ETSI GR CIM 052 V1.1.1 (2025-01)

Context Information Management (CIM); VR and AR for Smart Learning: Guidelines for using NGSI-LD to train personnel in Smart Industries

Reference

DGR/CIM-0052

Keywords

API, augmented reality, IoT, NGSI-LD, smart industry

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group (ISG) cross-cutting Context Information Management (CIM).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document focuses on the selection of relevant and representative implementations, methodologies, and standards employed in the training of personnel within Smart Industries. It establishes guidelines for the effective use of the NGSI-LD API in Smart Learning applications enhanced by VR/AR technologies. Additionally, it includes the mapping of at least one Smart Learning standard to NGSI-LD to ensure compatibility and interoperability across various platforms.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] Milgram, Paul, H. Takemura, A. Utsumi, F. Kishino (1994), "Augmented Reality: A class of displays on the reality-virtuality continuum", Proceedings of SPIE - The International Society for Optical Engineering Vol. 2351. Retrieved 2021-06-01.

[i.2] Meta, Immersive Experiences: "Best Practices".

[i.3] Billy T.M. Wong, Kam Cheong Li: "Research and Practice in Smart Learning: A Literature Review", in IEEE™ Access, pp. 23-26, 2020.

[i.4] ISO/IEC TR 23844:2023: "Information technology for learning, education, and training — Immersive content and technology".

[i.5] ISO/IEC 5927:2024: "Computer graphics, image processing and environmental data representation — Augmented and virtual reality safety — Guidance on safe immersion, set up and usage".

[i.6] Experience API (xAPI) Standard.

[i.7] ADL: "Documentation about xAPI data model and features".

[i.8] ADL: "Sharable Content Object Reference Model SCORM®".

[i.9] IETF RFC 2046 (November 1996): "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types".

[i.10] ETSI GS ARF 004-2 (V1.1.1) (08/2021): "Augmented Reality Framework (ARF) Interoperability Requirements for AR components, systems and services Part 2: World Storage and AR Authoring functions".

[i.11] "Minerva project", HT s.r.l, P.M.F. s.r.l, Sfera s.r.l, European Project F/190045/01-02-03/X44.

[i.12] Khronos GROUP: "OpenXR".

[i.13] W3C® Immersive Web Working and Community Groups: "WebXR".

[i.14] OGC 06-103r4 V1.2.1: "OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture".

[i.15] IETF RFC 5646: "Tags for Identifying Languages".

[i.16] ISO 8601 (2019): "Data elements and interchange formats — Information interchange — Representation of dates and times.

[i.17] IETF RFC 7946: "The GeoJSON Format".

3 Definition of terms, symbols and abbreviations

3.1 Terms

Void

3.2 Symbols

Void

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

3D	three-Dimensional
ADL	Advanced Distributed Learning
API	Application Programming Interface
APP	Achievements, Progress and Position tracking software for LET
AR	Augmented Reality
ARF	Augmented Reality Framework
ETSI	European Telecommunications Standards Institute
GPS	Global Positioning System
GS	Group Specification
HMD	Head Mounted Display
IEC	International Electrotechnical Commission
IEEE	Institute for Electrical and Electronic Engineers
IETF	International Engineering Task Force
IRI	Internationalized Resource Identifier
IRL	Internationalized Resource Locator
ISG	Industry Specification Group
ISO	Internation Standardization Organisation
JSON	JavaScript Object Notation
LBS	Location-Based Service
LET	Learning, Education and Training
LMS	Learning Management System
MR	Mixed Reality
OpenXR	Open standard for virtual and augmented reality
RFC	Request For Comments
s.r.l.	società a responsabilità limitata
SCORM®	Sharable Content Object Reference Model
SRS	Spatial Reference System
TR	Technical Report
UI/UX	User Interface/User eXperience
VR	Virtual Reality
W3C	World Wide Web Consortium
XAPI	eXperience API
XR	eXtended Reality

4 Immersive technologies used for Smart Learning based on 3D representation of real environments

4.1 Types of immersive technologies and their use

4.1.0 Foreword

Based on the available literature [i.1], the realm of immersive technologies is characterized by the 'distance' between the 3D experiences developed and the reality on which they are based. The main category of immersive content is the eXtended Reality (XR), it is the full range of virtual content added to the real environment. This spectrum includes various technologies ranging from the simplest augmentation of reality of Augmented Reality (AR), to more complex and virtual experiences of Mixed Reality (MR) and Virtual Reality (VR) technologies. The usage of immersive technologies is growing, these enhance the quality and the efficiency of traditional Learning, Education and Training (LET) environments and make experiences more inclusive than ever.

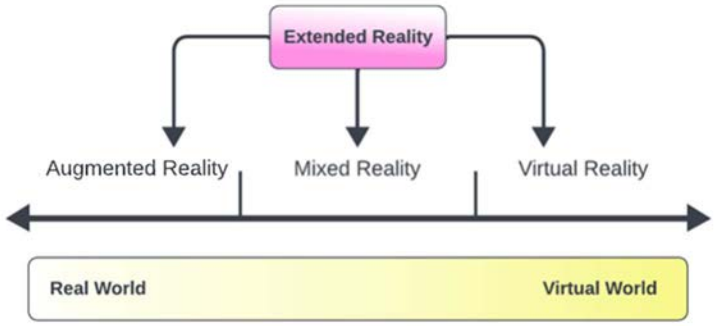


Figure 4.1.0-1: The reality-virtuality continuum diagram

4.1.1 AR-based content

AR-based content [i.4] consists of overlaying the real world with virtual 3D objects. This type of content requires mobile devices or compatible see-through devices. These immersive experiences are the least computationally demanding among others and are consumed daily by millions of people around the world. In fact, AR face filters, popular mobile games, and many interior design customization applications use AR technology to create engaging and personalized experiences. AR-based content includes:

- **Location-Based Service (LBS) applications**, heavily use Global Positioning System (GPS), gyroscope sensor and can rely on images to gather information about user's position in the real world.
- **Image and object recognition applications**, make use of the device's camera module to recognize marker, images and objects and make possible to instantiate 3D objects on the plane of the scanned asset (museum or school applications and face filter used in social networks as reference).
- **Projection applications**, are limited to small projectors installed on a device that project images directly to the user's retinas or eyeglasses to display additional real-time information about the environment.

4.1.2 VR-based content

Virtual Reality-based (VR-based) content [i.4] is played through a Head Mounted Display (HMD) device, which typically needs high computing and graphics rendering capabilities and is often used with a pair of controllers for hand gestures and input devices. There are two distinct types of experiences:

- **360° images applications**, use panoramic images of real environment that can be virtually visited by the user, it is possible to play even on mobile devices.

- **3D simulations applications**, place 3D objects in a virtual environment that can be manipulated and interactable, this type of content needs devices with high specification (heavily used in gaming, simulations and training).

4.1.3 MR-based content

The Mixed Reality (MR) merges the best of both Augmented Reality (AR) and VR technologies, this type of content is characterized by the fusion of 3D synthetic objects and real world environments. MR technology allows users to interact with digital objects that co-exists with the tangible physical world, they can be even anchored to real world locations to be viewed through devices that need cameras such as HMD devices, smartphones or glasses. This technology is growing rapidly and the rate of adoption of devices capable of these features will increase in the near future.

4.2 Smart Learning and Immersive experiences

4.2.1 Standards

4.2.1.1 Overview

Clause 4.2.1 will present standards and de-facto standards about technical requirements, data models, implementation and User Interface/User eXperience (UI/UX).

At the moment, there are no available standards [i.4] for eXtended Reality (XR) experiences themselves or their implementations. However, there are standards related to peripherals, safety of HMD devices, image processing, 3D rendering, and audio; some of these are the same as those used in other software, such as video games or interactive experiences. XAPI and SCORM are the two standards in e-learning experiences data model, eXperience API (XAPI) in particular is the one used the most in interactive immersive learning experiences. The IEEE and Khronos group [i.4] are the main organizations that work on de-facto standards on VR and MR technologies, Khronos Group released OpenXR (open standard for virtual and augmented reality), that makes possible for developers to write code that runs everywhere, so it is a common interface for immersive applications.

4.2.1.2 XAPI

xAPI is a specification for learning technologies developed by Advanced Distributed Learning (ADL) [i.6] that makes possible to collect data about the online and offline experiences that users may have. APIs (Application Programming Interfaces) capture data from different technologies in a consistent format. In this way, very different systems can communicate securely by collecting and sharing the flow of activities with the xAPI vocabulary. xAPI has a syntax based on basic statements: an **actor**, a **verb** and an **object**.

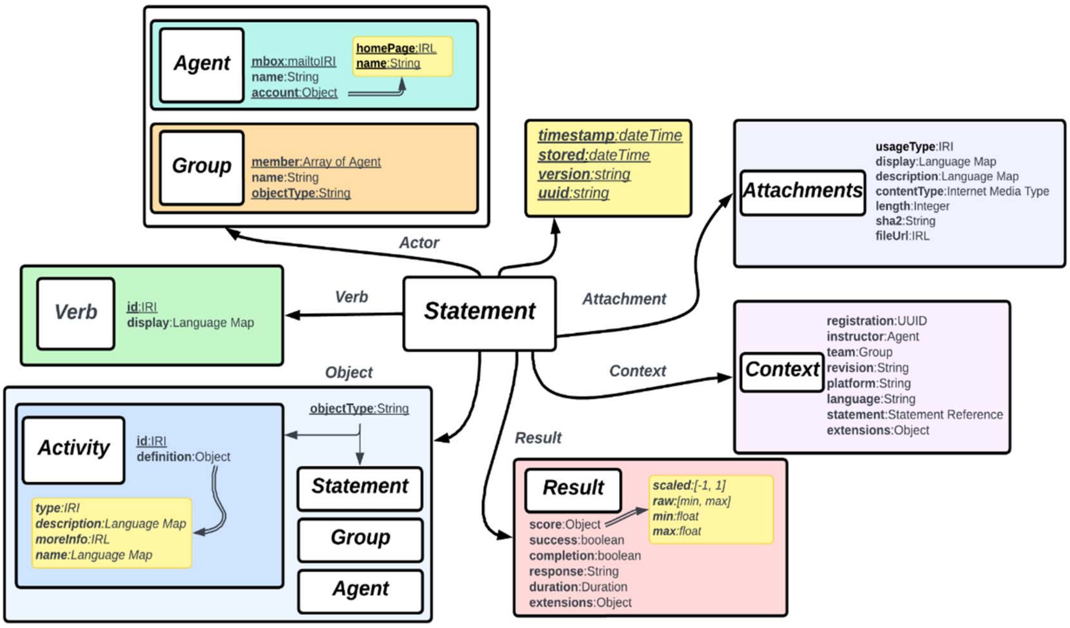


Figure 4.2.1.2-1: xAPI statement's diagram

The statement can also be expanded by adding more pieces of information; in Figure 4.2.1.2-1 on the left are represented the three basic main components of an xAPI statement, the optional components are on the right.

