Digital Twin Interoperability for Space Exploration Missions Using the Spatial Web

Alicia Sanjurjo Barrio

JPL Group 345C: Advanced Electronics Systems & Technology

Jet Propulsion Laboratory
California Institute of Technology

JVSRP Final Presentation, April 24, 2025

Mentors: **Dr. Thomas Lu** & **Dr. Edward Chow**

Content

About Me

Alicia Sanjurjo Barrio

From: Madrid, Spain

Academic Background:

- Bachelor in Aerospace Engineering
- Master of Space Studies (MSS24) at ISU

Professional Experience:

- Reinforcement Learning for UAVs, GMV (Madrid, Spain)
- Mobility and Control for Lunar Robots, SRL (Sendai, Japan)

JPL Group: 345C – Advanced Electronics Systems & Technology

Focus: Master's Thesis Research on Digital Twin Interoperability Using the Spatial Web Protocols

Interests: Photography, reading, painting, hiking, travelling

Background of the Project

1. Why now?

WE ARE GOING BACK TO THE MOON!

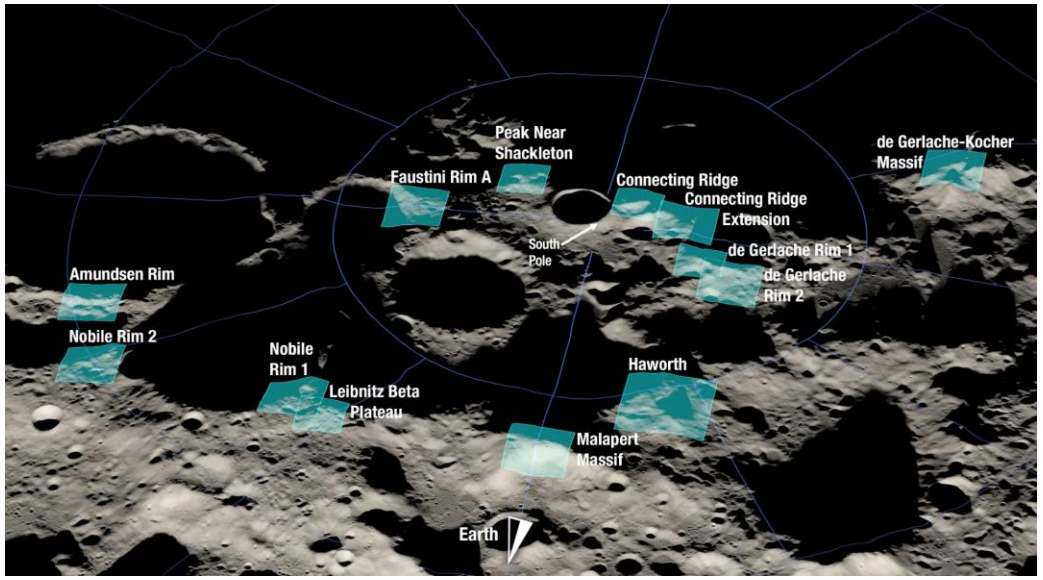
The Moon is the next frontier of space exploration, serving as a testing ground for the first extraterrestrial settlements and a crucial stepping stone for future Mars missions.

Rise of the Private Space Sector

↑ # Players

Limited space on the South Pole of the Moon

Collaboration & Interoperability Needed



NASA's 13 candidate landing regions for Artemis III mission.

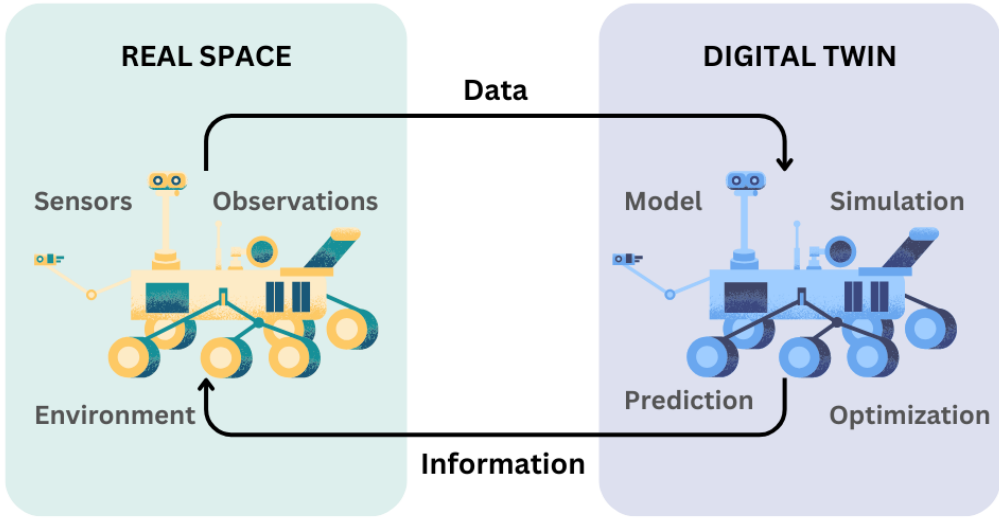
Background of the Project

2. Why Digital Twins?

Rise of Digital Twins across all industries, including aerospace, automotive, manufacturing, healthcare, etc.

Space Industry Interest: Essential for mission planning, rover operations, and system simulations.

The Need for Interoperability: DTs are generally developed in isolation and are very domain-specific.



Relationship between Real Space and Digital Twins

Lockheed Martin

SpaceX **NASA** **Boeing**

Space Force

Some Space Organizations using Digital Twins

Background of the Project

3. Why use the Spatial Web standard?

Interoperable, Safe, and Open for All

The **Spatial Web**, also known as **Web 3.0**, is a computing environment that will blur the lines between the physical and digital worlds

It is an **evolution of the internet** that will allow people to interact with places, things, and each other in a 3D virtual space.

It will allow users to **place information spatially and contextually** on objects and locations, and interact with them in natural ways.

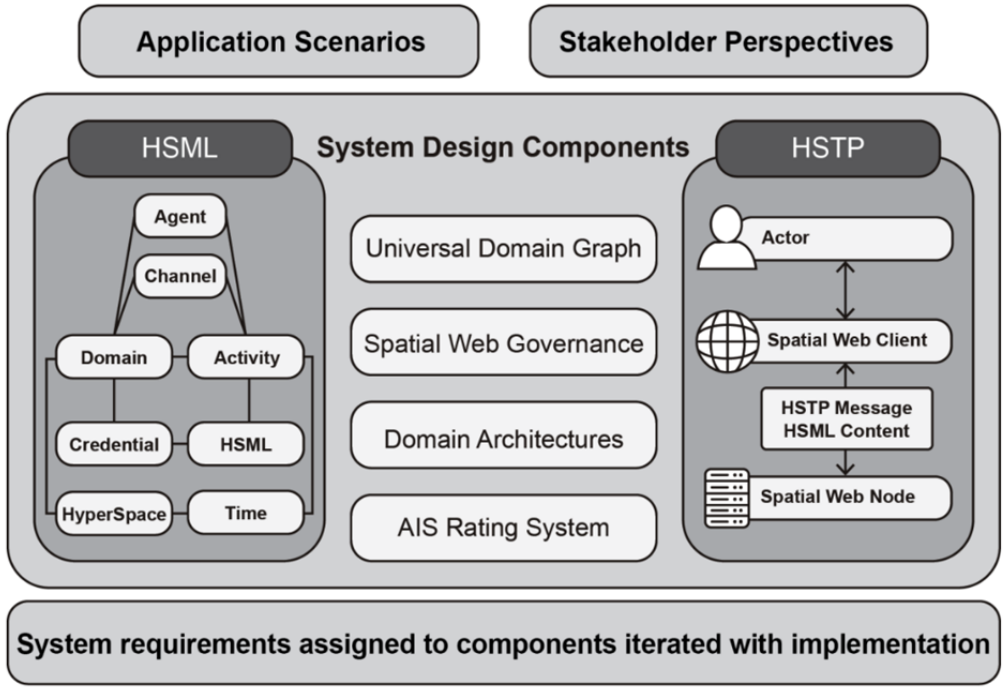


Figure 1—System architecting process overview

From 979-8-3503-5597-0/25/$31.00 ©2025 IEEE,
Enabling Interoperable Digital Twins for Collaborative Lunar Exploration

Goal and Objectives

Project Long Term Goal: Develop the **Virtual Environment for Collaborative Lunar Explorations (VIRCLE)**, a platform to exchange data, models, tools, and processes, ultimately creating a marketplace for Digital Twins products and services.

Connect Digital Twins across different platforms	Achieve real-time collaboration	Create an open-source platform/server
Expand the Spatial Web Foundation's work	Develop and implement HSML schema	Automatically generate exchangeable files
Ensure Data Trust following Spatial Web protocols	Business Logic conversion utilizing LLMs	Accept all kinds of data and connect all sorts of DTs

Problem Statement

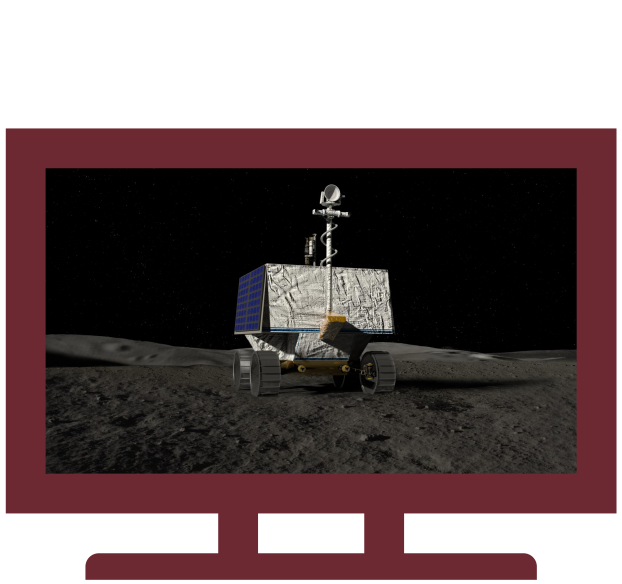
My Research Topic: **Digital Twins Interoperability for Space Exploration**

From **Literature Review** conducted for my Thesis, Digital Twins face **3 main challenges** that constraint collaboration for future lunar missions:

- **Standardization:** There are no common communication protocols, data formats, and regulations regarding DTs.
- **Interoperability:** DTs are developed using different platforms (Unity, Unreal, Omniverse), model formats and physics engines.
- **Data Trust:** Space missions generate large amounts of critical and sometimes sensitive data. Cybersecurity for DTs is a key concern. Security & validation across different DT systems remain unsolved.

Thesis Scope

Implement the Spatial Web Protocols to develop the Architecture necessary to achieve Interoperability among Digital Twins for Space Exploration Missions



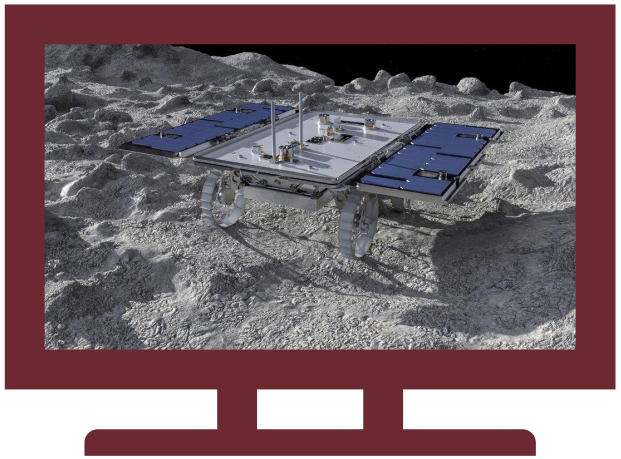
Unity

Spatial Web Protocols

HSML – Language

HSTP – Transactions

Standardization Interoperability Data Trust



Omniverse

HSML in Simple Words

HSML ≡ Hyperspace Modeling Language

- A **modeling language** that can be read by both humas and machines
- A **detailed schema/map** that explains how all the different data is related to each other in the system
- It describes objects, relations, actions, activities and their permissions

HSML does for the Spatial Web as HTML does for the WWW

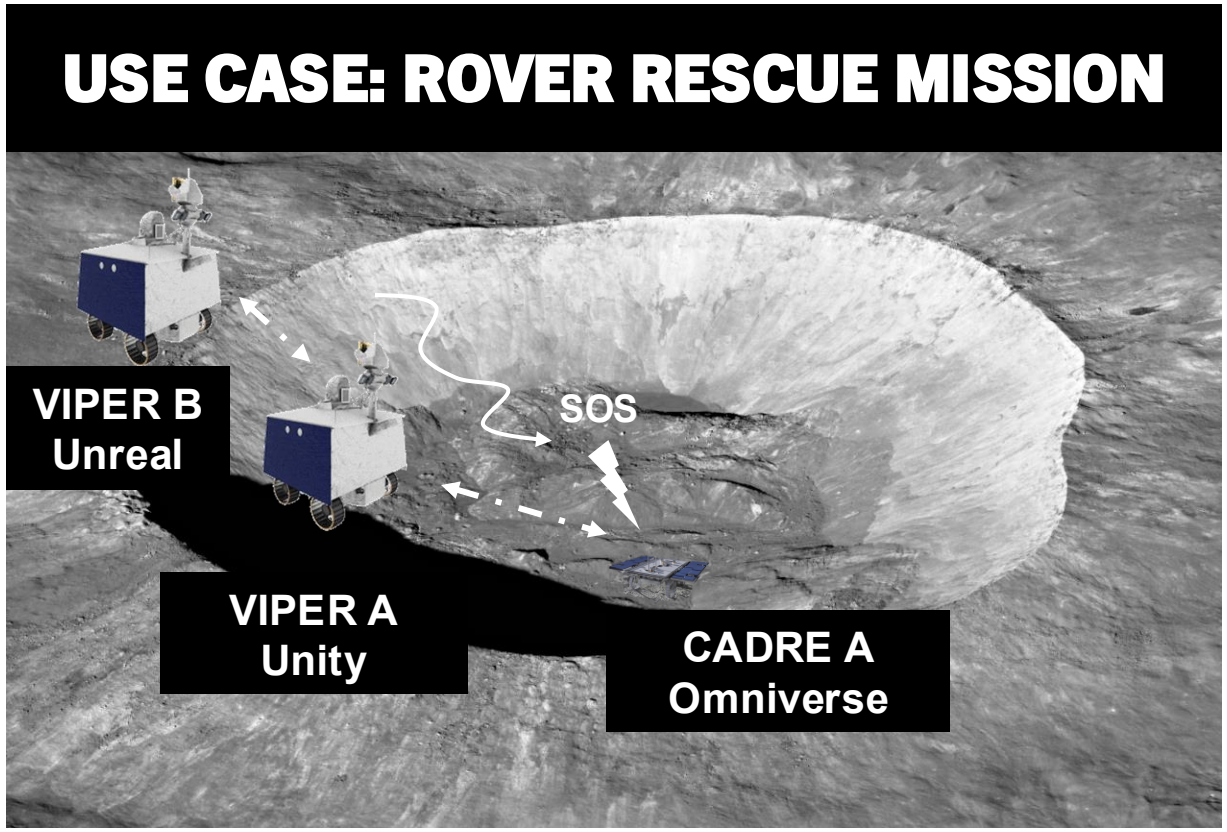
HSTP in Simple Words

HSTP ≡ Hyperspace Transaction Protocol

- System designed to enable smooth, decentralized, secure, and private communication
- Focuses on transactions (exchanges of information or actions)
- The **HSTP Objects** are "units of activity". Each object represents an action or event (called an **HSML Activity**)
- Allows different **Spatial Web nodes** to exchange messages and stay coordinated.

