool>	I want to <define what you need from the DL>	so that ... <define why you do that, what is the benefit in order to bring value to the DL>
Example	Location scouter	have access to several location lists from a single-entry point	I can find easier/faster the proper location without looking in different places/services
CERTH	external tool	Filter the available contents in the data lake	the tool will have a manageable amount of data to achieve its primary functionalities
MOG	video producer	Store video files and associated metadata	The videos can be available for distribution
		Group videos by collections	I can easily find related videos
		Retrieve other videos from the data lake	I can prepare them for distribution
		Retrieve information about a video	I can filter videos based on metadata

		Store a license template	I can license my video
	video consumer	Retrieve license templates	I can check what licenses are available for purchase
		Store a license template	I can register a video license purchase
		Retrieve videos from the data Lake	I can play them
AUTH	location builder	Be able to upload a new (acoustic) model with their own needed (audio) files and associate it with my built location <i>Note: Files that are stored in the data lake (small audio files) needed for audio modelling of a location should include a tag regarding to which location they are connected</i>	The edited location (or newly created one) is tagged for supporting an (acoustic) model
DTT	Cinematographer, Location scouter, Producer	be able to store 3D models and associated metadata	the 3D models can be accessible to the film personnel to decide on the film location, experiment with different effects (e.g., lighting and audio effect) and such

ii. SCENE-O

Table 50. Requirements summary for the SCENE ontology (feedback from 4 partners)

FEATURE	CERTH	MOG	AUTH	DTT
Conversion/mapping alignment (from one metadata standard into another)	n	n	n	r



<p><b>Annotation</b> (generate manual/automatic metadata from raw data)</p>	<p>n</p>	<p><b>R</b> (Media Asset Manager generates metadata associated to the video essence that are required by the Distribution Engine for video search and playout. It does not follow a standard, but videometadatahub by IPTC might be worth to use)</p>	<p><b>r</b> (The Audio Simulation module will store some files that are needed for the modeling of a location. These files should be annotated and linked to the location. Every location should include a tag on whether it contains an audio simulation model. Every location that includes an audio simulation model might have a tag describing the kind of simulation technique that is used (not defined yet)</p>
<p><b>Domain understanding</b> (metadata vocabulary, taxonomies)</p>	<p>n</p>	<p>n</p>	<p>n</p>
<p><b>Improved search</b> (define/restrict the context of user queries)</p>	<p>n</p>	<p>n</p>	<p>n</p>
<p><b>Information extraction</b> (extract relevant info from unstructured content like texts)</p>	<p>n</p>	<p>n</p>	<p>n</p>
<p><b>Reasoning and inference</b> (logical reasoning, deduction of implicit knowledge from explicit facts, inference to fill missing gaps, etc.)</p>	<p>n</p>	<p>n</p>	<p>n</p>
<p><b>Machine learning and AI</b> (structure training data, context for AI systems, etc.)</p>	<p>n</p>	<p>n</p>	<p>n</p>
<p><b>Other functionality ...&lt;please define&gt;</b></p>	<p>n</p>	<p>n</p>	<p>n</p>

Table 51. User stories summary for the SCENE-O (feedback from 4 partners)

	As a <name the user profile/consumer of your tool>	I want to <define what you need from the DL>	so that ... <define why you do that, what is the benefit in order to bring value to the DL>
Example	Location scouter	refine my current search with additional metadata	I can restrict the target potential locations that satisfy all my requirements
CERTH			
MOG			
AUTH	location scouter	Search for a location that has audio modelling information	I can have a simulation of how the acoustic space sounds by providing my own audio recordings
DTT	Cinematographer, Location scouter, Producer	be able to store 3D models and associated metadata	the 3D models can be accessible to the film personnel to decide on the film location, experiment with different effects (e.g., lighting and audio effect) and such

### iii. Interactions between tools

Table 52. Mapping between tools and DL/semantics (feedback from 4 partners)

	DATALAKE My <T3.X, T4.X tool> provides the following input	DATALAKE My <T3.X, T4.X tool> will request the following to the DL	ONTOLOGY My <T3.X, T4.X tool> requires the following semantic functionality when inserting data in the DL	ONTOLOGY My <T3.X, T4.X tool> requires the following semantic functionality when extracting data from the DL
Example	audio, video, images, metadata, text, etc.	audio, video, images, metadata, text, etc.	semantic annotation/tagging, semantic conversion, etc.	semantic search, filtering by category/keywords, etc.
CERTH	metadata enrichment (SCENE-O)	the metadata (SCENE-O) of the content	The metadata (SCENE-O) of the newly added/modified/deleted contents	Search functionalities. The users will be other tools in the SCENE framework (like Media Asset Manager)
MOG	1) video files of the same asset with different resolutions 2) metadata associated with the video (asset	videos produced by other tools		



	<p>metadata, film metadata such as date, producer, content rating, etc.)</p> <p>3) license templates and licenses associated with videos</p>			
<b>AUTH</b>	<p>For every acoustically modelled room, some very small files will be stored that act as a descriptor of this room</p>	<p>The user will provide some audio files to the tool to listen to how they would sound like if they were recorded in the simulated space. The tool will have to reach the files that are used for the modelling of this room as input, along with the user input to provide a simulated audio file. The output file will not be stored, it will only be returned to the user.</p>		<p>The locations that include acoustic modelling should have this information. So when a location scouter checks a location, they should be able to know if acoustic modelling is available in that location. Moreover, the users should be able to search for locations that include acoustic modelling</p>
<b>DTT</b>	<p>3D model and metadata</p>	<p>input data of a cultural site in the form of Images/point cloud</p>		