In the proposed statement are also present the result and context objects in addition to the actor, verb and object which are mandatory.

```
{
  "actor": {
    "name": "Sally",
    "mbox": "mailto:sally@example.com"
  },
  "verb": {
    "id": "http://adlnet.gov/expapi/verbs/completed",
    "display": { "en-US": "completed" }
  },
  "object": {
    "id": "http://example.com/activities/solo-hang-gliding",
    "definition": {
      "type": "http://adlnet.gov/expapi/activities/assessment",
      "name": { "en-US": "Solo Hang Gliding" },
      "extensions": {
        "http://example.com/gliderClubId": "test-435"
      }
    }
  },
  "result": {
    "completion":true,
    "success": true,
    "extensions": {
      "http://example.com/flight/averagePitch": 0.05
    }
  },
  "context": {
    "extensions": {
      "http://example.com/weatherConditions": "rainy"
    }
  }
}
```

The verb element allows the recording of the individual passages of which the student's interaction with a content is composed. From the tracking it emerges that the student has chosen a certain type of activity (attempted) and has begun to do it (accessed), has progressed and answered the questions provided by that activity (answered), has completed it (completed) and has successfully passed it (passed) up to the highest level of comprehension and competence (mastered). Below is a list of the main components of an xAPI statement:

- **Actor**, the subject of the action performed, can be an **Agent,** an Anonymous or Identified **Group** (of Agent) [i.7]:

Table 4.2.1.2-1

Agent property			
Attribute name	**Type**	**Description**	**Required**
name	String	The complete name of the user.	Optional.
objectType	String	The value of this attribute has to be "Agent".	Required only if Agent is used as object of a statement
An Inverse Functional Identifier at choice (**Table 4.2.1.2-4**)			Required

Table 4.2.1.2-2

Anonymous Group property			
Attribute name	**Type**	**Description**	**Required**
objectType	String	The value of this attribute has to be "Group".	Required
name	String	The name of the Group.	Optional
member	Array of Agent Objects	The member list of the Group.	Required

Table 4.2.1.2-3

Identified Group property			
Attribute name	**Type**	**Description**	**Required**
objectType	String	The value of this attribute has to be "Group".	Required
name	String	The name of the Group.	Optional
member	Array of Agent Objects	The member list of the Group.	Optional
An Inverse Functional Identifier at choice (**Table 4.2.1.2-4**)			Required

Table 4.2.1.2-4

Inverse Functional Identifiers		
Attribute name	**Type**	**Description**
openid	URI	An openID that uniquely identifies the Agent.
account	Object	Object that incapsulate information about a user existing account on e.g. LMS or intranet (**Table 4.2.1.2-5**).
mbox	mailtoIRI	The email address of the user.
mbox_sha1sum	String	The hex-encoded SHA1 hash of a mail to IRI (i.e. the value of an mbox property).

Table 4.2.1.2-5

Account Object			
Attribute name	**Type**	**Description**	**Required**
homePage	IRL	The canonical home page for the system the account is on.	Required
name	String	The unique id or name used to log in to this account. This is based on FOAF's accountName.	Required

- **Verb**, the action performed by the Actor [i.7]:

Table 4.2.1.2-6

Verb property			
Attribute name	**Type**	**Description**	**Required**
id	IRI	Corresponds to a Verb definition. Each Verb definition corresponds to the meaning of a Verb, not the word.	Required
display	LanguageProperty	The human readable representation of the Verb in one or more languages. It serves as a description.	Recommended

- **Object,** the object of the action performed can be a **Statement, Group, Agent** or an **Activity.** For example, "Ken pressed a button", "Marie interviewed Ken", "Marie commented on 'Ken pressed a button' " are all different actions that imply different object of the action itself [i.7]:

Table 4.2.1.2-7

Object of Activity type			
Attribute name	**Type**	**Description**	**Required**
objectType	String	HAS TO be Activity when present.	Optional
id	IRI	An identifier for a single unique Activity.	Required
definition	Object	Metadata of the activity (**Table 4.2.1.2-8**).	Optional

Table 4.2.1.2-8

Definition Object			
Attribute name	**Type**	**Description**	**Required**
name	LanguageProperty	The human readable/visual name of the Activity.	Recommended
description	LanguageProperty	A description of the Activity.	Recommended
type	IRI	The type of Activity.	Recommended
moreInfo	IRL	Resolves to a document with human-readable information about the Activity.	Optional
Interaction properties, more at [i.7].			
extensions	Object	A map of other properties as needed.	Optional

Table 4.2.1.2-9

Object of StatementRef type			
Attribute name	**Type**	**Description**	**Required**
objectType	String	HAS TO be StatementRef.	Required
id	UUID	The UUID of the Statement used as Object.	Required
NOTE: In this case the Object is a pre-existing Statement referred by its UUID [i.7].			

Table 4.2.1.2-10

Object of SubStatement type			
Attribute name	**Type**	**Description**	**Required**
objectType	String	HAS TO be SubStatement.	Required
NOTE: In this case the Object is a new Statement, called SubStatement, which is similar to a StatementRef but self contained into a Statement, it does not represent an event that has already occurred. The SubStatement has to be a valid Statement in addition to other SubStatement requirements like not containing "id", "stored", "version" or "authority" properties [i.7].			

Here is an example of Object of SubStatement type [i.7]:

```
{
    "actor": {
        "objectType": "Agent",
        "mbox":"mailto:test@example.com"
    },
    "verb" : {
        "id":"http://example.com/planned",
        "display":{
            "en-US":"planned"
        }
    },
    "object": {
        "objectType": "SubStatement",
        "actor" : {
            "objectType": "Agent",
            "mbox":"mailto:test@example.com"
        },
        "verb" : {
            "id":"http://example.com/visited",
            "display":{
                "en-US":"will visit"
            }
        },
        "object": {
            "objectType": "Activity",
            "id":"http://example.com/website",
            "definition": {
                "name" : {
                    "en-US":"Some Awesome Website"
                }
            }
        }
    }
}
```

Table 4.2.1.2-11

Result property			
Attribute name	**Type**	**Description**	**Required**
score	Object	The score of the Agent in relation to the success or quality of the experience (**Table 4.2.1.2-12**).	Optional
success	Boolean	Indicates whether or not the attempt on the Activity was successful.	Optional
completion	Boolean	Indicates whether or not the Activity was completed.	Optional
response	String	A response appropriately formatted for the given Activity.	Optional
duration	Duration	Period of time over which the Statement occurred.	Optional
extensions	Object	A map of other properties as needed.	Optional
NOTE: **Result** is an optional property that represents the outcome of the related Statement [i.7].			

Table 4.2.1.2-12

Score Object			
Attribute name	**Type**	**Description**	**Required**
scaled	Decimal number between -1 and 1, inclusive	The score related to the experience as modified by scaling and/or normalization.	Recommended
raw	Decimal number between min and max (if present, otherwise unrestricted), inclusive	The score achieved by the Actor in the experience described by the Statement. This is not modified by any scaling or normalization.	Optional
min	Decimal number less than max (if present)	The lowest possible score for the experience described by the Statement.	Optional
max	Decimal number greater than min (if present)	The highest possible score for the experience described by the Statement.	Optional

Table 4.2.1.2-13

Context property			
Attribute name	**Type**	**Description**	**Required**
registration	UUID	The registration that the Statement is associated with.	Optional
instructor	Agent (MAY be a Group)	Instructor that the Statement relates to, if not included as the Actor of the Statement.	Optional
team	Group	Team that this Statement relates to, if not included as the Actor of the Statement.	Optional
contextActivities	contextActivities Object	A map of the types of learning activity context that this Statement is related to. Valid context types are: parent, "grouping", "category" and "other".	Optional
revision	String	Revision of the learning activity associated with this Statement. Format is free.	Optional
platform	String	Platform used in the experience of this learning activity.	Optional
language	String (as defined in IETF RFC 5646 [i.15])	Code representing the language in which the experience being recorded in this Statement (mainly) occurred in, if applicable and known.	Optional
statement	Statement Reference	Another Statement to be considered as context for this Statement (**Table 4.2.1.2-9**).	Optional
extensions	Object	A map of any other domain-specific context relevant to this Statement. For example, in a flight simulator altitude, airspeed, wind, attitude, GPS coordinates might all be relevant.	Optional
NOTE: **Context,** is an optional property that adds contextual information to the Statement. It can contain information about the instructor, the environment of the activity and more [i.7].			

- **Attachment,** can be an important part of the Learning experience, it could be an essay, a video, an image, etc. These Attachments are meant to be stored in and retrievable from an LRS [i.6], [i.7].

- **Extensions**, are available in different part of an xAPI statement as the result and context properties. In every case, extensions are meant to add information and to extend those properties for specific use. The content can be relevant to only one application or it can be a standard used in many scenarios. Extensions are in form of a map with key:values format, the specification makes clear that the keys of an extensions map has to be IRIs [i.7] and that the values of an extensions can be any JSON value or data structure [i.7]. The meaning of extensions mapping values is defined by the developer or organization that controls the IRI of the key. Here is an example of extensions used in the result property of a statement:

```
"result": {
        "extensions": {
            "http://example.com/profiles/meetings/extensions/reviewlocation":
"C:\\meetings\\minutes\\examplemeeting.one"
        },
        "success": true,
        "completion": true,
        "response": "We agreed on some example actions.",
        "duration": "PT1H0M0S"
}
```

The example proposes highlights on how extensions can enrich statements information and adding more clues, in this case, on the result of the learning experience.

4.2.1.3 SCORM

Sharable Content Object Reference Model (SCORM) is a specification for learning technologies developed by ADL [i.8] to address e-learning interoperability. It is heavily used in today e-learning experiences even if xAPI is proposed as the new standard that, one day, could replace it. The main concept of SCORM is the Sharable Content Object (SCO) that defines a certain training unit or experience even if small in scale. The educational content has to be a stand alone experience (it cannot depend on other external content or other SCO). The LMS has the role of distributing the SCOs to learners and taking track of progress. **SCORM 1.2** and **SCORM 2004 3rd Edition** are the main versions of SCORM, but as of today SCORM 1.2 is used more than the 2004 version.