HSTP does for the Spatial Web as HTTP does for the WWW

Use Case Demonstration

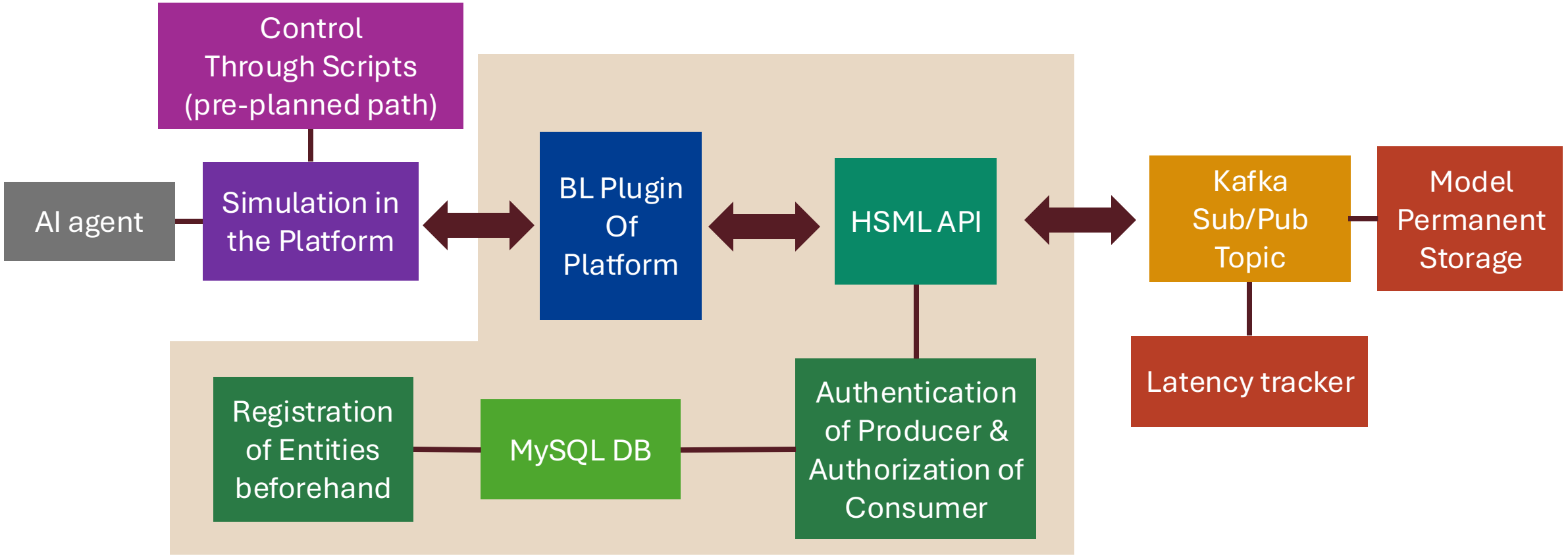


Simulation experiment to be carried out to evaluate the feasibility of the Spatial Web standard in achieving DT interoperability:

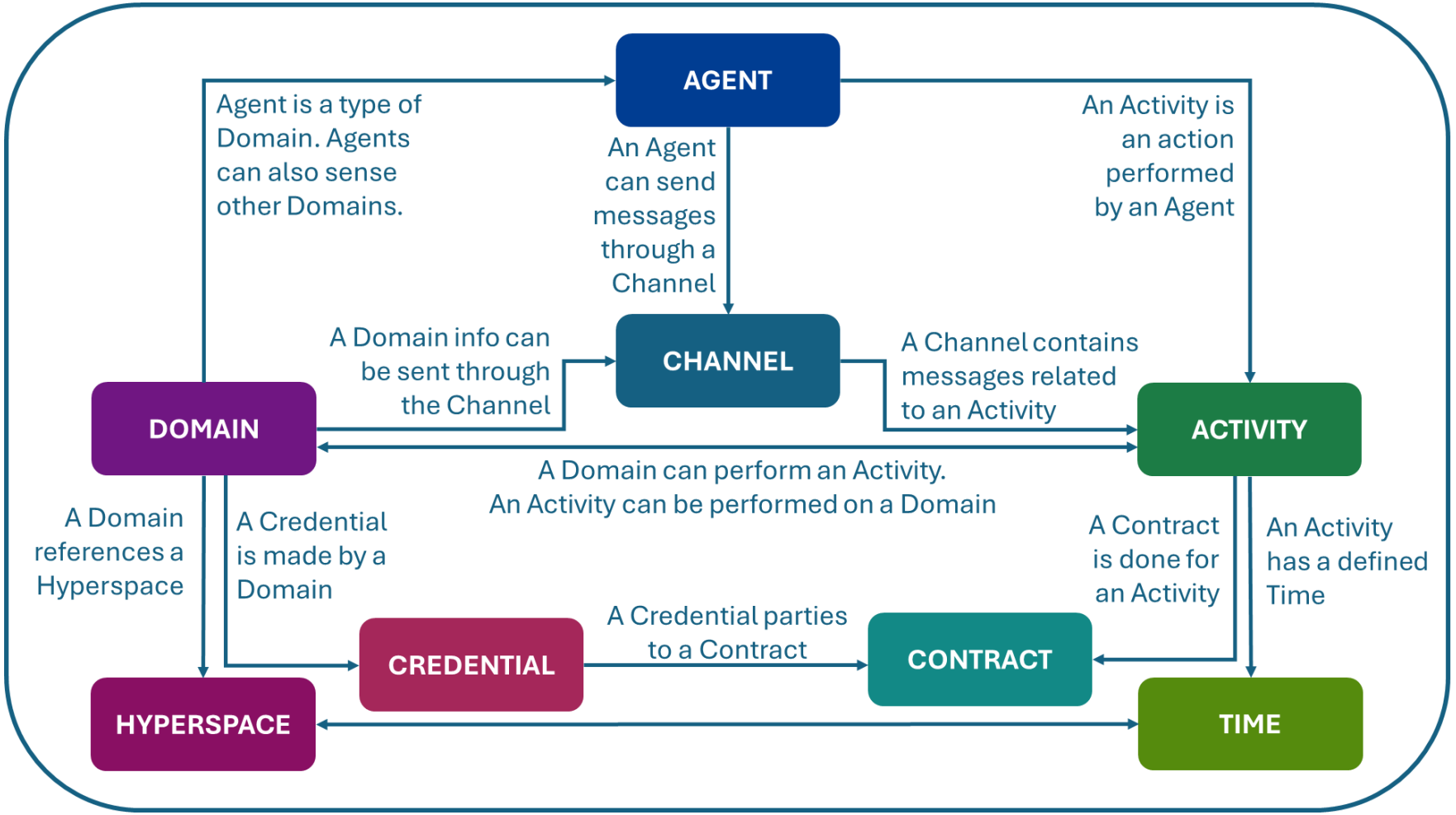
- Simulating **multi-agent robotic systems** in a lunar environment
- Testing **interoperability, data trust, and automation** across platforms.
- Demonstrating collaboration possibilities on the Moon, such as **rover assistance scenarios** using Digital Twins

Methodology – Demo Implementation

Prove Interaction is possible across Platforms using HSML, with minimal latency, and security protocols.
Script-controlled rovers follow a pre-planned path, with some kind of automation.



Methodology – HSML Schema

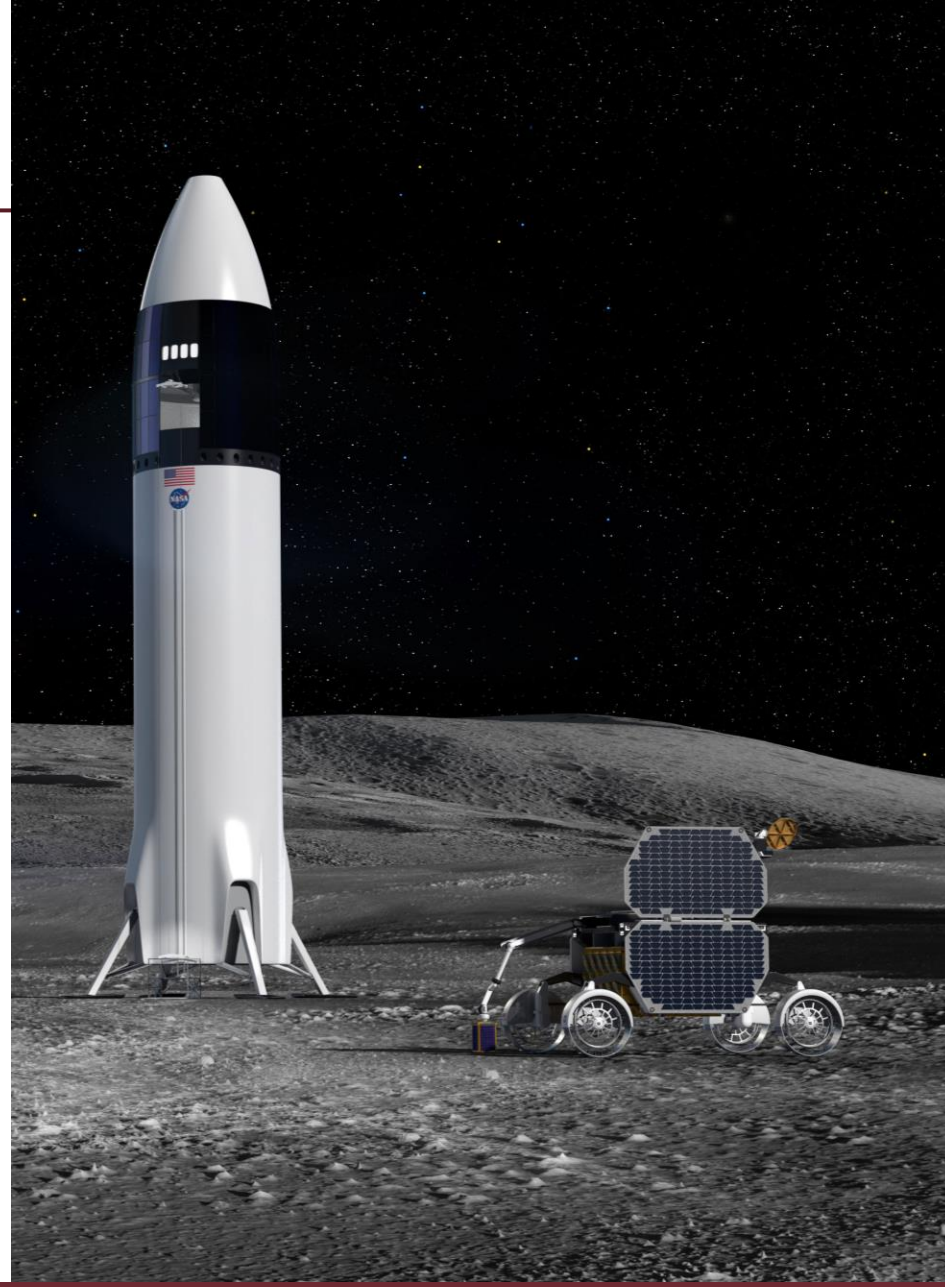


HSML Ontology Relationships according to the Spatial Web

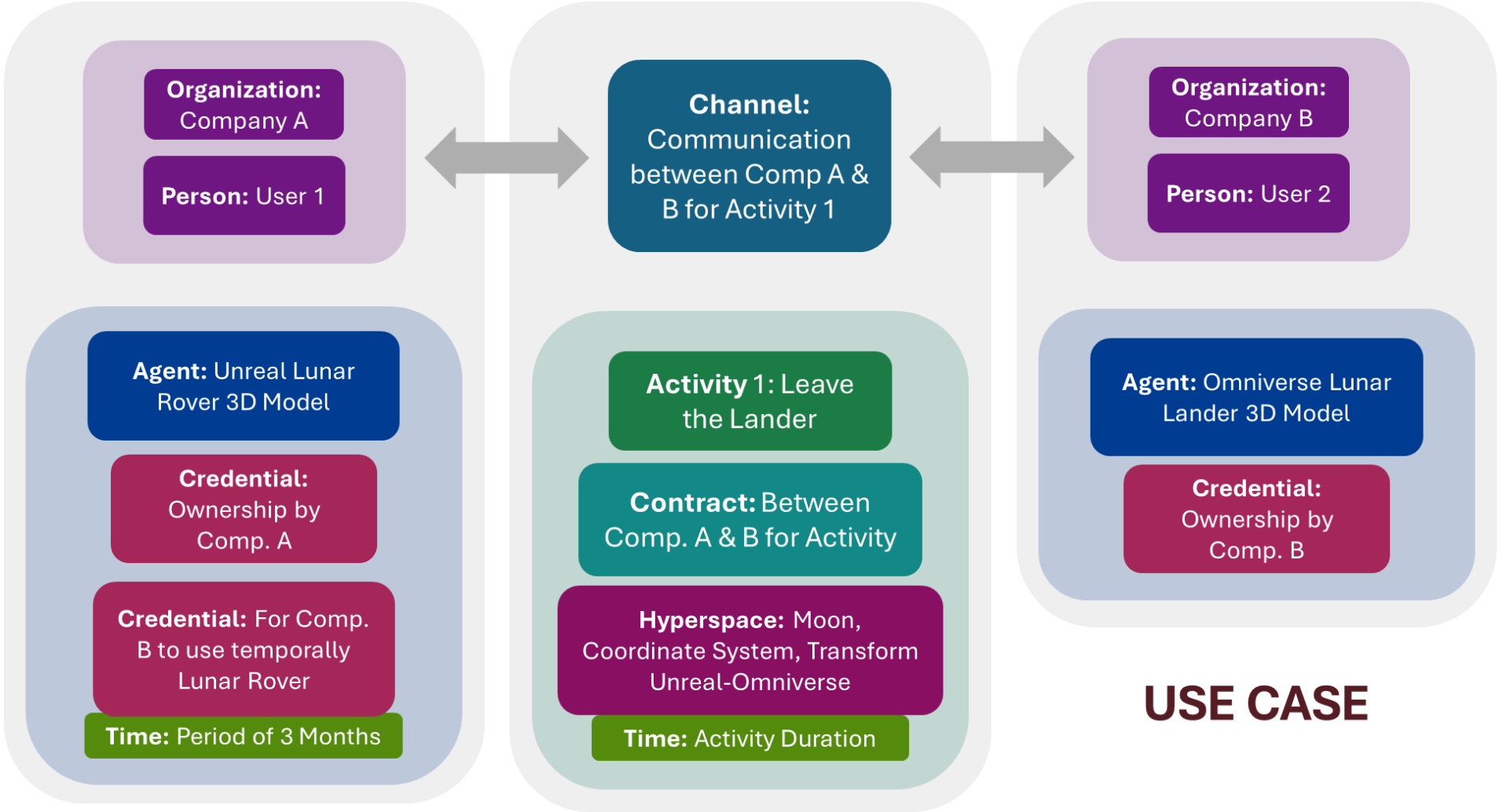
Methodology – HSML Schema

POSSIBLE USE CASE:

- Two **Organizations**, Company A and Company B, sign a **Contract** to work together on a space mission to the Moon.
- The Rover of Comp. A will be carried inside the Lander of Comp. B. (**Agents**)
- Once they reach the Moon, the Rover will leave the Lander. To make sure this **Activity** is developed without any problem, Comp. B needs to work with the model of the Lunar Rover and run some simulations.
- A **Credential** is issued by Comp. A, authorizing the Agent of Comp. B to have access to the 3D Model of their Rover Agent for a Period of 3 Months (**Time**).
- The model of the Lunar Rover is in Unity, and the model of the Lander is in Omniverse; both have a Moon surface environment modelled. The **Hyperspace** must consider, the location (the Moon), and the appropriate coordinate systems and transforms needed to go from one platform to the other.
- All the messages and communications related to this Activity are carried out in a **Channel**.



Methodology – HSML Schema



Methodology – HSML Schema

Created the schema with the **HSML ontology types,** using existing **properties from Schema.org** plus our own **custom properties** for each type.

HSML Documentation

Entity

An HSML Class

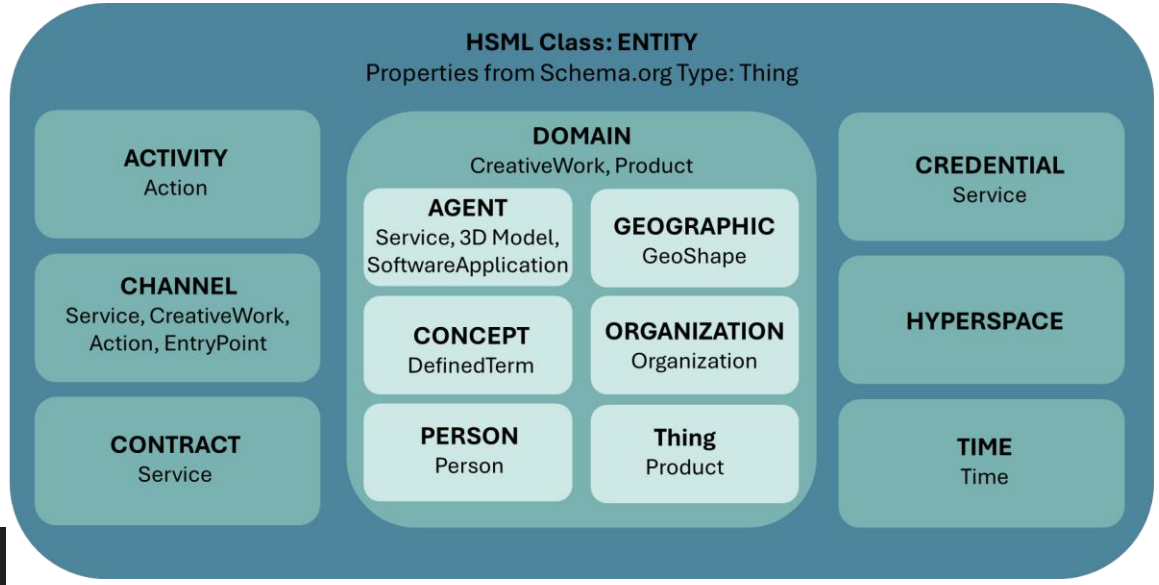
Entity

The base item in the Spatial Web Ontology

Property	Expected Type	Description
Properties from **Thing**		
additionalType	Text or URL	An additional type for the item, typically used for adding more specific types from external vocabularies in microdata syntax. This is a relationship between something and a class that the thing is in. Typically the value is a URI-identified RDF class, and in this case corresponds to the use of rdf:type in RDF. Text values can be used sparingly, for cases where useful information can be added without their being an appropriate schema to reference. In the case of text values, the class label should follow the schema.org style guide.
alternateName	Text	An alias for the item.
description	Text or TextObject	A description of the item.

Entity Example

```
{
  "@context": "https://digital-twin-interoperability.github.io/hsml-schema-context/hsml.jsonld",
  "@type": "Entity",
  "name": "Example Entity",
  "description": "This is an example entity type.",
  "swid": "did:key:6MkvWxmpULqn65HWJ7id7GF2G8CGWeGs9GvftPw3ZPESuLC"
}
```



HSML Ontology Classes Hierarchy & corresponding Schema.org types they draw properties from

Methodology – Verification in the Spatial Web

Created Unique Identifier Method for Safety and Trustworthiness

The **Identifier for the HSML schema**, according to the Spatial Web standard, is the SWID. The **Spatial Web Identifier (SWID)** is a Decentralized Identity (DID) that conforms to W3C did-core, https://www.w3.org/TR/did-core/.

Figure 1 A simple example of a decentralized identifier (DID)

NO EXISTING SWID METHOD

USE AN EXISTING DID METHOD

```
"swid": "did:key:6MktsGJxcNwZaC1vuimSYai7Zs9ykPwwrAfeVQtMLDU3nqQ"
```

Chosen method: **did:key**

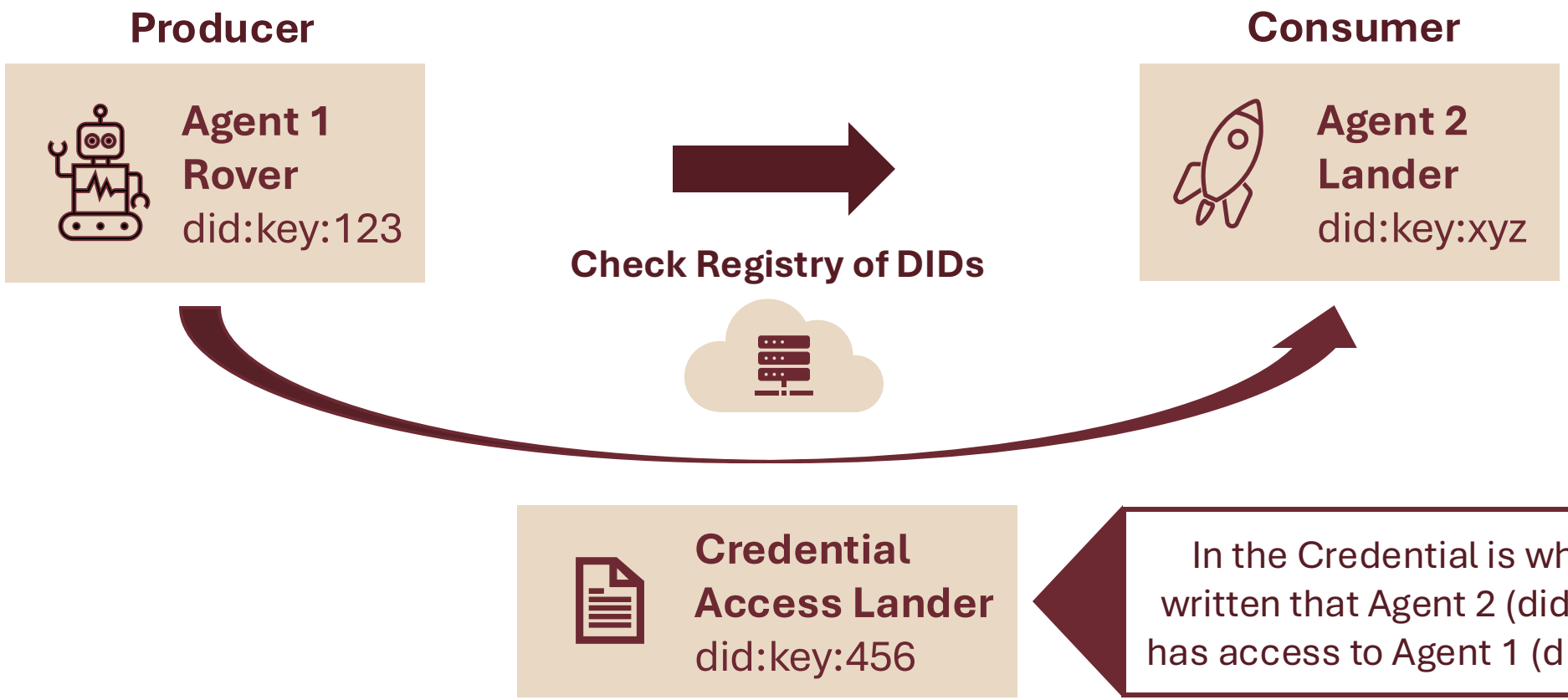
Methodology – Verification in the Spatial Web

VERIFICATION LOGIC – Registration, Authentication and Authorization

1. An **Entity is registered**, a DID key pair is generated for it. These Entity's **public key** gets stored in a database together with its important metadata (name, description, creator…), and the user stores the **private key** on their own.
2. If the Entity is an **Agent**, during the registration an **associated Kafka topic is created**. This topic is where the Agent (rover, lander) can act as a producer.
3. **On the producer side**, before Agent can start publishing, program asks for **private key of Agent**, which is **checked** against public key in the database (to protect against spoofing).
4. If private key is correct, **message is sent** through server's topic.
5. **On the consumer side**, to have access to Agent's topic, user is asked for their private key. In database, it is **checked if** associated User's public key is **part of a Credential** that authorizes access to that Agent.
6. If User has Credential, they can start consuming the data.

Methodology – Verification in the Spatial Web

VERIFICATION LOGIC



Methodology – Verification in the Spatial Web

Registration using MySQL Database

The Registration Tool allows users to register new Entities by generating a unique SWID (using the DID:key method). It stores the Entity's metadata (HSML JSON) and public key in a MySQL database, and returns the private key and updated JSON file to the user.

Structure of MySQL Registry Table

Field	Type	Description
id	Int	Row number automatically generated.
did	Varchar(255)	The **SWID of the Entity** being registered
registered_by	Varchar(255)	The SWID of the Person/Organization that registered the Entity
public_key	Text	The public key corresponding to the DID.
metadata	JSON	**HSML-encoded JSON metadata** associated with the Entity.
created_at	Timestamp	Timestamp of when the record was created.
kafka_topic	Varchar(255)	**Kafka topic** associated with an Agent for pub/sub messaging.
allowed_did	Text	List of **SWIDs allowed** to interact with this Entity (based on Credentials).