SCORM 2004 3rd Edition added new features that were not available in version 1.2, but even after that, SCORM 1.2 is the most used version of the standard today thanks to its ease of adoption. Here is a breakdown of the key elements of cmi.core data models:

1) Student Information:
   - **cmi.core.student_id:** Unique ID of the student (Read-Only).
   - **cmi.core.student_name:** Name of the student (Read-Only).
2) Lesson Information:
   - **cmi.core.lesson_location:** Current location in the lesson (Read/Write).
   - **cmi.core.credit:** Whether the learner will be credited for the lesson ("credit" or "no-credit") (Read-Only).
   - **cmi.core.lesson_status:** Completion status of the lesson ("passed", "completed", "failed", "incomplete", "browsed", "not attempted") (Read/Write).
   - **cmi.core.entry:** Indicates if the learner is starting the lesson anew or resuming ("ab-initio", "resume", "") (Read-Only).
3) Scoring Information:
   - **cmi.core.score.raw:** The learner's score (Read/Write).
   - **cmi.core.score.max:** Maximum possible score (Read/Write).
   - **cmi.core.score.min:** Minimum possible score (Read/Write).
4) Session Information:
   - **cmi.core.total_time:** Total time spent in the lesson (Read-Only).
   - **cmi.core.session_time:** Time spent in the current session (Write-Only).
   - **cmi.core.exit:** How the learner left the lesson ("time-out", "suspend", "logout", "") (Write-Only).

Here is a breakdown of the key elements of cmi data models:

1) Suspend and Launch Data:
   - **cmi.suspend_data:** Stores data between sessions (Read/Write).
   - **cmi.launch_data:** Data provided to the lesson when it starts (Read-Only).
2) Comments.
3) **cmi.comments:** Learner's comments about the lesson (Read/Write).
4) **cmi.comments_from_lms:** Comments from the LMS about the lesson (Read-Only).
5) Objectives:
   - **cmi.objectives._count:** Number of objectives stored (Read-Only).
   - **cmi.objectives.n.id:** Unique ID for each objective (Read/Write).
   - **cmi.objectives.n.score.raw:** Score for each objective (Read/Write).
   - **cmi.objectives.n.score.max:** Maximum score for each objective (Read/Write).
   - **cmi.objectives.n.score.min:** Minimum score for each objective (Read/Write).
   - **cmi.objectives.n.status:** Completion status of each objective ("passed", "completed", "failed", "incomplete", "browsed", "not attempted") (Read/Write).

6) Student Data:

- **cmi.student_data.mastery_score:** Score needed to master the lesson (Read-Only).
- **cmi.student_data.time_limit_action:** Action to take when time limit is exceeded ("exit,message", "exit,no message", "continue,message", "continue, no message") (Read-Only).

7) Interactions:

- **cmi.interactions.n.result:** Result of an interaction ("correct", "wrong", "unanticipated", "neutral", or a decimal value) (Write-Only).

4.2.1.4 ETSI ISG ARF

Augmented Reality Framework (ARF) is an ETSI Industry Specification Group (ISG, like ETSI ISG CIM) that developed the Group Specification ETSI GS ARF 004-2 [i.10]. It is a framework for immersive and functional AR experiences designed by the ETSI ISG ARF; it is an important standard in the immersive AR experiences landscape as it gives a well formed workflow of steps for the creation of such applications.

The framework is composed by **different technologies** and **functions**, one of the main concepts of the standard (and the one that will be covered the most in the present document) is the **World Storage function** (in ETSI GS ARF 004-2 [i.10] that is described all the steps required to help the immersive application to **maintain the relative poses of object** in the real world thanks to concepts like **Trackables, GeoTrackables, Features** and **World Anchors**:

- **Trackables:** "*are models of parts of the real world […] of which features are available and/or could be extracted. [They] provide a Coordinate Reference System in which a pose can be expressed.*"
- **GeoTrackables:** "*are set of positions and orientation on earth in a geodetic referential.*"
- **Features:** "*are characteristics of a real world element that can be searched, recognized or tracked […] [They] are extracted from Trackables and store in a World Storage function. Positions and orientations of Features shall be expressed in the Coordinate Reference System of the Trackable from which they have been extracted.*"
- **World Anchors:** "*represents a fixed position in relation to one or more elements of the real world […]. [They] shall define a Coordinate Reference System of the real world […], may have zero, one or several AR Assets placed on its coordinate reference system.*"

Using a combination of **Trackables** of different nature (like QR Codes, images, objects and signs) and **World Anchors** it is possible to determine the **right pose** relative to real world points of interest. Trackables and World Anchors **need to be linked and related** to one another so that the **World Storage function** can **query** the **World Graph** created (through API calls) and return the wanted **spatial information**; the data is then **processed** by the other components and function (e.g. the **World Analysis function**) and passed to the client (e.g. the AR application).

The schema for entities like "Trackables" is:


```
"UUID": "fa8bbe40-8052-11ec-a8a3-0242ac120002",
"name": "myTrackableXYZ",
"creatorUUID": "bd6ce7ce-7fe8-487d-a179-fddfe914f293",
"trackableType": "string",
"trackableEncodingInformation": "string",
"trackablePayload": "10110101",
"localCRS": "string",
"unit": "string",
"trackableSize": "string",
"keyvalueTags": {
  "Place": [
      "Building 123"
    ],
  "Room": [
      "007"
    ]
```

```
    },
    "confidence": 50
```


The property "keyvalueTags" is used to store meta data attached to the main entity. Following ETSI GS ARF 004-2 [i.10], AR Objects are referred to as AR assets. An AR Asset is attached to a Trackable or a World Anchor (components of the world graph in the World Storage) and a 3D transform describes the relative position and rotation between them. In the API, a 3D transform is described as a 4X4 matrix. At runtime, Trackables poses are updated in real time in a cartesian coordinate system in order to update the poses of the World Anchors and AR assets. GeoTrackable is a specific type of Trackable that can be included in the world graph allowing the system to transform a pose from a local cartesian coordinate system to a global geodetic coordinate system.

4.2.2 De-facto standards

4.2.2.1 Overview

Clause 4.2.2 de-facto standards will be presented in a high level of details.

4.2.2.2 OpenXR

OpenXR [i.12] is an open standard developed to streamline the development of VR and AR applications, it focuses on **graphics and physics API** to enable immersive experiences. OpenXR allows developers to write code that works across a wide range of hardware devices (mobile, desktop, HMD devices, game consoles) and platforms without particular adjustments, reducing fragmentation in the immersive technology landscape.

4.2.2.3 WebXR

WebXR [i.13] is an API developed to develop VR and AR experiences to web browsers; it provides a standardized framework for creating immersive experiences that can run in a browser without requiring dedicated applications or software.

4.2.3 Common methodologies and best practices

Clause 4.2.3 will discuss methodologies and best practices for Smart Learning experiences that are commonly used in real world products. There will be different classifications of methodologies and best practices, based on the type of content, some of which are the result of only research and review in the educational field.

The six fundamentals [i.3] in smart learning that can make the difference in immersive technologies can be summarized as follows:

1) **Collaborative Learning:** In today's educational landscape collaborative learning environments are more important than ever. By encouraging users to collaborate on projects and assignments, educators facilitate the development of crucial interpersonal skills such as teamwork, communication, and conflict resolution. Moreover, collaborative learning enhances critical thinking as students engage in discussions, share diverse perspectives, and collectively problem-solve. Through collaborative endeavours, students not only deepen their understanding of the subject or the task matter but also develop skills essential for success in various personal and professional contexts.

2) **Personalized Content:** Recognizing that every user has unique learning styles, preferences, academic or professional needs, the integration of personalized content into learning practices has become essential. Tailoring learning materials and activities to align with each student's individual strengths, weaknesses, and interests enhances their overall learning experience. Personalized content ensures that learners are appropriately challenged, remain motivated, and can progress at their own pace. Whether through adaptive software, differentiated instruction, or varied assessment strategies, personalized content empowers educators and tutors to cater to the diverse needs of their students, thereby maximizing learning outcomes.

3) **Adaptivity:** The implementation of adaptive learning technologies represents a significant advancement in educational methodologies. These technologies leverage algorithms to analyse students' performance data and adapt learning experiences in real-time. By assessing individual learning styles, comprehension levels, and progress, adaptive learning technologies personalize instruction to meet each learner's needs effectively. Whether through adjusting the difficulty of tasks, providing targeted feedback, or offering supplementary resources, adaptivity ensures that learners receive tailored support and guidance, ultimately optimizing their learning journey.

4) **Context-Awareness:** With the proliferation of mobile and ubiquitous computing, integrating context-aware technologies into educational settings has emerged as a promising approach to enhance learning experiences. Context-aware systems leverage sensors and data analytics to perceive and respond to students' contextual cues, such as location, time, and activity. By dynamically adjusting content and interactions based on these contextual factors, educators can create immersive and personalized learning environments. Whether by delivering relevant content based on a student's physical surroundings or adapting instructional strategies to accommodate situational constraints, context-awareness enriches learning experiences by making them more relevant, engaging, and adaptive.

5) **Facilitation of Interaction:** Interaction lies at the heart of effective learning experiences, promoting active engagement, collaboration, and knowledge construction. Educational tools and platforms designed to facilitate interaction empower students to participate actively in the learning process, whether through discussions, debates, simulations, or collaborative projects. By leveraging interactive technologies such as online forums, virtual classrooms, and multimedia resources, educators can create dynamic and participatory learning environments that cater to diverse learning styles and preferences. Through fostering meaningful interactions among students and between students and instructors, educators nurture a vibrant learning community conducive to deeper understanding and knowledge retention.

6) **Intelligent Recommendation:** In the era of information abundance, guiding users towards relevant and high-quality learning resources can be a daunting task. Intelligent recommendation systems alleviate this challenge by leveraging data analytics and machine learning algorithms to provide personalized recommendations tailored to each student's needs and preferences. By analysing factors such as past performance, interests, and learning goals, these systems suggest supplementary materials, enrichment activities, or further reading that complement and extend the curriculum. Intelligent recommendations not only support self-directed learning but also foster curiosity, exploration, and lifelong learning habits among students, empowering them to take ownership of their educational journey and pursue areas of interest with enthusiasm and confidence.

There are best practices that all HMD device manufacturers publish to help developers make and deploy applications for their systems, these guidelines are useful from the first-time XR designer to the more experienced one. Here is a collection of some of the best practices that major XR-involved companies made publicly available for developers around the world, many of these principles are in common between different technologies (AR and VR) [i.2].

In general, as these **users** start to make use of the software, **they should not need to learn how to interact with familiar virtual objects**. By aligning virtual interactions with common, real-world experiences, developers can create a seamless and intuitive user experience. This approach not only enhances usability but also grows user engagement and satisfaction. For instance, when users encounter virtual objects that resemble handles or buttons, they should be able to interact with them in a natural and intuitive manner, such as grabbing the handle to open a virtual door or pressing a button to activate a virtual device. This familiarity eliminates the need for users to relearn new interaction techniques, thus streamlining the user experience and reducing cognitive load.