Methodology – Verification in the Spatial Web

AUTHENTICATION & AUTHORIZATION

User is prompted to provide their **private_key.pem** to identify themselves before being allowed to act as a **producer/consumer** in the **Kafka topic**:

Authentication of Kafka Producer:

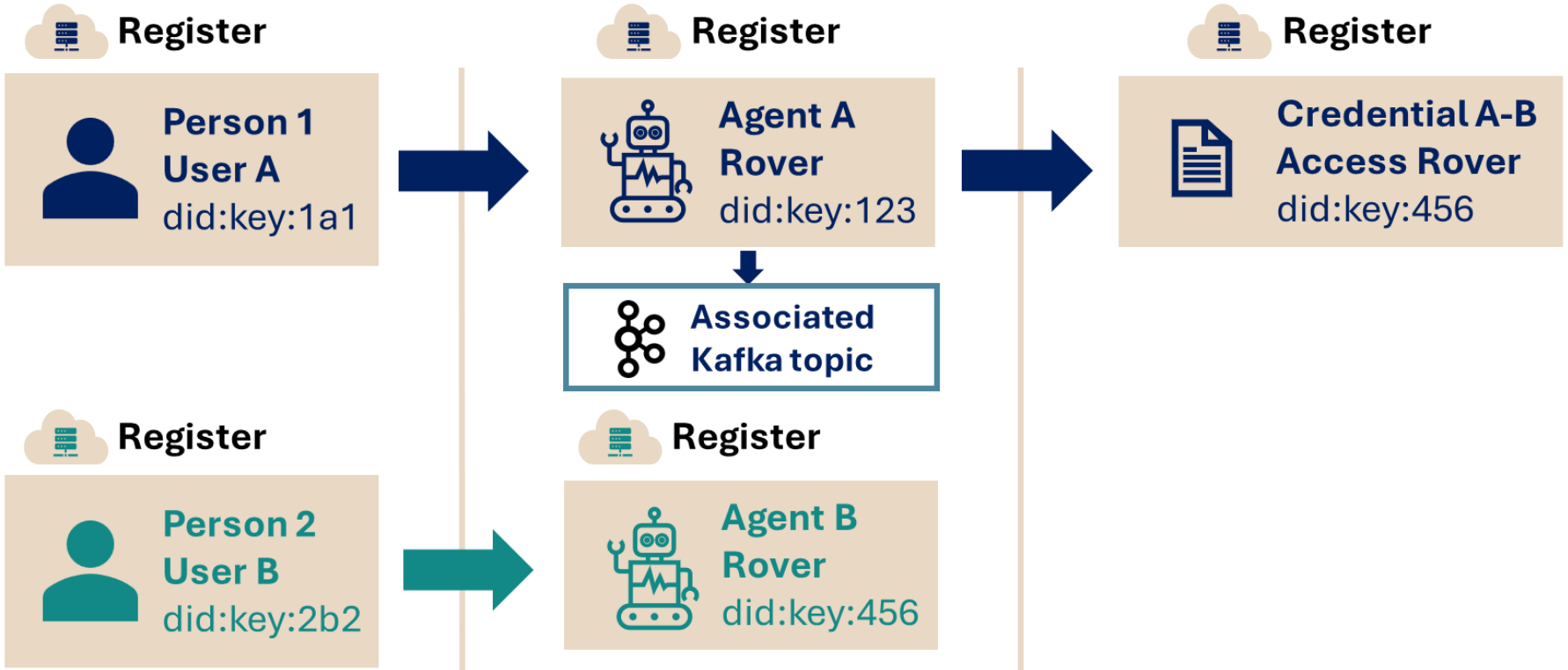
1. Asks for path to private key of producer Agent to use as password/identification.
2. Public key did:key is derived from private_key.pem using Cryptographic tool.
3. Checks against MySQL Registry that did:key generated is the did of the Agent associated to that topic_name
4. If successful, producer can start publishing data in the Kafka topic for the Agent.

Authorization of Kafka Consumer:

1. Asks for path to private key of consumer Entity to use as password/identification.
2. Public key did:key is derived from private_key.pem using CLI tool.
3. Checks against MySQL Registry that did:key generated is inside the allowed_did of the producer Agent associated to that topic_name
4. If successful, consumer can start subscribing to the data in the Kafka topic.

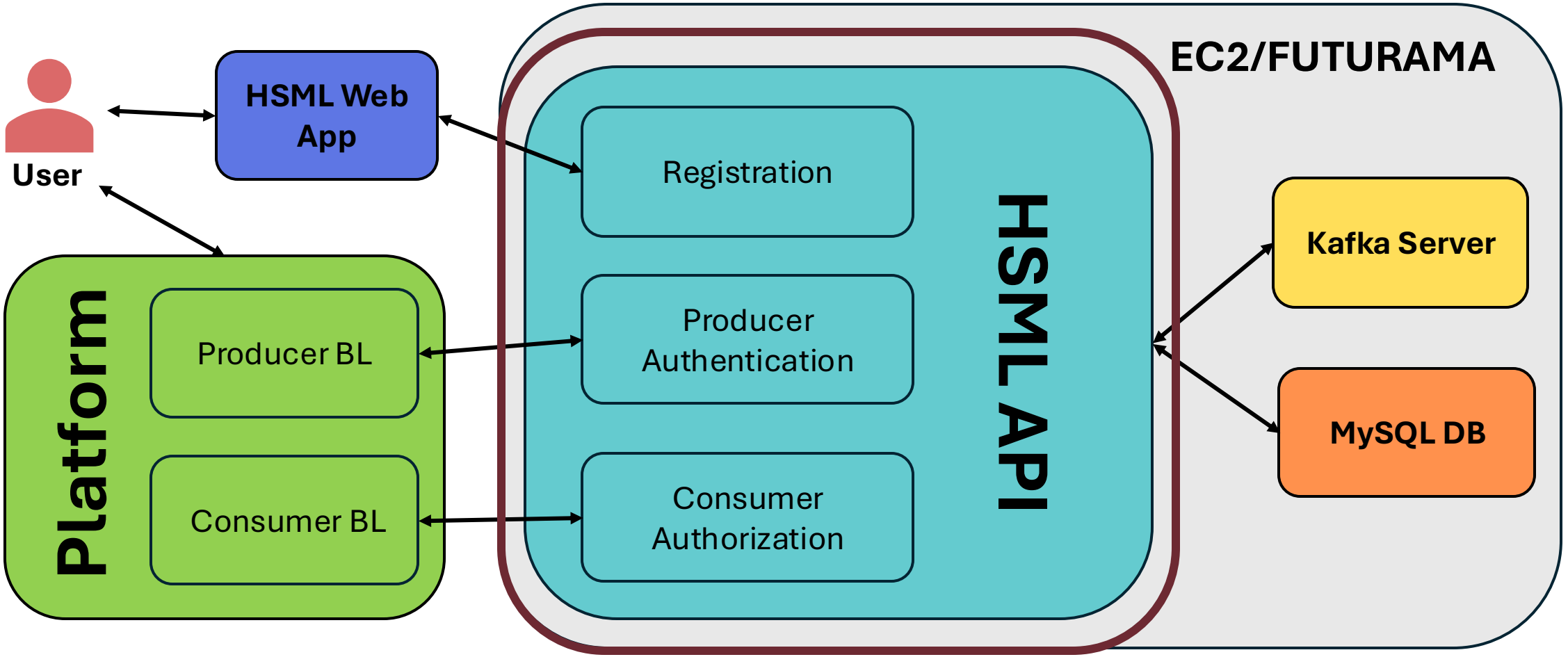
Methodology – Verification in the Spatial Web