Furthermore, ensuring that **virtual objects adhere to realistic physics principles** enhances immersion and believability. Objects should feel grounded within the virtual environment, meaning they should not pass through walls or fall through the floor unexpectedly. This realism reinforces the user's sense of presence and adds to the overall credibility of the virtual experience. Additionally, proper spatial arrangement of virtual objects is crucial for maintaining consistency with real-world perceptions. Objects should be scaled and positioned in a way that accurately reflect their size and depth relative to the user's perspective. The alignment with reality helps users navigate the virtual environment more effectively and reduces confusion or disorientation; this approach promotes usability, enhances user engagement, and contributes to a more satisfying overall experience in virtual environments. The designer can add in the scene portals, windows and doors that lead into virtual worlds, or even bring remote users (as avatars) into a shared environment; connect virtual content with real world objects, like virtual buttons to control a smart light bulb.

The user should be able to easily see distinct objects, **even if they are virtual and blended into the real world** environment, in the same way shadow, lighting, dimming and object occlusion are all graphical effects that contribute making the experience more immersive and realistic and can facilitate the interaction with the objects instantiated in the scene. Graphical effects and rendering techniques should also increase real-world awareness in HMD based experiences. Those are commonly the more complex experiences among the MR range and users could not be able to easily understand the limits of the virtual content merged into the real world; this could lead to safety problems about cognitive loads, spatial awareness, motion sickness, disorientation and other sensorial issues; users should be warned from time to time to take a break [i.2], [i.4] and [i.5].

The UI of immersive content should be designed to ease the overall user's experience. For instance, menus should move smoothly with the user and always ready to be interacted with.

Furthermore, the following UI elements are commonly used in this context:

- Tooltips
- Ray cast elements
- Highlight effects

The above UI elements are very effective because they can help the user to select objects to interact with, they can be used to display objects around the user and they can show the boundaries if needed [i.2]. Objects should not be placed near boundaries to avoid unintended contacts with real world objects.

The user should be able to take on tasks and experiences without spending too much time in log-in screens which require specific input that, at times, are not natural to execute in a virtual mixed environment. This need of ease of use and fast starting makes it clear that this would be appropriate to interconnect the entire ecosystem used as much as possible.

5 Mapping of xAPI to NGSI-LD

5.1 Preliminary Overview

5.1.0 Foreword

Clause 5.1 will discuss more about xAPI and NGSI-LD as well as the mapping and the capabilities that NGSI-LD can add to the smart learning scene.

5.1.1 Similarities with NGSI-LD

1) **Semantic Interoperability:** Both xAPI and NGSI-LD advocate for semantic interoperability, facilitating the exchange of data in a standardized and machine-understandable format. While xAPI employs Activity Streams and the JSON-LD format to express learning experiences, NGSI-LD leverages RDF triples to represent entities and their relationships in the smart city context. Despite differences in domain-specific vocabularies, the underlying principles of semantic coherence remain consistent across both frameworks.

2) **Linked Data Principles:** At the core of NGSI-LD lies the foundational principles of Linked Data, which emphasize the publication and interlinking of structured data on the web. Similarly, xAPI's specification encourages the creation of interoperable learning data by employing URIs to reference resources and enabling the aggregation of learning experiences across distributed platforms. By adhering to Linked Data principles, both frameworks facilitate the discovery and integration of diverse data sources, fostering a holistic understanding of complex environments.

3) **RESTful APIs:** Leveraging RESTful APIs, xAPI and NGSI-LD provide standardized interfaces for accessing and manipulating data, ensuring compatibility with existing web technologies. While xAPI utilizes RESTful endpoints for querying learning records and statements, NGSI-LD exposes entities and their attributes through HTTP-based interfaces. By embracing RESTful architectural principles, both frameworks enable seamless integration within distributed environments, promoting scalability and extensibility.

4) **Contextual Data:** Recognizing the importance of contextual information, both xAPI and NGSI-LD support the capture and exchange of metadata alongside primary data. In the context of xAPI, contextual data enriches learning experiences by providing insights into the learning environment, user interactions, and performance metrics. Similarly, NGSI-LD captures contextual information pertaining to the physical environment, sensor observations, and situational awareness within smart city deployments. By incorporating contextual metadata, both frameworks enable a nuanced understanding of complex phenomena, facilitating informed decision-making and analytics.

5) **Interoperability Standards:** While xAPI and NGSI-LD serve distinct domains-learning ecosystems and smart city infrastructures respectively - they adhere to interoperability standards to ensure compatibility with heterogeneous systems and technologies. By aligning with established protocols such as JSON-LD, RDF, and HTTP, both frameworks facilitate seamless integration within diverse ecosystems, mitigating interoperability challenges and promoting data interoperability at scale.

5.1.2 Limitations

Clause 5.1.2 will discuss limitations of both xAPI and NGSI-LD:

- xAPI is not capable of referencing to the same student or learner by different email addresses, social media accounts, institutional accounts, so it is not easy to unify and collect unique information across different domains and experiences; NGSI-LD can easily do it thanks to its query language and the brokers' implementations and capabilities.

- It is worth noting that many words used in xAPI (such as success, min, max) or more in general in the e-learning scene, are already used in the core context of NGSI-LD. This fact limits the usage of those words.

5.2 Mapping

5.2.1 Overview

Clause 5.2.2 to clause 5.2.8 will focus on the mapping of xAPI in NGSI-LD and what NGSI-LD can add to smart learning experiences.

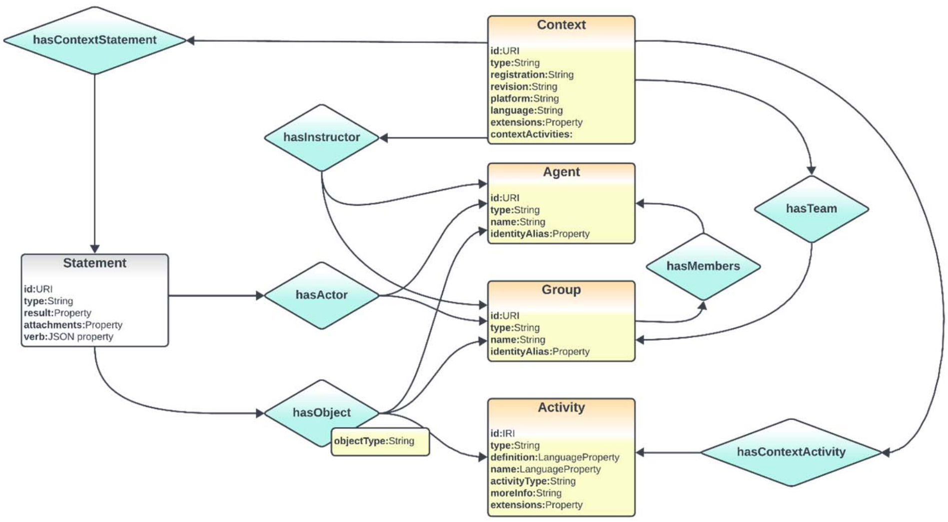


Figure 5.2.1-1: Overview of the mapping

5.2.2 Statement

Clause 5.2.2 will focus on the mapping of xAPI Statement object in a NGSI-LD entity. Complex properties and relationships will be covered in clause 5.2.3.

Table 5.2.2-1

Properties			
Name	**Type**	**Description**	**Required**
id	String	An URI that identifies the Entity.	Required
type	Entity Type name or an unordered JSON array	The types of the Entity is "Statement".	Required
verb	JSON property	The verb related to the statement.	Required
result	JSON object	The result of the action.	Optional
attachments	JSON object	The attachments related to the action.	Optional

Table 5.2.2-2

Relationships			
Name	**Type**	**Description**	**Required**
hasActor	Array of Agents or Groups	The Actor of the Statement.	Required
hasObject	Array of Agents, Groups or Activity	The Object which the Statement refers to. It is recommended to use the property of relationship "objectType" to identify the type of the Object entity (that can be "agent", "group" or "activity").	Required

5.2.3 Actor

5.2.3.1 Overview

Clause 5.2.3 will focus on the mapping of xAPI Actor object in a NGSI-LD entity. From xAPI specs [i.7] the Actor is the entity that performs the action represented by the statement itself; it can be of type Agent or Group. The following clause 5.2.3.2 and clause 5.2.3.3 will cover these two cases.

5.2.3.2 Agent

The Agent object from xAPI (covered in table 4.2.1.2-1) can be mapped in NGSI-LD as an entity of type Agent; it seems appropriate to represent an Agent as an entity with its lifecycle and properties, as each agent is unique (typically identified by an e-mail address) and engages in various actions across different environments. Below is the mapping for entities of Agent type.

Table 5.2.3.2-1

Properties			
Name	**Type**	**Description**	**Required**
id	String	An URI that identifies the Entity.	Required
type	Entity Type name or an unordered JSON array	The types of the Entity is "Agent".	Required
name	String	The complete name of the user.	Optional
identityAlias	JSON object	It can contain a mbox, openId or other identifiers used in experiences; it should contain at least one of this unique identifiers.	Required

It is worth noting that the identityAlias property is used to preserve and improve the different methods to identify an Agent in xAPI. In fact in xAPI it is not possible to refer to an Agent by using different keys (stated in depth in clause 4.2.1.2). By using this property it will be possible to make queries and operations by alias (e.g. e-mail, social account). Here is an example of an Agent entity in NGSI-LD:

```
{
        "id": "Agent:1",
        "type": "Agent",
"name": {
            "type": "Property",
            "value": "Billy Black",
            "observedAt": "2023-11-24T16:21:11.56Z"
        },
        "identityAlias": {
            "type": "Property",
            "value": {
                "mbox": "billyb@test.com",
                "openId": "12345"
            }
        }
        "@context": [
            "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.7.jsonld"
        ]
}
```

QUERY EXAMPLE 1: `?q=identityAlias[mbox]=="billyb@test.com".`

The `identityAlias` property, a complex JSON object, contains different subitems (any identifier that refers to the Agent); in this case it is wanted to inspect the mbox subitem and check if its value is equal to the desired one. With this approach it can be possible to collect actions performed by an agent even if executed in different platforms, environments or applications that refer to the same person (agent) with different identifiers (e.g. social network account, institutional applications).

5.2.3.3 Group

In xAPI a Group is made up of Agents, in NGSI-LD a Group is an Entity that presents a relationship *hasMembers* that points to an array of Agent entities.