VERIFICATION TESTING – MINIMUM ENTITIES NEEDED



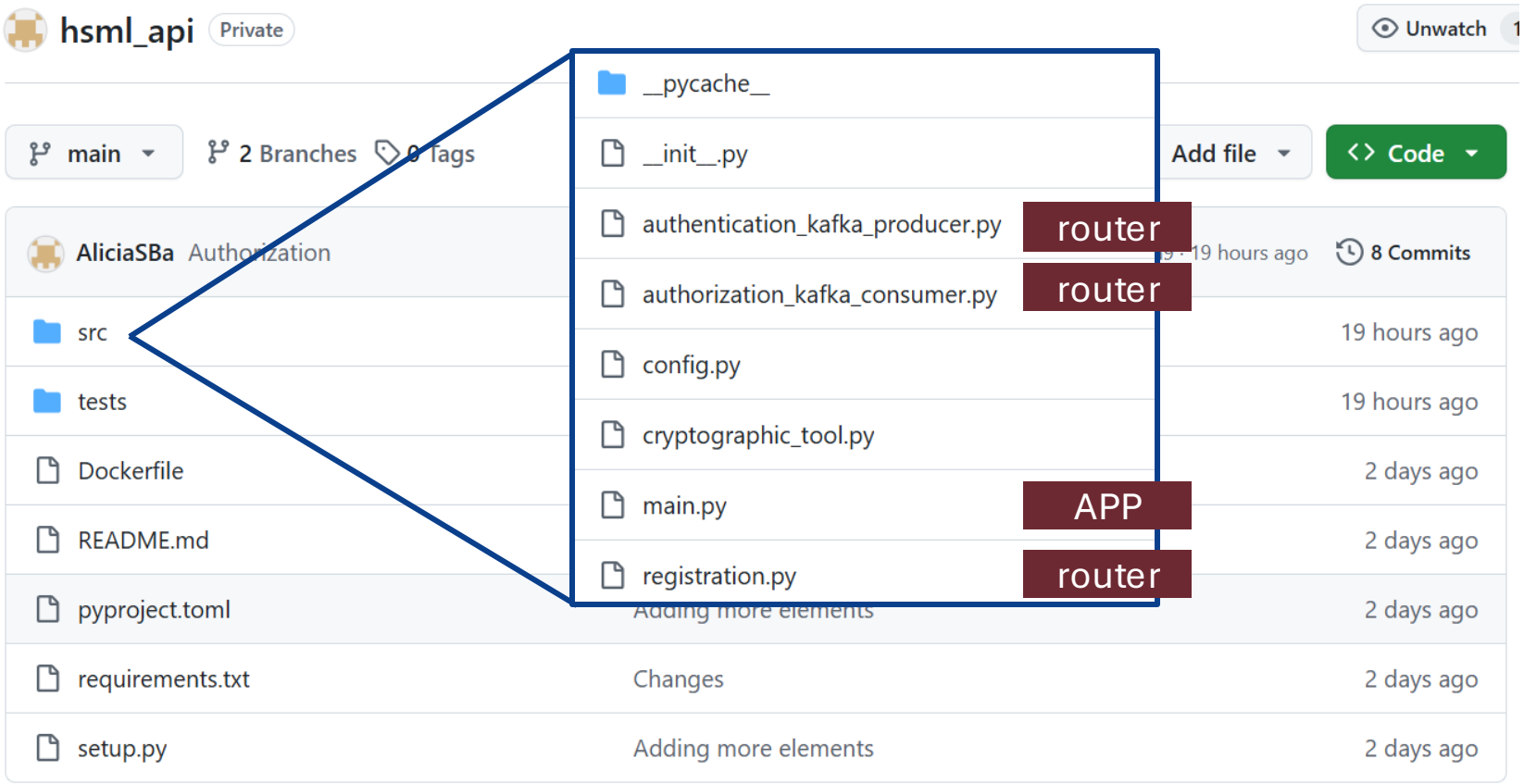
Methodology – HSML API

HSML API ARCHITECTURE



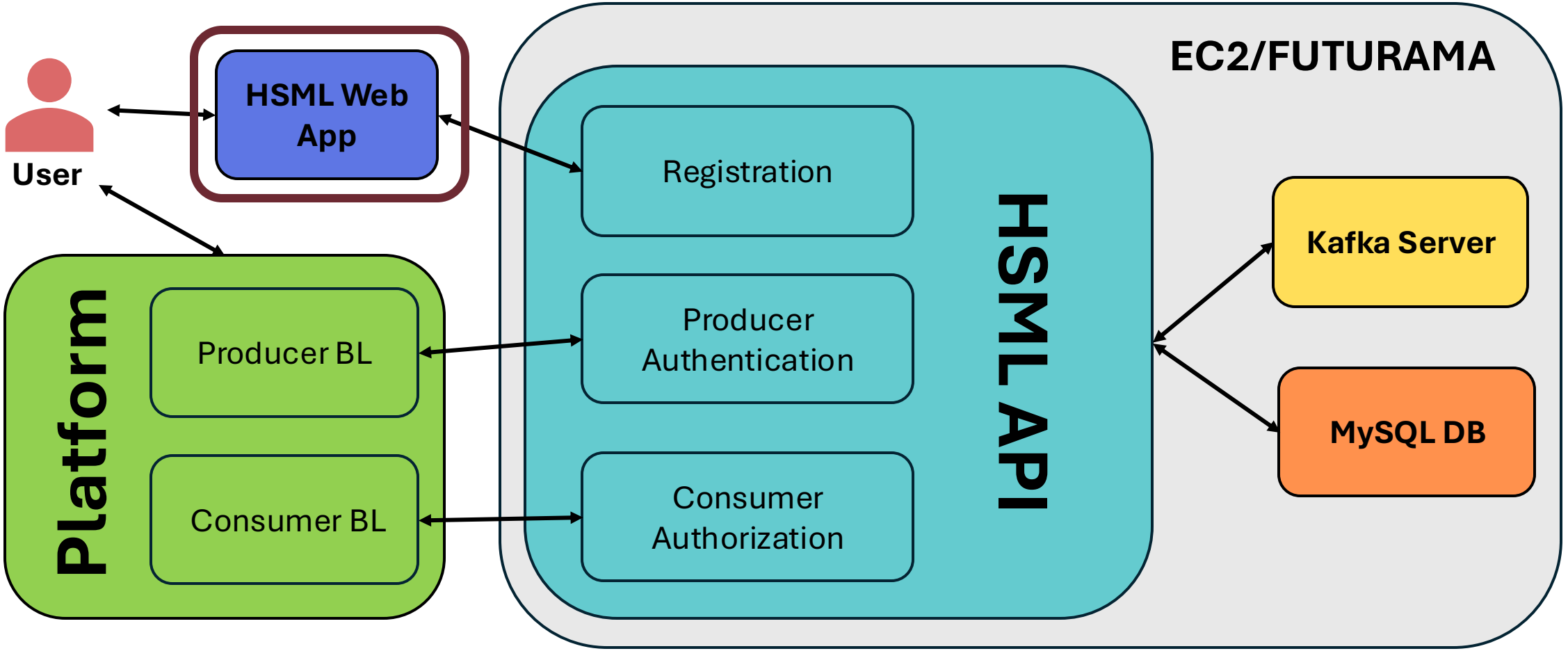
Methodology – HSML API

HSML API PACKAGE

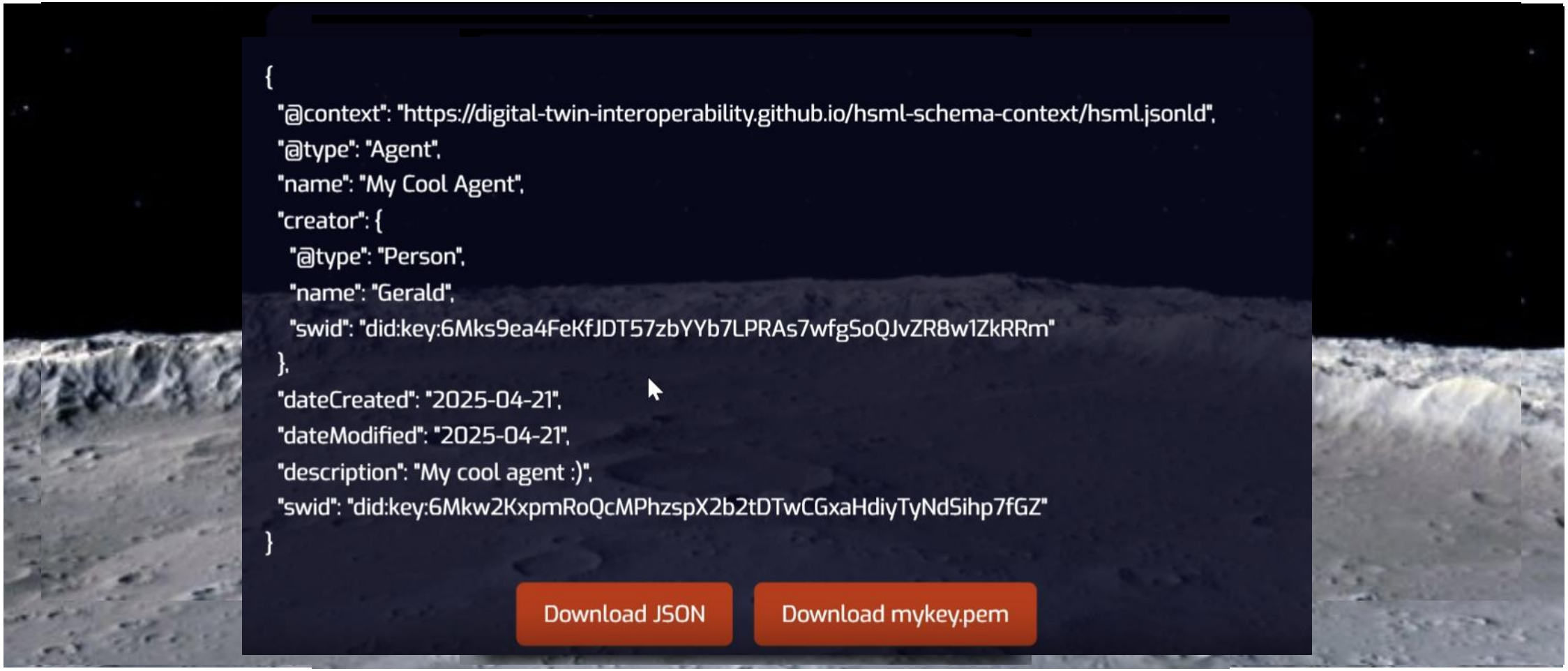


Methodology – HSML API

HSML API ARCHITECTURE



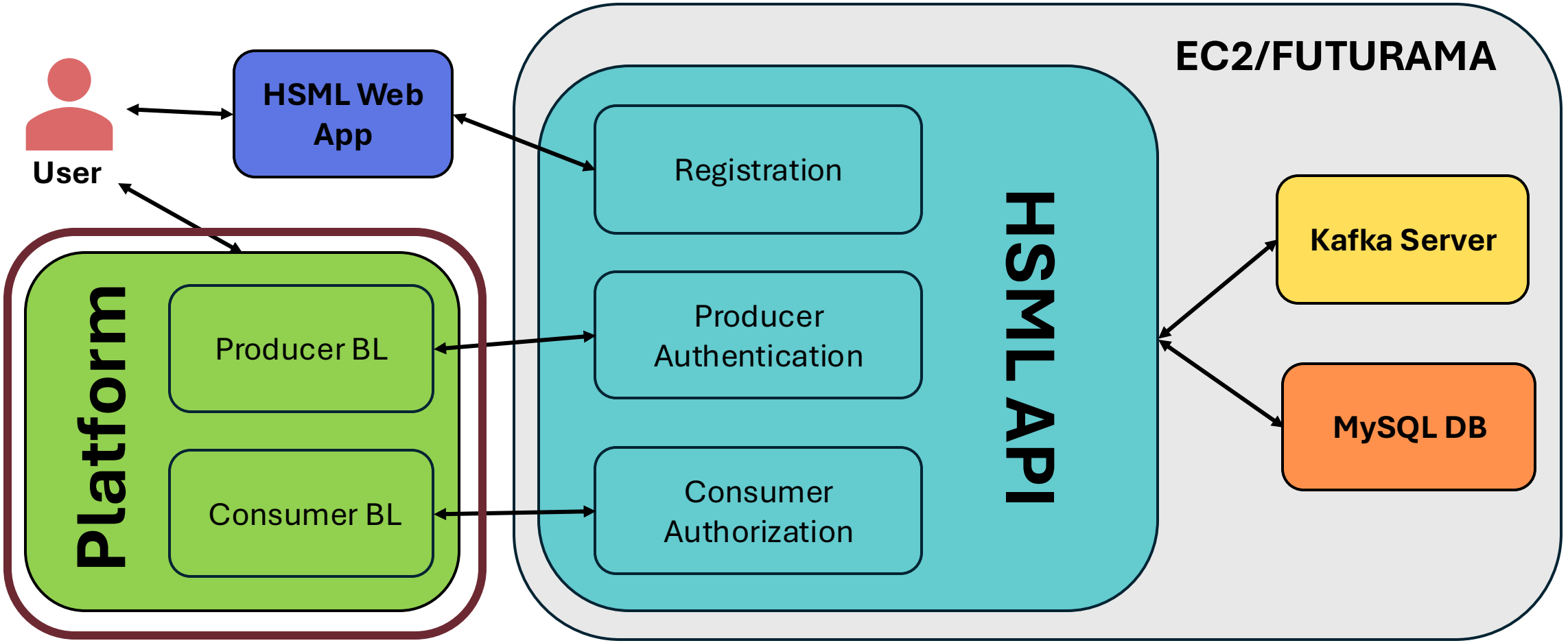
Methodology – HSML Web App for Registration



HSML Web App developed by Gerald & Gabriel

Methodology – HSML API

HSML API ARCHITECTURE



Methodology – Business Logic Plugins

Integrating the HSML API with the Platforms

NVIDIA OMNIVERSE

Attached to rover

- CadreAOmniProducer.py
- ViperBOmniConsumer.py
- ViperAOmniConsumer.py

Overwritten JSONs

- kafkaOmniProducer.json (Cadre A)
- kafkaOmniConsumer_1.json (Viper A)
- kafkaOmniConsumer_2.json (Viper B)

Run in terminal

- CadreAOmniAPIJSON.py
- ViperAOmniAPIJSON.py
- ViperBOmniAPIJSON.py

Unity

Attached to rover

- omniApiConsumer.cs
- unrealApiConsumer.cs
- unityApiProducer.cs

UNREAL ENGINE

Attached to rover

- omniConsumerCadreA.py
- unrealProducerViperB.py
- unityConsumerViperA.py

Overwritten JSONs

- UnrealConsumer_1.json (Cadre A)
- UnrealProducer.json (Viper B)
- UnrealConsumer_2.json (Viper A)

Run in terminal

- CadreAUnrealConsumer.py
- ViperBUnrealProducer.py
- ViperAUnrealConsumer.py

Experimental Setup

Platforms

Unity

UNREAL ENGINE

NVIDIA OMNIVERSE

Data Exchange

kafka

Verification & Registry

DID:key Method

SQL Database

MySQL

System Integration & Interface:

HSML API

FastAPI

WEB APP

Communication Standard:

HSML Schema

JSON

Infrastructure:

24.04 LTS

Windows 11

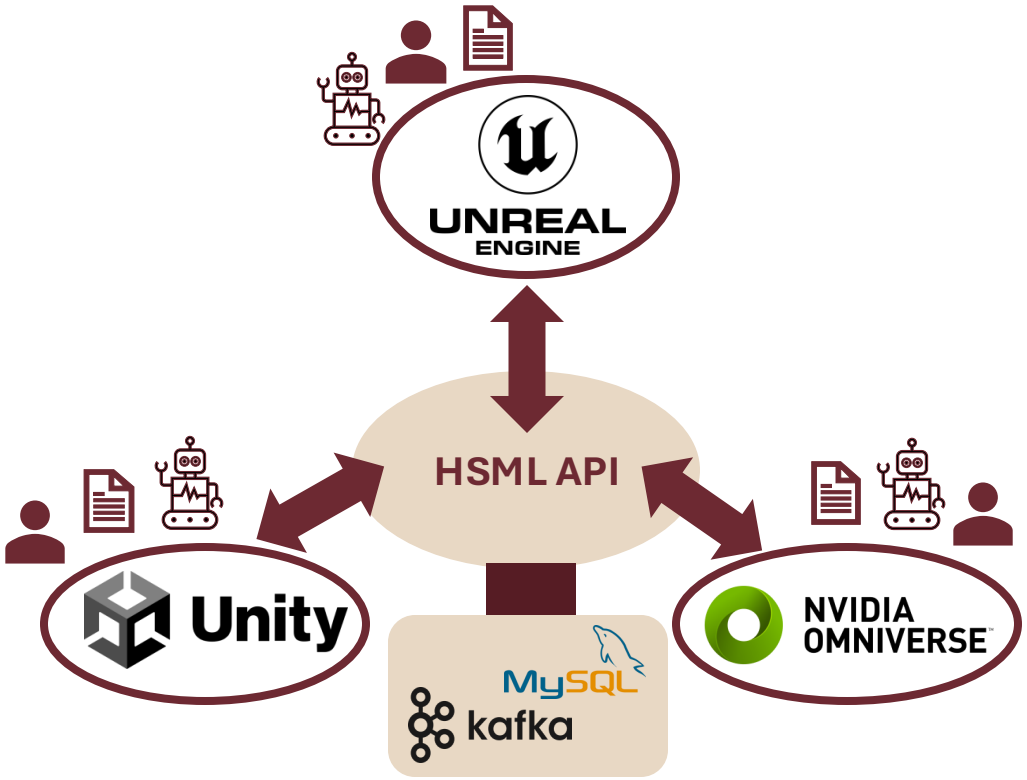
Experimental Setup

TESTING FRAMEWORK

TEST 1: Streaming **3D assets & positional data** between Unity, Unreal, and Omniverse using JPL Kafka Server

TEST 2: Interoperability across platforms using standardized **HSML protocol for data streaming**

TEST 3: Verification Test with Platforms using DID:key and SQL queries for Authentication & Authorization through the **HSML API**

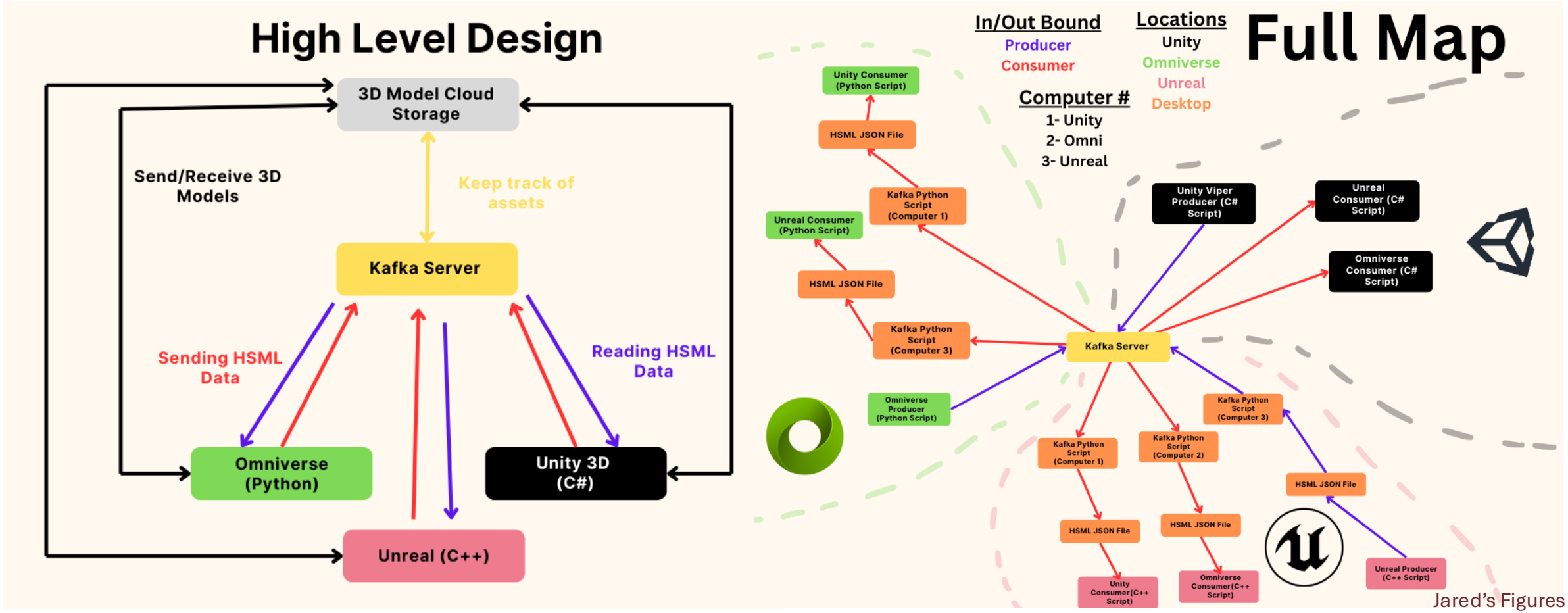


SOME VALIDATION METRICS:

- **Successful cross-platform data exchange** (Unity ⇄ Unreal ⇄ Omniverse)
- **Successful trust & security verification** using DID:key & MySQL Database
- **Latency** in sending data through Kafka Server and HSML AP