Table 5.2.3.3-1

Properties			
Name	**Type**	**Description**	**Required**
id	String	An URI that identifies the Entity.	Required
type	Entity Type name or an unordered JSON array	The type of the Entity is "Group".	Required
name	String	The name of the Group.	Optional
identityAlias	JSON object	It can contain a mbox, openId or other identifiers used in experiences; it should contain at least one of this unique identifiers.	Required only if it is an Identified Group, NOT compatible with Anonymous Group concept

Table 5.2.3.3-2

Relationships			
Name	**Type**	**Description**	**Required**
hasMembers	Array of Agents	The participants of the Group.	Required if it is an Anonymous Group, Optional otherwise

An example of a Group entity:

```
{
        "id": "Group:1",
        "type": "Group",
"name": {
            "type": "Property",
            "value": "LAB12_R&D"
        },
"identityAlias": {
            "type": "Property",
            "value": {
                "mbox": "rd_lab12@test.com"
            }
        },
"hasMembers": {
            "type": "Relationship",
            "objects": ["Agent:1", "Agent:2", "Agent:3"]
        },
        "@context": [
            "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.7.jsonld"
        ]
}
```

5.2.4 Verb

Clause 5.2.4 will focus on the mapping of xAPI Verb object in a NGSI-LD property. The Verb object is mapped as a JSON property of the Statement Entity itself.

Table 5.2.4-1

Name	Type	Description	Required
verb	JSON property	The action performed by the Actor.	Required

Table 5.2.4-2

Verb object			
Name	**Type**	**Description**	**Required**
id	String	A URI definition that corresponds to the meaning of a Verb, not the word.	Required
display	LanguageProperty	The human readable name of the Verb in one or more languages.	Recommended

An example of the verb property usage:

```
{
        "id": "Statement:1",
        "type": "Statement",
            . . .
"verb": {
            "type": "JsonProperty",
            "json": {
                        "id": "https://example.org/verbs/planned",
                        "display": {
                            "en-US": "planned",
                            "it": "pianificato"
}
  }
        }
            . . .
        "@context": [
            "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.7.jsonld"
        ]
}
```

In xAPI the Verb can also give an update on the activity (e.g. if verb is started, completed, attempted) but this type of information is carried out by the Result object, so with this clarification in mind it is clear that the verb can be a property (even a static one, so that its value does not change throughout the life cycle of the statement).

5.2.5 Object

5.2.5.1 Overview

Clause 5.2.5.1 will focus on the mapping of xAPI Object property in a NGSI-LD. From xAPI specs [i.7], an Object can be of type Agent, Group or Activity. In clause 5.2.3 it is explained how to refer to different types of Objects of Statements. The mapping in this case is to have a Relationship that points to an Agent, a Group or an Activity entity, the "objectType" Property of the "hasObject" Relationship represents the objectType property present in xAPI.

Table 5.2.5.1-1

hasObject relationship			
Name	**Type**	**Description**	**Required**
objectType	String	A string that identifies the object type, it can be: "Agent", "Group" or "Activity".	Required

5.2.5.2 Activity

Activity is mapped as a NGSI-LD entity as follows:

Table 5.2.5.2-1

Properties			
Name	**Type**	**Description**	**Required**
id	String	An URI that identifies the Entity.	Required
type	Entity Type name or an unordered JSON array	The type of the Entity is "Activity".	Required
name	LanguageProperty	The visual name of the Activity.	Required
description	LanguageProperty	A description of the Activity.	Recommended
activityType	String	An IRI that represents the type of the Activity.	Recommended
moreInfo	String	It points to a document with information about the Activity, it can also include a way to launch the activity itself.	Optional
extensions	JSON object	A map of other properties.	Optional
interaction	JSON object	A property that maps the SCORM 2004 4th Edition Data Model, based on xAPI spec.	Optional

Table 5.2.5.2-2

Interaction object			
Name	**Type**	**Description**	**Required**
interactionType	String	The type of interaction. Possible values are: true-false, choice, fill-in, long-fill-in, matching, performance, sequencing, likert, numeric or other.	Required
correctResponsesPattern	Array of Strings	A pattern representing the correct response to the interaction. The structure of this pattern varies depending on the interactionType. More details at [i.7].	Optional
Only one or two other sub properties among those listed below based on the specs at [i.7], accordingly to the interactionType value.			
choices	JSON property	A list of the options available in the interaction for selection or ordering (only if interactionType is equal to "choice" or "sequencing").	Optional
scale	JSON property	A list of the options on the likert scale (only if interactionType is equal to "likert").	Optional
source	JSON property	Lists of sources (only if interactionType is equal to "matching").	Optional
target	JSON property	Lists of targets (only if interactionType is equal to "matching").	Optional
steps	JSON property	A list of the elements making up the performance interaction (only if interactionType is equal to "performance").	Optional

An example of Activity entity:

```
{
        "id": "Activity:1",
        "type": "Activity",
"name": {
            "type": "LanguageProperty",
            "languageMaps":[{"languageMap": {"en": "Workplace Safety Course 1",
                      "it": "Corso Sicurezza sul lavoro pt. 1"}, "2024-05-16T12:00:00Z}]
        },
"activityType": {
            "type": "Property",
            "value": "https://example.com/safety/workplaces/course/1"
        },
        "@context": [
            "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.7.jsonld"
        ]
}
```

An example of Activity entity with other optional properties:

```
{
        "id": "Activity:1",
        "type": "Activity",
"name": {
            "type": "LanguageProperty",
            "languageMaps":[{"languageMap": {"en": "Workplace Safety Course 1",
                      "it": "Corso Sicurezza sul lavoro pt. 1"}, "2024-05-16T12:00:00Z}]
        },
"activityType": {
            "type": "Property",
            "value": "https://example.com/safety/workplaces/course/1"
        },
        "@context": [
            "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.7.jsonld"
        ]
}
```

5.2.6 Result

Clause 5.2.6 will focus on the mapping of xAPI Result object in a NGSI-LD property.

Table 5.2.6-1

Name	Type	Description	Required
result	JSON object	The result of the action performed by the Actor.	Required

Table 5.2.6-2

Result object			
Name	**Type**	**Description**	**Required**
success	Boolean	Indicates if the Activity was successful.	Optional
completion	Boolean	Indicates if the Activity was completed.	Optional
response	String	A formatted response.	Optional
duration	String	ISO 8601:2019 [i.16] (E) section 4.4.3.2 duration.	Optional
score	JSON object	The score of the Agent related to the experience.	Optional
extensions	JSON property	A map of other properties.	Optional

Table 5.2.6-3

Score object			
Name	**Type**	**Description**	**Required**
scaled	Decimal number between -1 and 1, inclusive.	The score related to the experience as modified by scaling and/or normalization.	Recommended
raw	Decimal number between min and max, inclusive.	The score achieved by the Actor in the experience described by the Statement.	Optional
min	Decimal number less than max (if present).	The lowest possible score for the experience described by the Statement.	Optional
max	Decimal number greater than min (if present).	The highest possible score for the experience described by the Statement.	Optional

An example of the use of the Result property:

```
{
        "id": "Statement:1",
        "type": "Statement",
            . . .
"result": {
            "type": "Property",
            "score": {
                "type": "Property",
                "value": {
                    "max": 100,
                    "min": 50,
                    "raw": 87,
                    "scaled": 0.87
                }
            },
            "value": {
                "success": true,
                "completion": true,
                        "duration": "PT4H35M59.14S",
            }
        }
. . .
        "@context": [
            "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.7.jsonld"
        ]
}
```

5.2.7 Context

Clause 5.2.7 will focus on the mapping of xAPI Context object in a NGSI-LD entity.

Table 5.2.7-1

Properties			
Name	**Type**	**Description**	**Required**
id	String	An URI that identifies the Entity.	Required
type	Entity Type name or an unordered JSON array	The type of the Entity is "Context".	Required
registration	String	The UUID registration that the Statement is associated with.	Optional
contextActivities	contextActivities Object	A map of the types of learning activity context that this Statement is related to. Valid context types are: parent, "grouping", "category" and "other".	Optional
revision	String	Revision of the learning activity associated with this Statement. Format is free.	Optional
platform	String	Platform used in the experience of this learning activity.	Optional
language	String (as defined in IETF RFC 5646 [i.15])	Code representing the language in which the experience being recorded in this Statement (mainly) occurred in, if applicable and known.	Optional
extensions	Object	A map of any other domain-specific context relevant to this Statement. For example, in a flight simulator altitude, airspeed, wind, attitude, GPS coordinates might all be relevant (Table 4.2.1.2-13)	Optional

Table 5.2.7-2

Relationships			
Name	**Type**	**Description**	**Required**
hasInstructor	Array of Agents or Groups	The instructor of the activity if not included as Actor of the Statement itself, it can be a Group.	Optional
hasTeam	Array of Groups	One Team that the Statement relates to.	Optional
hasContextStatement	Array of Statements	Another Statement to be considered as context for the main Statement.	Optional
hasContextActivities	Array of Activities	An array of activities that are related to the statement.	Optional

Example of Context:

```
{
        "id": "Context:1",
        "type": "Context",
"extensions": {
            "type": "Property",
            "value": {
                        "http://example.org/weather/20240507/Rome": "Sunny",
"http://example.org/chemistry/laboratory": "Lab 254F, 2nd floor"
  }
        },
"hasInstructor": {
            "type": "Relationship",
            "object": "Agent:1"
        },
        "hasContextStatement": {
            "type": "Relationship",
            "object": "Statement:2"
        }
        "@context": [
            "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.7.jsonld"
        ]
}
```

5.2.8 Attachments

Clause 5.2.8 will focus on the mapping of xAPI Attachment object in a NGSI-LD property, it is a JSON complex object.

Table 5.2.8-1

Properties			
Name	**Type**	**Description**	**Required**
usageType	String	The IRI that identifies the usage of the Attachment.	Required
display	LanguageProperty	Display name of the Attachment.	Required
description	LanguageProperty	Description of the Attachment.	Optional
contentType	String	The content type of the Attachment (it corresponds to the "Content-Type" HTTP request parameter, refers to [i.9]).	Required
length	Integer	The length of the Attachment in octets (it corresponds to the "Content-Length" HTTP request parameter).	Optional
sha2	String	The SHA-2 hash of the Attachment data (it corresponds to the "X-Experience-API-Hash" HTTP request parameter).	Required
fileUrl	String	The location at which the Attachment data can be retrieved.	Optional

An example of the use of the Attachment property:


```
{
        "id": "Statement:1",
        "type": "Statement",
            . . .
"attachment": {
            "type": "Property",
            "value":
                "usageType": "https://example.org/attachments//certificates",
                        "contentType": "image",
                        "length": 128,
                        "sha2": "227bebaf69ddba3...",
                        "fileUrl": "https://example.org/attachments/certificates/123456789"
            },
 "display":
                "type": "Property",
                        "value": {
            "en-US": "Certificate of attendance",
            "it": "Certificato di partecipazione"
}
            },
                "description":
                "type": "Property",
                        "value": {
            "en-US": "Certificate image",
            "it": "Immagine del certificato"
}
            }
        }
. . .
        "@context": [
            "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.7.jsonld"
        ]
}
```

6 Using NGSI-LD with XR scenarios

6.1 Overview

Clause 6 will discuss guidelines regarding standards, technologies and the mapping proposed in the ETSI ISG CIM and proof of concepts while describing advantages of using NGSI-LD and its Temporal API with XR scenarios, and particularly its use in smart learning environments.

6.2 Guidelines

- The concept of xAPI Statement's "voiding" and how to manage it in NGSI-LD:
    - Voiding is a concept used in xAPI Statements exposed in [i.7]. Basically, in **xAPI it is not possible to update a Statement that was created before**, so after creating and storing an xAPI statement in a LRS, it will be guaranteed as immutable, the only way to tell the LRS that a specific Statement is outdated or invalid (for some reasons) is by creating a new Statement where *verb* is "voided" and *object* refers directly to the Statement to invalidate it. In NGSI-LD it is possible to execute all CRUD operations on a certain entity, so a voided xAPI statement could be (at need) **deleted or updated.**
- How to manage users and their *identityAlias* property:
    - **identityAlias property is intended to increase the interoperability across different learning environments and systems**, it can help in the task of collecting under one valid identifier different experiences and outcomes that refer to one physical person by using different *identifiers*, *nicknames, social accounts,* etc. This property has to be managed in a way that even identityAlias are unique between users on the same broker(s).
- Differences between *anonymous* and *identified* Group(s):
    - **In xAPI there is a distinction between Anonymous and Identified Groups**, the first are intended to be Groups of Agents that are not formally united. Imagine a group of researchers that for a particular reason need to collaborate on a task, in this case the Group cannot have an identifier like an e-mail, social account etc, but only the list of participants; on the other hand, if the Group is, for instance, a permanent research Group or a working group in industrial scenarios, it could require a unique identifier. As NGSI-LD works on entities that have id(s) it is not possible to replicate this concept of anonymity, the use of identityAlias property can be restricted to an identified Group.
- SCORM concepts and how to apply to NGSI-LD:
    - SCORM and its SCO object fully focuses on the learner statistics and progress and less on the context and domain specific aspects of an immersive experience. This is why it is recommended to encapsulate the SCO or CMI object into an NGSI-LD entity related to a Learner type entity.
- Some guidelines and clarification on ETSI ISG ARF:
    - It can be possible to use as backend of an ARF instance an NGSI-LD broker, but it would need some features not available right now.
    - Authentication capabilities are recommended, there is no specific details on how to accomplish it.
    - The World Storage Function is dedicated to the manipulation of the World Grap, where is it possible to execute CRUD operations.
    - The World Storage Function makes simple calculations to give back relative poses between Trackables and Anchors to the World Analysis to position the AR asset accordingly to the user's position. More complex calculations are made by the World Analysis Function.

6.3 Proof of concepts

6.3.1 Introduction

Clause 6.3 will focus on proof of concepts for VR and AR for Smart Learning, which involves e-learning formats, NGSI-LD and AR/VR technologies for immersive experiences.

6.3.2 APP Tracking Software for LET

6.3.2.1 Overview

The chosen proof of concept is **APP,** a software that uses a standard format in the e-learning scene (e.g. xAPI, SCORM) to store and manage the user's achievements, progress and position of virtual activities. This approach is based on existing similar solutions [i.11].

APP is composed of different services that interact with the end user (**learner**), **LMS** and **NGSI-LD broker.**

The following image shows a high level diagram of the architecture.

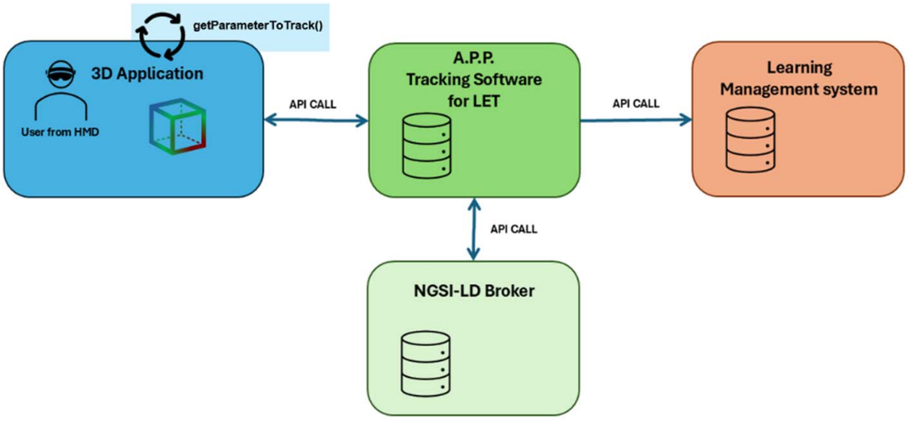


Figure 6.3.2.1-1: Overview of the APP architecture

Here is a breakdown of the core architecture in **Figure 6.3.2.1-1**:

- **3D Application,** a software developed for immersive experiences in the LET landscape, which makes API calls to the APP module.
- **LMS,** a system that expose/make API calls to/from APP, which manages the users and their e-learning experiences.
- **APP,** a software that acts as a bridge between the 3D Application and the LMS, which also translates e-learning data in NGSI-LD complaint data interacting with the NGSI-LD Broker.
- **NGSI-LD Broker,** which implements temporal API.

6.3.2.2 3D Application

Information on 3D Application is provided below:

- The learner **uses an HMD** (and a controller or hand tracking device) **to interact** with the digital world.
- The user **completes tasks** that ranges **from simple placement of object to complex manipulation** of the digital environment.
- The application uses a **standard format in the e-learning scene** (e.g. xAPI) to store and manage the **user's achievements and progress.**

- The application makes API calls to **APP** to **synchronize** and authorizes requests that are passed to the LMS of reference.
- The application defines a way to **send periodically the events and user's progress** to APP by API calls.
- The learner **has to be registered and logged in** at its target LMS.

6.3.2.3 LMS

Information on the LMS module is provided below:

- The LMS exposes a **set of endpoints** to store and potentially retrieve the e-learning data.
- The LMS makes available an **authorization mechanism to secure requests.**
- The LMS **is registered by an admin user to the APP** software that manages its learners.
- The LMS **can also include a front-end web app** that uses the data from APP to visualize it and more.

6.3.2.4 APP

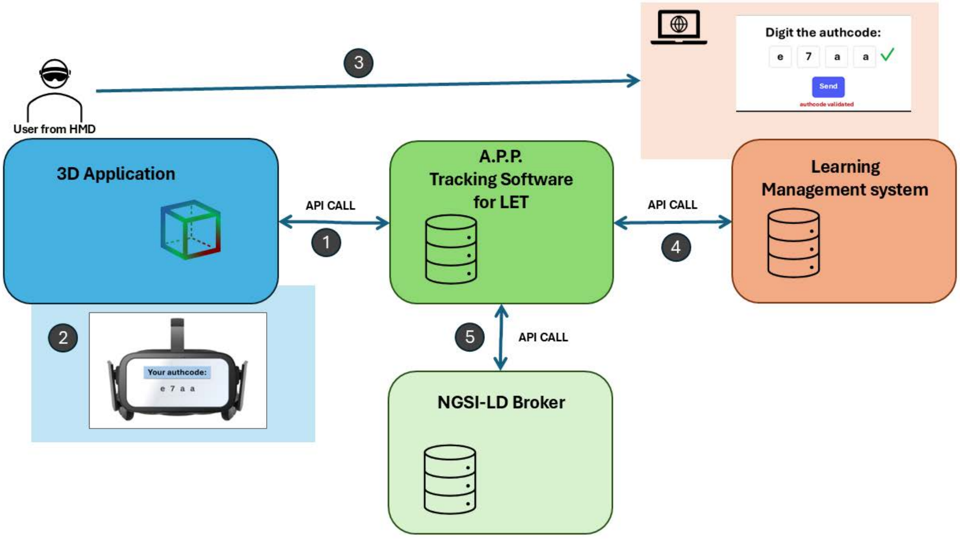
Information on the APP software is provided below:

- APP exposes a **set of endpoints, one for 3D Application and one for LMS modules.**
- APP makes available an **authorization mechanism to secure requests.**
- APP stores and **manages learners** based on the **API calls of the LMS modules.**
- **The backend of APP can be entirely based on NGSI-LD Broker** but it can also be a Database of choice in addition to NGSI-LD Broker.
- APP **takes well formed data based on the API definition and transform it in two separate packages**, one that is an **e-learning** complaint and the other that is a **NGSI-LD** complaint.
- The **well formed data can also be e-learning complaint data** (e.g. xAPI) ready to be translated to **NGSI-LD** by the translation APP routine.
- APP **makes the retrieval of NGSI-LD entities available** through API calls for the LMS module.
- APP c**an track the temporal evolution of a certain feature of an NGSI-LD entity** by exposing an endpoint, which relies on temporal API of NGSI-LD, to the LMS module so that it can use the data.

6.3.2.5 Data and operations workflows

Clause 6.3.2.5 will discuss workflows in high level detail, which will be presented as diagrams with descriptions on every step.

Learner-3D Application pairing

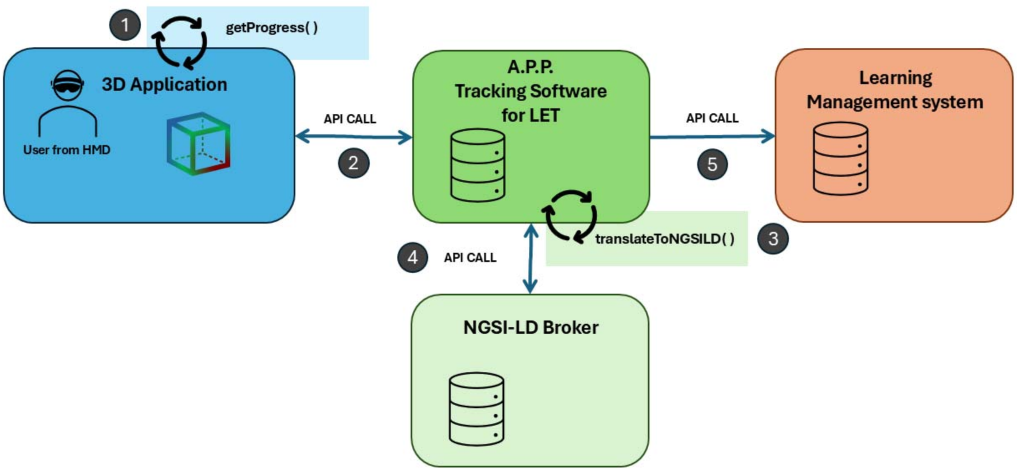


Here is a breakdown of the diagram:

1) The 3D Application makes an API Call to APP software to initialize a user session, the APP returns an authcode related to the 3D Application identifier.
2) The 3D Application is responsible of visualizing the authcode to the user.
3) The user authenticates to the LMS web site and digits the authcode into an input area.
4) The LMS module makes an API Call to APP software to injects the user information to be bound to the corresponded 3D Application identifier (related to the authcode).
5) The APP software makes an API Call to the NGSI-LD broker/backend to create the corresponding entities.