Experimental Setup

SYSTEM ARCHITECTURE WITHOUT HSML API - DIRECTLY THROUGH KAFKA

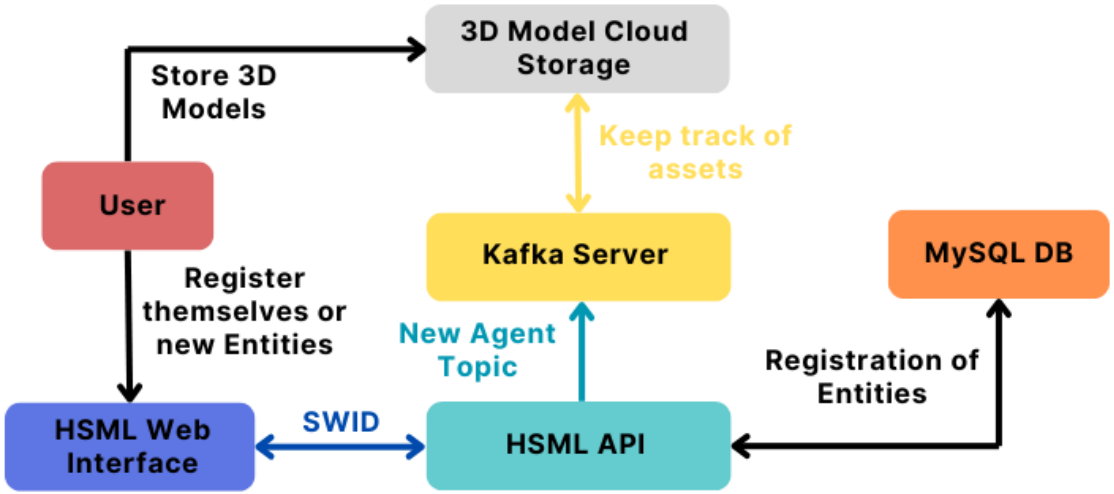


Jared's Figures

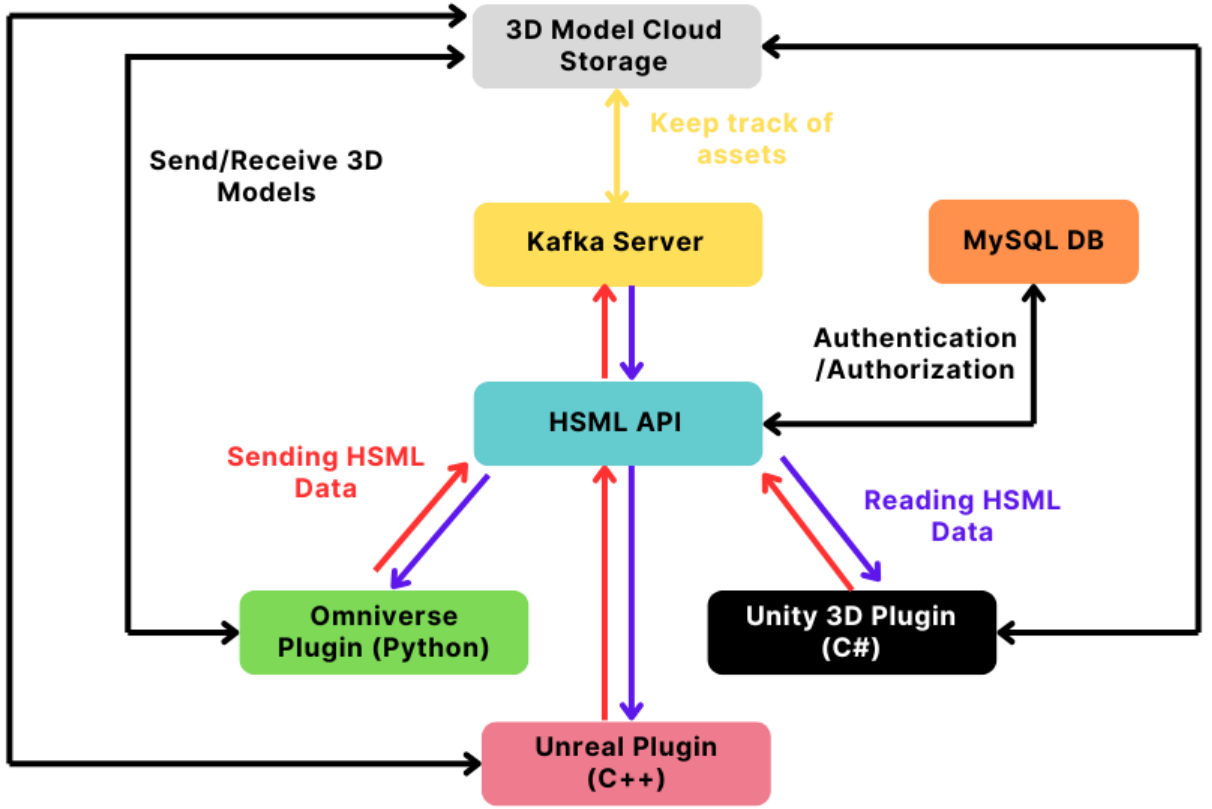
Experimental Setup

SYSTEM ARCHITECTURE WITH HSML API

Registration



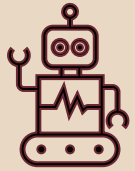
High Level Design



Experimental Setup

Producing/Consuming with HSML API

Producer

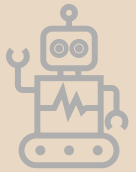


Original Viper A (Unity)

Topic: viper_a_0wawiy
Private Key: private_key_Viper_A.pem
Message: Viper A - HSML JSON

HSML API

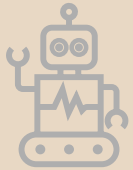
Consumer



Ghost Viper A (Omniverse)

Topic: viper_a_0wawiy
Private Key: private_key_Cadre_A.pem

Consumer

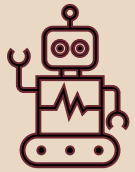


Ghost Cadre A (Unity)

Topic: cadre_a_qbnpru
Private Key: private_key_Viper_A.pem

HSML API

Producer



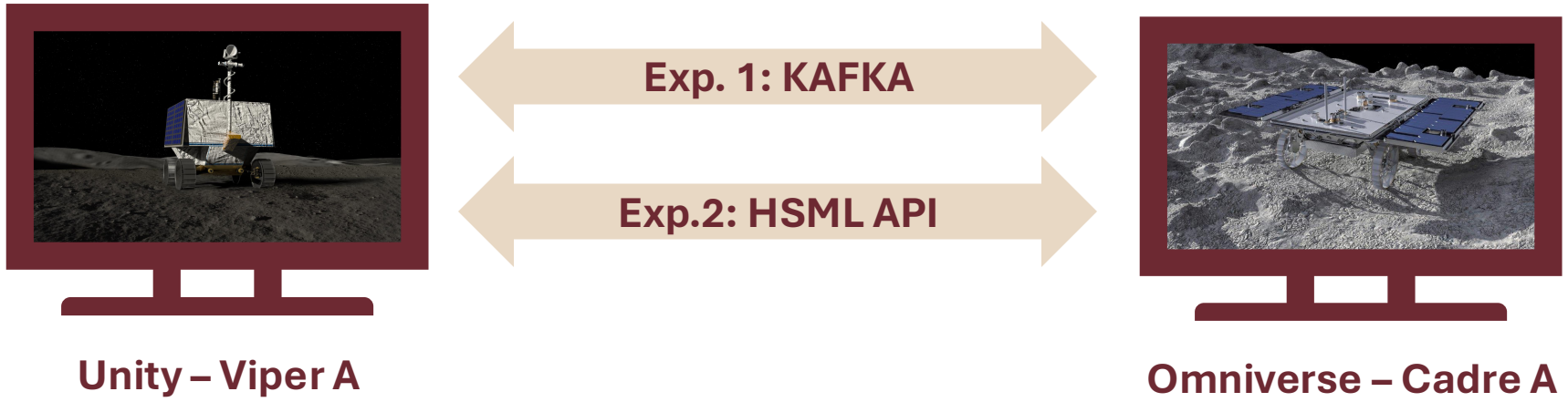
Original Cadre A (Omniverse)

Topic: cadre_a_qbnpru
Private Key: private_key_Cadre_A.pem
Message: Cadre A - HSML JSON

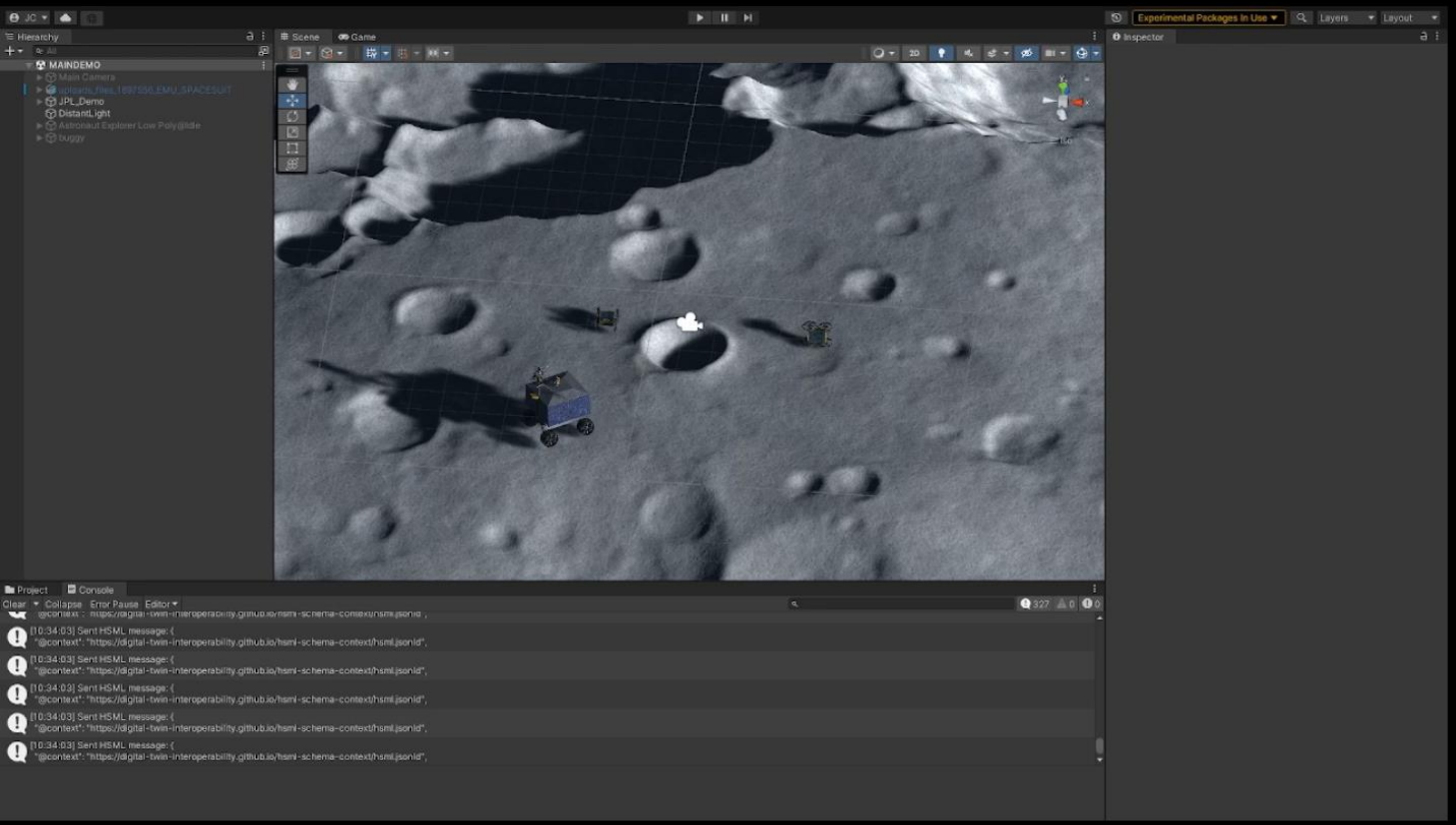
Testing and Integration

EXPERIMENT 1: Test of Unity and Omniverse exchanging positional and rotational data using HSML Schema directly through the Kafka Server

EXPERIMENT 2: Test of Unity and Omniverse exchanging positional and rotational data incorporating the HSML API for Verification of Identities for Producers/Consumers.



Digital Twin Interoperability using Kafka & HSML

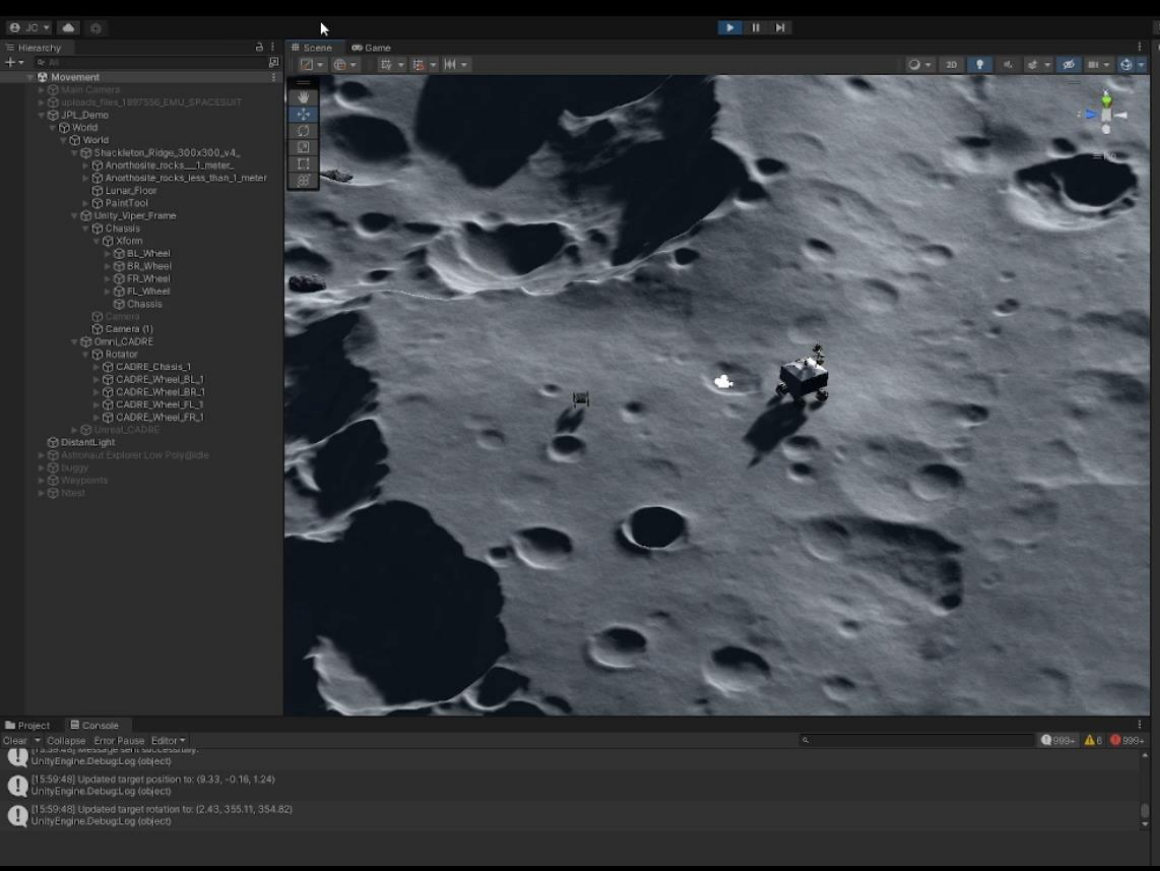


Kafka

Unity

Omniverse

Digital Twin Interoperability using HSML API



Unity

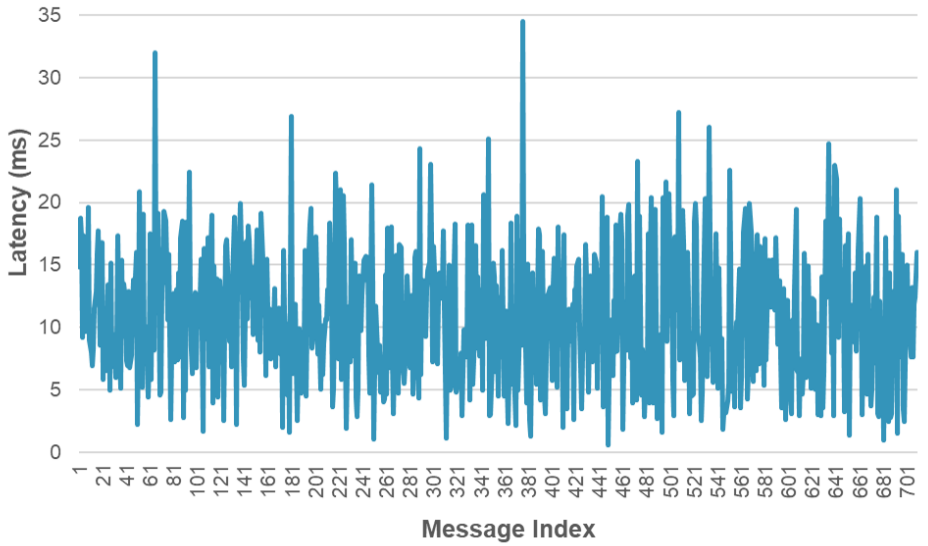
Omniverse

Key Results

WHY THE HSML API ACHIEVES LOWER LATENCY

- **Local Caching:** The HSML API stores the latest Kafka message in a local JSON file, allowing instant access via a simple file read instead of a real-time Kafka poll.
- **Persistent Connections:** Kafka producer and consumer objects are created once and reused, avoiding delays from repeated connection setups.
- **Authentication Once per Session:** Agents authenticate once and maintain session state, reducing the overhead of repeated credential checks.
- **Reduced System Load:** By separating message retrieval from direct Kafka queries, the API reduces pressure on the message broker and speeds up response time.

LATENCY WITH HSML API



Average latency = 10.83 ms, Max = 34.54 ms,
Min = 0.60 ms, Median = 10.57 ms, Std. dev. = 5.39 ms.

Key Results

Cross-Platform Interoperability Achieved (Unity ⇄ Omniverse)

HSML messages were successfully exchanged between Unity and Omniverse, demonstrating real-time digital twin synchronization using the **HSML Schema and the HSML API**.

Secure and Efficient Communication Implementing HSTP

Authentication and authorization using **DID:key and MySQL database** were integrated without compromising performance.

Real-Time Performance Validated

Average latency was 18.20 ms without verification and 10.83 ms with the HSML API. Being both well **below our objective of 50ms** and the threshold of human perception.

Scalable Framework Aligned with the Spatial Web Standard

HSML and HSTP proved effective for structured, trusted data exchange across distributed simulation platforms.

Challenges & Limitations

Challenges Encountered

- **Real-Time Interoperability Across Platforms.** Synchronizing digital twin behavior in Unity and Omniverse required careful coordination due to differences in physics engines.
- **Unreal Engine Integration.** Planned testing with Unreal was not completed due to persistent platform issues.
- **Data Consistency Across Platforms.** Platforms handled position and rotation data differently, affecting seamless message interpretation.

Known Limitations

- **Narrow HSML Data Scope.** Only basic attributes (e.g., position, rotation) were used; no complex payloads or sensors were tested.
- **Centralized Identity Management.** The system used a SQL database for authentication rather than a decentralized identity system.
- **Constrained Testing Environments.** Tests were performed on local Wi-Fi and EC2 servers, which may not reflect remote or space mission conditions.

Lessons Learned

- **HSML Schema** development requires balancing flexibility with structure — ontology design is iterative and platform-agnostic implementation is harder than expected.
- Building the **HSML API** showed the importance of modular design and validation checks to ensure reliability and ease of debugging.
- The **Spatial Web standard** offers a strong conceptual framework, but translating its abstract architecture into a working prototype revealed areas that still need clarification or extension.
- **Platform integration** (Unity, Omniverse, Unreal) showed that interoperability isn't just about data format, but differences in platform physics, protocols, and infrastructure all affected integration.
- **Simulation testing** helped surface real integration issues and showed that secure data exchange can still achieve real-time performance when optimized.

Recommendations & Future Work

1. **Complete Unreal Engine Testing:** Finalize the integration and validation of Unreal Engine to complete the intended cross-platform setup.
2. **Expand HSML Schema and Platform Coverage:** Add mission-relevant data types (e.g., sensor readings, telemetry, commands) and extend validation to proprietary and space-industry platforms (i.e., DARTS).
3. **Simulate Realistic Multi-Agent Scenarios:** Move beyond basic synchronization to test agent interaction, coordination, and feedback loops in more complex mission setups.
4. **Implement Additional Spatial Web Components:** Incorporate other elements of the Spatial Web standard, such as Channels, Activities, and Contracts, to enable richer workflows.
5. **Advance Decentralized Identity and Governance:** Replace centralized verification with decentralized identity systems, and introduce role-based access and copyright tracking mechanisms.

Conclusion

- **Demonstrated Real-Time, Secure Interoperability.** Achieved cross-platform communication between digital twins using the Spatial Web standard (HSML + HSTP).
- **Validated the HSML API Stack.** Integrated registration, authentication, and authorization via DID:key & MySQL Database; all with low-latency performance.
- **Confirmed Platform Compatibility (Unity & Omniverse).** Showed consistent message handling across engines; Unreal integration remains a next step.
- **Established a Scalable Foundation for Space Applications.** The system provides a flexible framework for future multi-agent collaboration, data trust, and interoperability in space missions.
- **Contributes to the Vision of the Spatial Web in Aerospace.** Offers a working proof of concept toward using semantic, decentralized standards for digital twin systems in lunar and planetary exploration.

Acknowledgements

A special thank you to all the people in my lab who supported and contributed to this project:

My Mentors

- Dr. Thomas Lu (Senior Researcher, Advanced Electronics Systems & Technology Group, JPL)
- Dr. Edward Chow (Manager, Civil Program Office, JPL)

Fellow Interns (Group 345C)

- Gerald Parish
- Jared Carrillo
- Subhobrata Chakraborty
- Joshua Drye
- Niki Namian
- Sohee Kim
- Diego Cordova
- Aaron Hui
- Gabriel Venezia
- Sydni Yang


THANK YOU!
NASA
JPL
Holiday
BADWATER BASIN
282 FEET/855 METERS
BELOW SEA LEVEL
PERON

ADDITIONAL SLIDES

Skills Gained

- **Ontology and Schema Design.** Learned how to build and iterate on a semantic data model (HSML) for platform-agnostic communication.
- **API Development and Validation.** Gained experience developing a RESTful API with FastAPI, including schema validation, modular architecture, and error handling.
- **Kafka-Based Communication.** Implemented secure producer-consumer workflows using Kafka, integrating identity-based access control.
- **Cross-Platform Simulation Integration.** Worked with Unity, Omniverse, and Unreal to set up and test real-time digital twin synchronization.
- **Secure Identity Management.** Applied DID:key for decentralized authentication and authorization using cryptographic tools and SQL-based registries.
- **System Testing and Performance Analysis.** Designed and ran experiments to measure latency, verify message integrity, and evaluate interoperability under load.

Challenges

Real-Time Interoperability Across Different Physics Engines

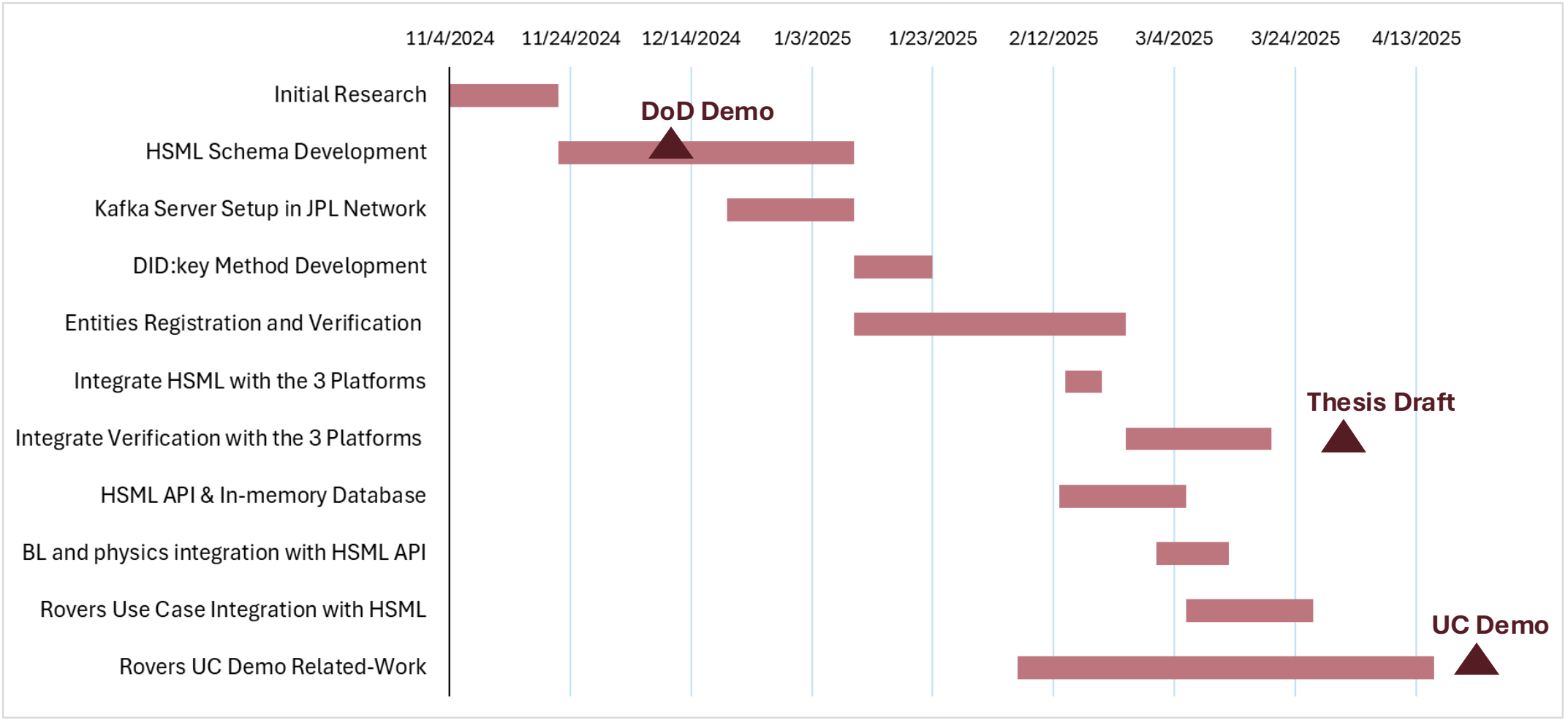
Integrating **Diverse Data Types** into HSML

Ensuring **Cross-Platform Compatibility** for Digital Twins

Minimizing Data Streaming Overhead for Low-Latency Performance

Servers and Physics Engines Limitations and Issues (EC2, Unreal, Local Wi-Fi)

Project Timeline – Gantt Chart



Research Questions

RQ: How can **standardization** be applied to achieve **digital twin interoperability in space exploration**?

SQ1: Can **different simulation platforms** (Omniverse, Unity, and Unreal Engine) effectively **exchange data using the HSML ontology**?

SQ2: How does the **implementation of HSML and HSTP** impact interoperability, data integrity, and communication latency across digital twins?

SQ3: Does the authentication framework based on HSTP and DIDs ensure **secure and trusted data sharing** in a multi-stakeholder space environment?

SQ4: What are the **challenges of integrating standardized digital twins** into real-time space operations, and how can they be addressed?

MySQL Database – Registry Database

MySQL is an open-source **relational database management system** (RDBM) that stores data on disk by default (note: it is not an in-memory database).