Figure 6.3.2.5-1: Learner-3D Application pairing diagram

Progress translation and storage



Here is a breakdown of the diagram:

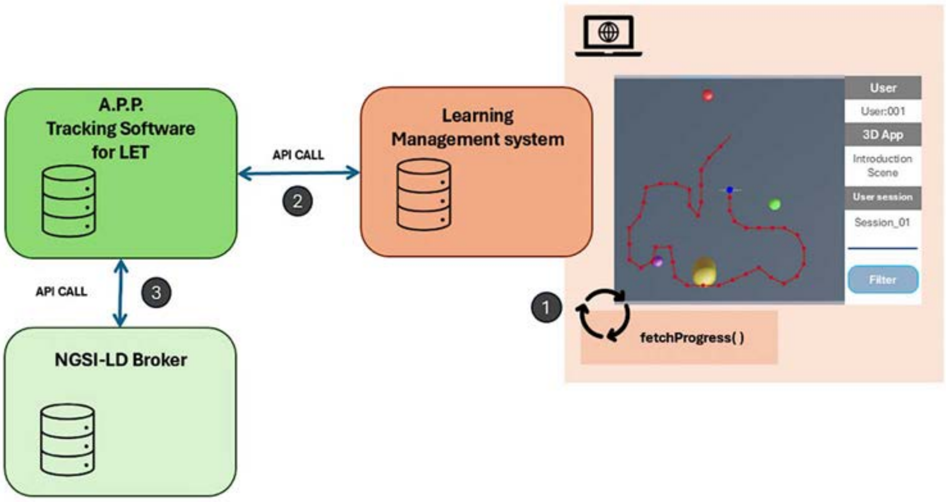
1) The 3D Application executes a temporized routine to collect progress to send to APP.
2) The 3D Application makes an API Call to APP software to send the collected progress to translate and store.
3) The APP software executes the translation routine that takes as input the data from the 3D Application and a e-learning standard format and return as the output the translated version in NGSI-LD.
4) The APP software makes an API Call to the NGSI-LD broker/backend to update the corresponding properties.
5) The APP software makes an API Call to the LMS store the corresponding data in e-learning format.

Figure 6.3.2.5-2: Progress translation and storage

- The 3D Application is responsible for the rate of progress translation on the APP software. The developer of such applications decides what achievements the user is expected to accomplish.

- In step 4 the 3D Application can send the data already in e-learning format, making the process of translation for the APP software limited to NGSI-LD, otherwise, the data can be formatted in a well-known structure so that the APP software will be responsible of translating it in both e-learning format and NGSI-LD.

Tracking position of an entity or other Properties



Here is a breakdown of the diagram:

1) The web application hosted by the LMS makes a request to the LMS itself to fetch progress (such as positions or other properties) so that it can visualize the data.
2) The LMS makes an API call to the APP software to retrieve progress data.
3) The APP software makes an API call to the NGSI-LD broker using the temporal query language to have the temporal evolution of an entity and its properties.

Figure 6.3.2.5-3: Tracking position of an entity or other Properties diagram

- In this use case is the LMS and an affiliated web application to make use of the temporal evolution of entities, but there are other use cases where this data could be useful to be available even in the 3D Application or other software (e.g. AI chat, analytics software).

7 Improvements to NGSI-LD

7.1 Introducing 3DProperty type for properties

7.1.1 Overview

The need of a 3D based Cartesian coordinate system has increased during the work on the present document. Here is a list of reasons that makes this proposal more convenient in terms of AR/VR immersive experiences:

- Game engines like Unity, Unreal Engine, Godot and even JavaScript Frameworks like ThreeJS use a Cartesian coordinate system.

- ISG ARF, shown in clause 4.2.1.4, uses the Cartesian coordinates system and performs calculations on Cartesian coordinates.

- AR/VR immersive experiences are developed using the exposed technologies.

- NGSI-LD lacks functionalities to represent, in a similar way as for GeoJSON data, Cartesian coordinates.

It turns out that the OGC 06-103R4 document [i.14] represents a *good starting point* to define this **new Property Type.**

As of IETF RFC 7946 [i.17], GeoJSON supports: "Point", "MultiPoint", "LineString", "MultiLineString", "Polygon", "MultiPolygon", **except** "GeometryCollection".

All these types should be supported by **3DProperty,** and the functions should be supported as well**.** It is worth noting that this new Property type will basically allow support for positions and other Geometrical entities on Local SRS; a Local SRS is a Cartesian coordinate system which is not referenced to the earth's surface, that is exactly the type of coordinate system needed by virtual, immersive experiences.

7.1.2 Data Model

A data model for 3DProperty/CartesianProperty is proposed based on the current GeoProperty framework, PostGIS, and the OGC 06-103R4 standard. Similarly with GeoProperty the following fields should be defined for the 3DProperty/CartesianProperty:

- **type** to determine the Geometry type of the property
- **coordinates (name to change)** to define the actual Geometry

The property will also have a non mandatory **CRSOrigin** Relationship (Relationship of Property), which points to an entity that provides a GeoProperty that serves to a relative position the 3DProperty in a Augmented Reality environment, where virtual objects merge into the real world.

Here is an example of a RoboticArm that will be displayed during an AR experience for training:

```
{
  "id": "urn:ngsi-ld:RoboticArm:001",
  "type": "RoboticArm",
  "position": {
    "type": "3DProperty",
    "value": {
"type": "CartesianPoint",
"coordinates" : [1.2, 0.5, 0.8]
}
"CRSOrigin": {
"type": "Relationship",
"object" : "urn:ngsi-ld:Point:001"
}
  }
}
```

- The RoboticArm has its own position in the Cartesian reference system which is relative to the CRSOrigin that points to an entity that provides a GeoProperty, so the actual position of the object will be interpreted (by a game engine or 3D software) as 1,2 units along the X axis, 0,5 units along the Y axis and 0,8 units along the Z axis from the position defined by the GeoProperty provided by CRSOrigin.

```
{
  "id": "urn:ngsi-ld:TrainingRoom:001",
  "type": "TrainingRoom",
  "position": {
    "type": "3DProperty",
    "value": {
"type": "CartesianPolygon",
"coordinates" : [
[[100.0, 0.0, 20],
[101.0, 0.0, 20],
[101.0, 1.0, 10],
[100.0, 1.0, 10],
[100.0, 0.0, 10]] ]
}
  }
}
```

The proposed Geometry type(s) are:

1) **CartesianPoint:** Defines a 3-dimensional point in a Cartesian coordinate system.

2) **CartesianLineString:** Defines paths or edges in 3D, useful for representing the movement of objects or boundaries in training environments.

3) **CartesianPolygon:** Supports surfaces and 2D boundaries in 3D space**.**

For future investigations:

4) **CartesianPolyhedron:** Defines volumetric shapes (e.g. rooms, equipment) using a closed set of faces, offering more accurate representations for applications like AR training simulations.

5) **Paths:** A CartesianLineString could be used to define a robot's movement path or a worker's route through a facility.

6) **Zones:** A CartesianPolyhedron could represent interactive zones within a training room, allowing AR interactions within specific 3D volumes.

7.1.3 Query Language

From **OGC 06-103R4** [i.14], query functions on the Geometry type are defined as follows (these definitions are "quoted" from [i.3] and exceptionally contain the modal verb "shall" which are normally forbidden in ETSI Group Reports like the present document):

- **equals:** Returns true if the two geometries are spatially equal (i.e. have identical coordinate points and topology).
- **disjoint:** Returns true if the two geometries do not share any points in common.
- **intersects:** Returns true if the two geometries share any point in common.
- **touches:** Returns true if the two geometries touch at the boundary but do not overlap in their interiors.
- **crosses:** Returns true if the two geometries intersect in such a way that they cross over one another.
- **within:** Returns true if the geometry is completely within another geometry.
- **contains:** Returns true if one geometry completely contains another.
- **overlaps:** Returns true if the geometries share some but not all points in common (i.e. they partially overlap).
- **relate:** Returns true if the relationship between the two geometries matches the specified DE-9IM matrix (a matrix that describes spatial relations based on interiors, boundaries, and exteriors).

From **OGC 06-103R4** [i.14]**,** analysis functions on Geometry type are defined as:

- **distance:** Computes the shortest distance between two geometries.
- **buffer:** Creates a buffer region around the geometry at a specified distance, returning a new geometry that represents this buffer area.
- **convexHull:** Returns the smallest convex geometry that encloses the original geometry (often visualized as a convex polygon around a set of points).
- **intersection:** Computes the intersection of two geometries, returning a geometry that represents the overlapping area.
- **union:** Computes the union of two geometries, combining them into a single geometry that includes all points from both.
- **difference:** Returns the part of the geometry that does not intersect with another geometry (essentially a "subtraction").
- **symDifference:** Returns the symmetric difference of two geometries, which consists of the parts of each geometry that do not intersect with each other.

This is a comprehensive review of functions defined by OGC and others implemented by POSTGIS; It is worth noting that some of the derivative calculations could impact the performance of NGSI-LD brokers.

For comparison, here is a list **of functions supported by the NGSI-LD Geoquery Language** on the **GeoProperty** type:

- **near** statement (production rule named nearRel):
  - **maxDistance** modifier. For an entity to match it has to be within the buffer geometric object (as defined by [i.14]) given by the reference geometry, with distance (in meters) equal to the number conveyed (production rule named PositiveNumber).
  - **minDistance** modifier. For an entity to match it has to be disjoint with the buffer geometric object (as defined by [i.14]) given by the reference geometry, with distance (in meters) equal to the number conveyed (production rule named PositiveNumber).
- **equals** relationship (production rule named equalsRel). For an entity to match, the target geometry "shall" be equal to the reference geometry.
- **equals** relationship (production rule named equalsRel). For an entity to match, the target geometry "shall" be equal to the reference geometry.
- **disjoint** relationship (production rule named disjointRel). For an entity to match, the target geometry "shall" be disjoint, to the reference geometry.
- **intersects** relationship (production rule named intersectsRel). For an entity to match, the target geometry "shall" intersect, with the reference geometry.
- **within** relationship (production rule named withinRel). For an entity to match, the target geometry "shall" be within, the reference geometry.
- **contains** relationship (production rule named containsRel). For an entity to match, the target geometry "shall" contain, the reference geometry.
- **overlaps** relationship (production rule named overlapsRel). For an entity to match, the target geometry "shall" overlap, the reference geometry.

It is also worth noting that, from a search on the web, some databases have implementation and support for a 3D Geometry type of data.

7.2 Collisions between user @context and core @context

7.2.1 Overview

During the development of the present document and the mapping of e-learning format to NGSI-LD, an issue emerged concerning the collision of terms between user-generated @context definitions and the NGSI-LD core @context, for example the terms "`max`" and "`min`" used as value for "`aggrMethods`, they could not be used to name properties. It would be suggested to implement at least one of these strategies:

- For POST Queries the following change can be made:
  - The **q** parameter to construct a query could be a **JSONProperty**, so that the query parameters used inside it will not be expanded. This could reduce the amount of terms that collide with other keywords defined in **NGSI-LD core @context.**
- Use longer names for core @context names:
  - This could impact the GET HTTP requests as longer names makes URL with queries longer and less readable.

7.3 Introducing Temporal Interval Query

7.3.1 Overview

During the development of the first proof of concept, a limitation was identified in handling queries related to **"when something occurred"**. Specifically, determining the time intervals during which a specific condition or state was true. This limitation can at times reduce the ability to analyse temporal patterns and perform detailed event-driven queries, which are crucial for dynamic and time-sensitive applications used in immersive experiences for Smart Industries. This new feature enables the extraction of datetime intervals during which entities exhibit specific characteristics.

7.3.2 Scenario and objective

There is the scenario to showcase the proposed improvements:

- A **smart factory** wants to **train personnel**. The training session consists in **VR and AR activities** by using **HMD devices and mobile devices** instead of machines used in the production process. The training is all about **sensors and systems** controlling pipes and **machines temperature** during a production line in a virtual and/or mixed environment. The **user has control** over the **temperature** and **any error** can change the state of machines drastically.

The objective:

- The **goal** is to **identify specific datetime intervals** during which the target's **temperature exceeds** a predefined **threshold**, indicating **potential issues** in the production line that require actions.

7.3.3 Proposed "interval" aggregation function

It is supposed to fetch the temporal evolution of an entity of type "Machine" and its "temperature" property, the HTTP will look like:

- `GET /ngsi-ld/v1/temporal/entities/urn:ngsi-ld:Machine01?timeproperty=observedAt&timerel=after&timeAt=2023-11-04T11:39:00.00Z&attrs=temperature`

For now it is not possible to answer the question "**when was** the "*temperature*" **equal** to 62° C?" directly to the broker. The user interacting with the broker could, in order to have an answer:

1) Make a **HTTP GET** request like:

```
GET /ngsi-ld/v1/temporal/entities/urn:ngsi-ld:Machine01?
timeproperty=observedAt&timerel=after&timeAt=2023-11-
04T11:39:00.00Z&attrs=temperature
```

2) **Search** and **aggregate** time intervals that **satisfy the condition** over the "*temperature*" property (e.g. temperature == 62).

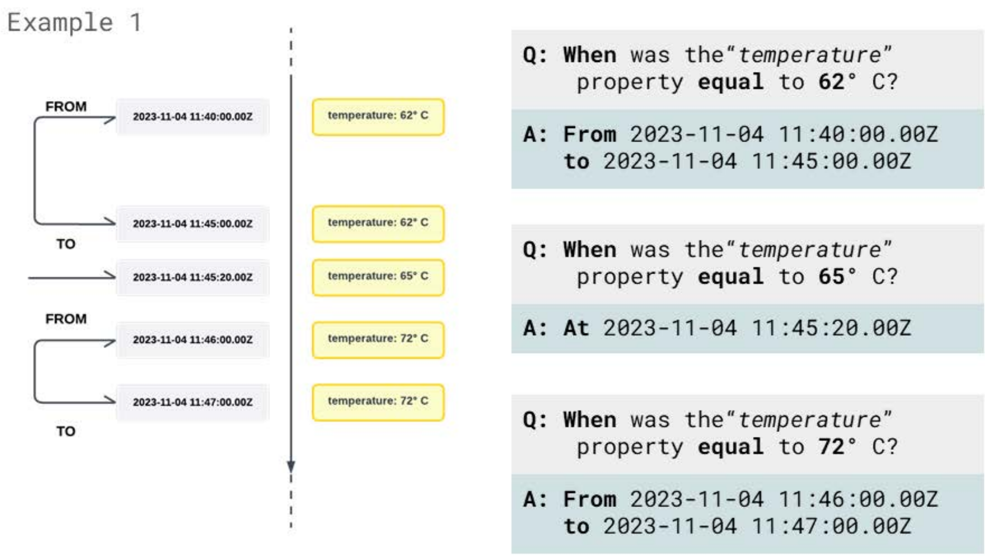


Figure 7.3.3-1: Example 1 of intervals during which condition is satisfied, single moment

Figure 7.3.3-1 shows a first example of questions and answers while inspecting the temporal evolution of a certain property of an entity. This example shows how intervals can also be a single moment (timestamps) in the timeline of an entity lifecycle.

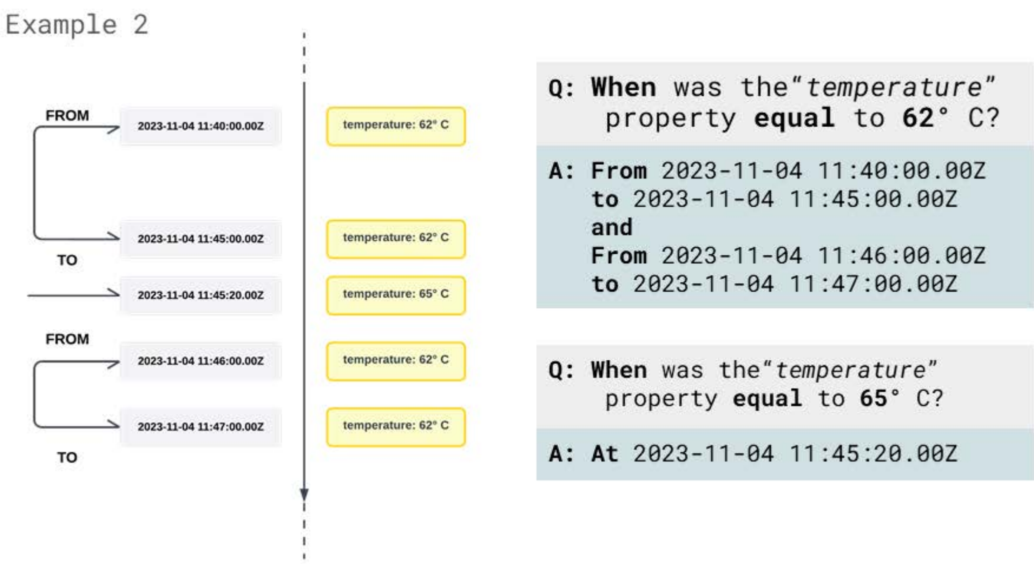


Figure 7.3.3-2: Example 2 of intervals during which condition is satisfied, unified intervals

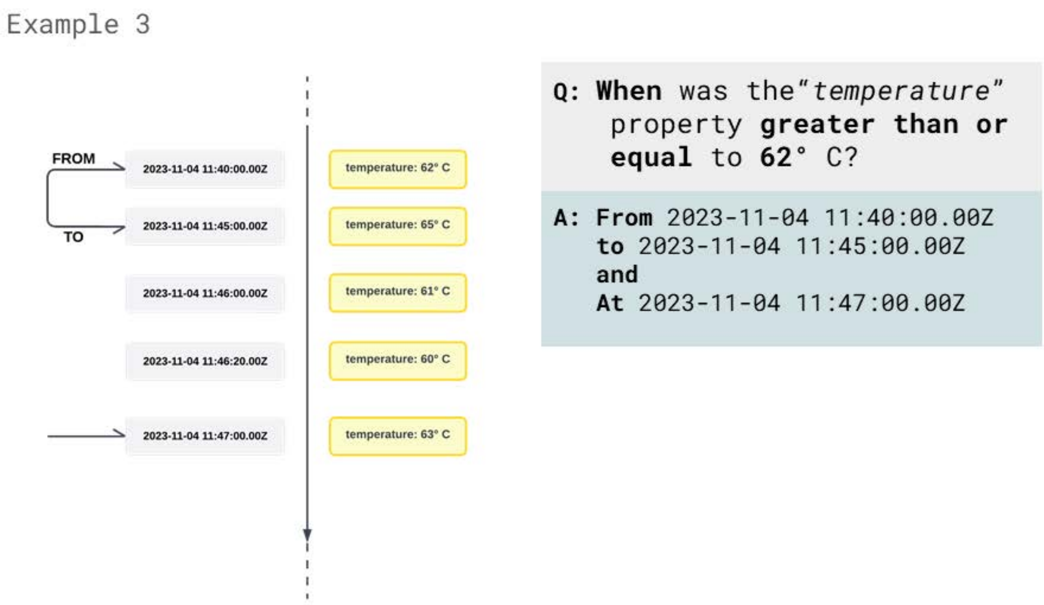


Figure 7.3.3-3: Example 3 of intervals during which condition is satisfied, more unified intervals

Figure 7.3.3-2 and Figure 7.3.3-3 show examples of how intervals can also be aggregated and unified based on the value of the property tested during the timeline of an entity lifecycle. The proposed change is the introduction of the "**interval**" aggregation function.

The QUERY EXAMPLE 1 shows the proposed way to fetch the time intervals during which the value of temperature property of Machine type entity was greater than or equal to 62 using the proposed aggregation function "**interval**".

QUERY EXAMPLE 1:

```
GET /ngsi-ld/v1/temporal/entities?type=Machine&timerel=after&timeAt=2020-04-
04T11:38:00.000000Z&aggrMethods=interval&options=aggregatedValues&
q=temperature>=62
```

Annex A:
Change history

Date	Version	Information about changes
2024/04	0.0.1	Setting up document structure
2024/04	0.0.1	Adding preliminary content
2024/05	0.0.1	Early draft of the document
2024/06	0.0.2	Start of the work for the stable draft of the document
2024/07	0.0.2	Stable draft of the document
2024/07	0.0.2	End of the work for the stable draft of the document
2024/08	0.0.3	Start of the work for the final draft of the document
2024/09	0.0.4	Work for the final draft of the document
2024/10	0.0.4	Work for the final draft of the document
2024/10	0.0.4	Final draft of the document
2024/11	0.0.4	End of the work for the final draft of the Document
2024/11	0.0.4	Final draft of the Document
2024/12	1.1.1	Technical Officer Review after TB approval for EditHelp publication pre-processing

History

Document history		
V1.1.1	January 2025	Publication