Purpose: We want to use it as a **permanent registry of the HSML Entities**, in which we **store the SWID (the DID public key) and the Entities metadata in HSML JSON files**. This database would be used as a verification mechanism as it would contain all the HSML Entities identities and their connections to others. This way we can identify the users (Person type) and the Organizations they belong to, the Agents and their owners (Domain Authorities), the Hyperspaces they occupy, which Agents have Credentials associated to them and who has therefore access, the Contracts associated to Activities…

Why MySQL is the best option for Verification:

- Ideal for **long-term, structured storage**
- **Supports JSON** (files) columns & for **multiple users** to access/manage data simultaneously
- Supports complex SQL queries for **metadata filtering & verification**

HSML API - FastAPI docs

FastAPI 0.1.0 OAS 3.1

/openapi.json

default

POST	`/entity/register/`	Register Entity Api
POST	`/producer/authenticate`	Api Authenticate
POST	`/producer/start`	Start Producer
POST	`/producer/stop`	Stop Producer
POST	`/producer/send-message`	Send Message
POST	`/consumer/authorize`	Api Authorize

HSML API from the terminal

HSML API

```
C:\Users\abarrio\OneDrive - JPL\Desktop\github_repos\hsml_api\src>python main.py
INFO:     Started server process [24780]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

Producer (Agent 4)

```
C:\Users\abarrio\OneDrive - JPL\Desktop\github_repos\hsml_api\tests>python test_authenticate.py
{'message': 'Authentication successful for topic example_agent_4'}
{'message': 'Producer started for topic example_agent_4'}
{'message': 'Message sent successfully'}
Press Enter to stop the producer...
```

Consumer (Agent 4)

```
C:\Users\abarrio\OneDrive - JPL\Desktop\github_repos\hsml_api\tests>python test_authorize.py
{'message': 'Authorization successful for topic example_agent_4'}
{'message': 'Consumer started for topic example_agent_4'}
Press Enter to stop the consumer...
```

Message (HSML JSON)


```
Received message on 'example_agent_4': {"@context": "https://digital-twin-interoperability.github.io/hsml-schema-context/hsml.jsonld"
, "@type": "Agent", "name": "unrealActor", "swid": "did:key:generate-unrealActor-001", "identifier": {"@type": "schema:PropertyValue"
, "propertyID": "schema_id", "value": "unrealActor-001"}, "url": "objectLink", "creator": {"@type": "Person", "name": "Jared Carrillo
"}, "dateCreated": "12-06-24", "dateModified": "12-06-24", "encodingFormat": "application/x-obj", "contentUrl": "https://example.com/
models/3dmodel-001.obj", "description": "Unreal engine object", "platform": "Unreal Engine", "spaceLocation": [{"@type": "Hyperspace"
, "name": "Moon"}], "additionalType": "https://schema.org/CreativeWork", "position": [{"@type": "schema:PropertyValue", "name": "xCoo
rdinate", "value": 424.4965405147641}, {"@type": "schema:PropertyValue", "name": "yCoordinate", "value": -269.99999999999994}, {"@typ
```

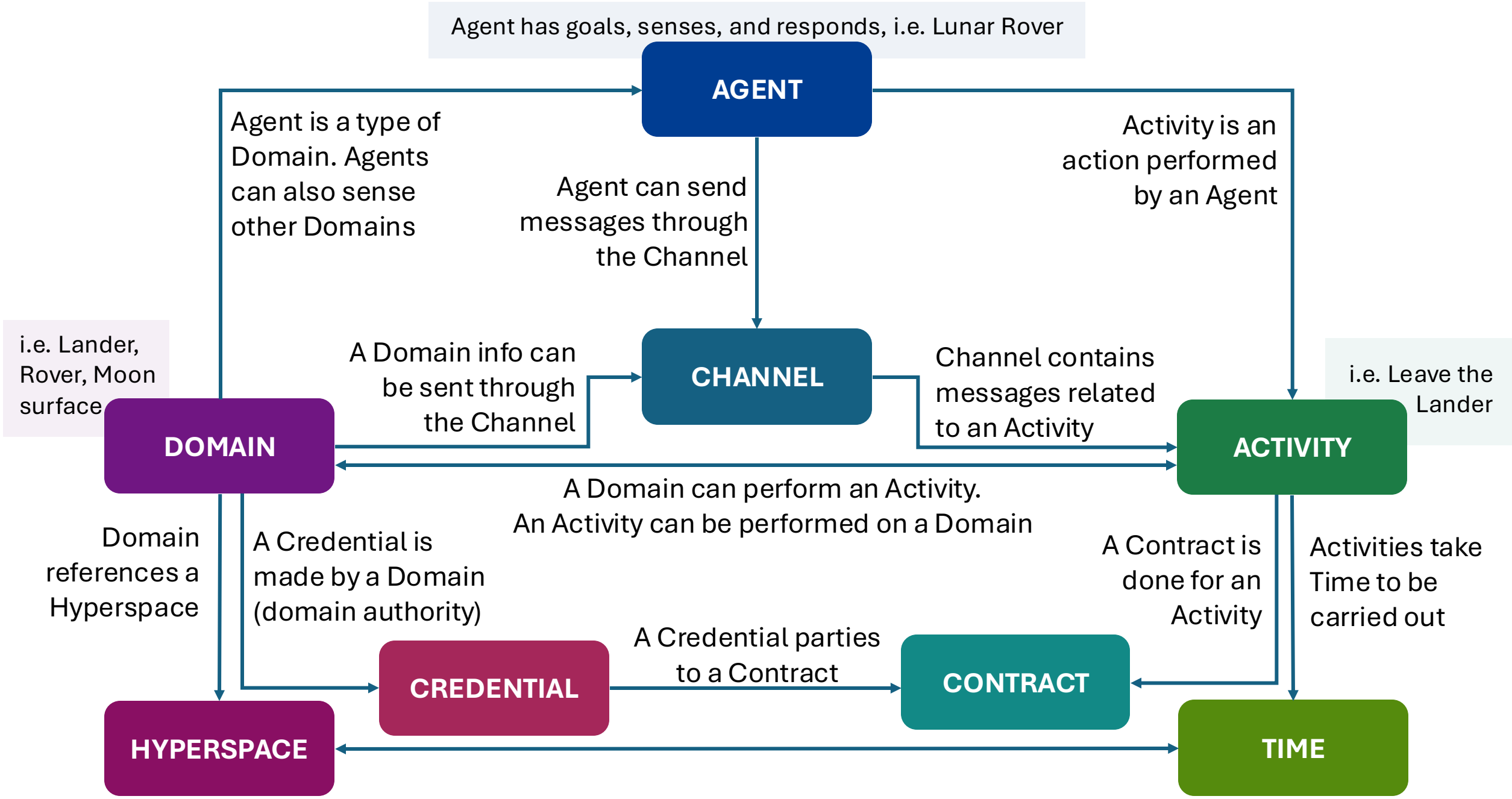
Spatial Web Ontology Entries

- **ENTITIES** are the base items used across the Spatial Web in HSML

The Spatial Web ontology defines several classes. All classes are types of the Entity base class:

- **ACTIVITY**
- **AGENT**
- **CONTRACT**
- **CHANNEL**
- **CREDENTIAL**
- **DOMAIN**
- **HYPERSPACE/SPACE**
- **TIME**

HSML encodes ENTITIES in a format, e.g., JSON


Agent has goals, senses, and responds, i.e. Lunar Rover
AGENT
Agent is a type of Domain. Agents can also sense other Domains
Agent can send messages through the Channel
Activity is an action performed by an Agent
i.e. Lander, Rover, Moon surface
A Domain info can be sent through the Channel
CHANNEL
Channel contains messages related to an Activity
i.e. Leave the Lander
DOMAIN
ACTIVITY
A Domain can perform an Activity.
An Activity can be performed on a Domain
Domain references a Hyperspace
A Credential is made by a Domain (domain authority)
A Contract is done for an Activity
Activities take Time to be carried out
A Credential parties to a Contract
CREDENTIAL
CONTRACT
HYPERSPACE
TIME

In summary,

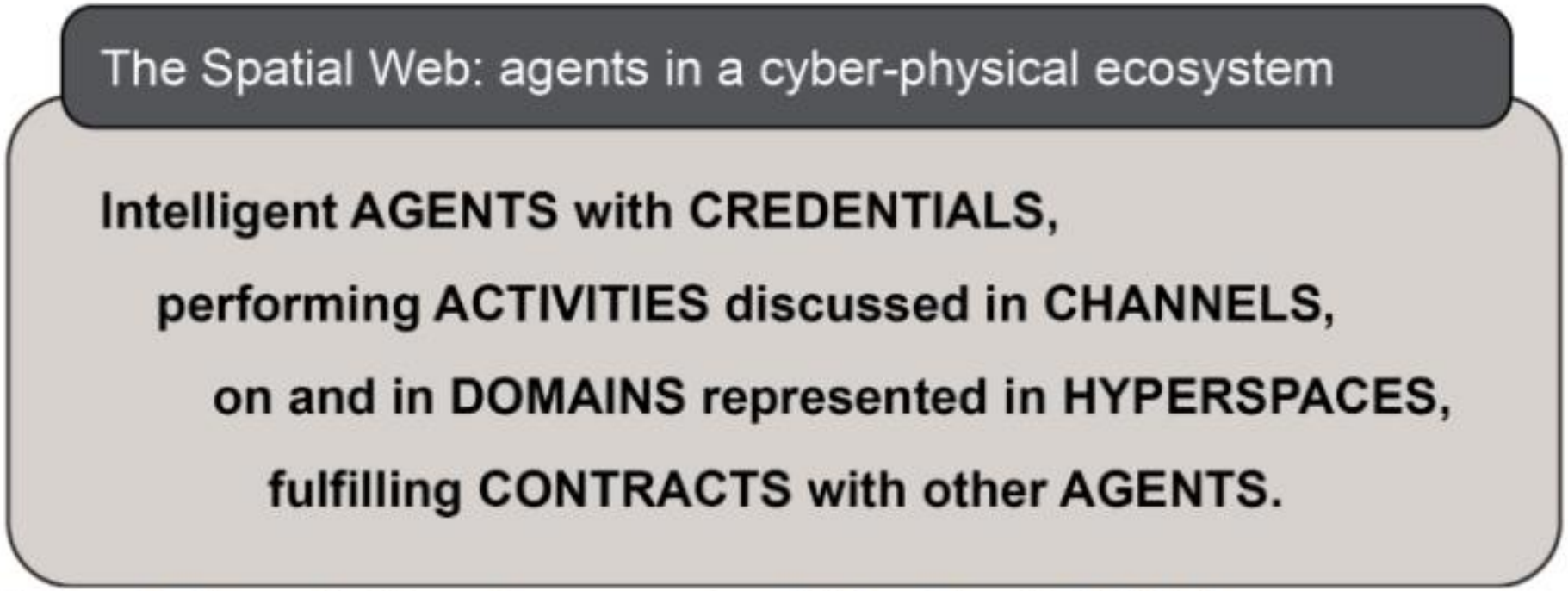


Figure 2 foundational concepts of the Spatial Web

From 979-8-3503-5597-0/25/$31.00 ©2025 IEEE,
Enabling Interoperable Digital Twins for Collaborative Lunar Exploration

HSML Types Structure

Entity

An HSML Class

Entity

The base item in the Spatial Web Ontology

Property	Expected Type	Description
Properties from **Thing**		
additionalType	Text or URL	An additional type for the item, typically used for adding more specific types from external vocabularies in microdata syntax. This is a relationship between something and a class that the thing is in. Typically the value is a URI-identified RDF class, and in this case corresponds to the use of rdf:type in RDF. Text values can be used sparingly, for cases where useful information can be added without their being an appropriate schema to reference. In the case of text values, the class label should follow the schema.org style guide.
alternateName	Text	An alias for the item.
description	Text or TextObject	A description of the item.

HSML Documentation developed by me

Thing

A Schema.org Type

Thing

[more...]

The most generic type of item.

Property	Expected Type	Description
Properties from **Thing**		
additionalType	Text or URL	An additional type for the item, typically used for adding more specific types from external vocabularies in microdata syntax. This is a relationship between something and a class that the thing is in. Typically the value is a URI-identified RDF class, and in this case corresponds to the use of rdf:type in RDF. Text values can be used sparingly, for cases where useful information can be added without their being an appropriate schema to reference. In the case of text values, the class label should follow the schema.org style guide.
alternateName	Text	An alias for the item.
description	Text or TextObject	A description of the item.

Schema.org webpage

HSML Schema Context and Example

[…]

HSML schema context **hsml.jsonld** file stored in GitHub pages domain: https://digital-twin-interoperability.github.io/hsml-schema-context/hsml.jsonld

Contains Schema.org properties and types as well as the HSML types and custom properties

```
{
  "@context": "https://digital-twin-interoperability.github.io/hsml-schema-context/hsml.jsonld",
  "@type": "Entity",
  "name": "Example Entity",
  "description": "This is an example entity type.",
  "swid": "did:key:6MkvWxmpULqn65HWJ7id7GF2G8CGWeGs9GvftPw3ZPESuLC"
}
```

Entity Example

Attributes of HSML ENTITY

- **Identifiers** – SWIDs[1] that conform to W3C did-core, https://www.w3.org/TR/did-core/.
- **Name**
- **Description**
- **Links** to other ENTITIES
- **Properties**
- **Type**: either “ENTITY” or one of the eight semantic entities in the Spatial Web ontology

All HSML objects derive from the Entity class.

[1]SWID ≡ Spatial Web Identifier


```
{
  swid: "swid:schema:Domain000000000000000"
  __archived: false
  schema: [
    0: "swid:schema:Schema000000000000000"
    1: "swid:schema:Entity000000000000000"
  ]
  name: "domain"
  description: "An HSML domain"
  additionalProperties: false
  properties: {
  }
  type: "object"
}
```

SWID and DID

The **Identifier for the HSML schema**, according to the Spatial Web standard, is the SWID. The **Spatial Web Identifier (SWID)** is a Decentralized Identity (DID) that conforms to W3C did-core, https://www.w3.org/TR/did-core/.

```
swid: "swid:schema:Domain000000000000000"
```

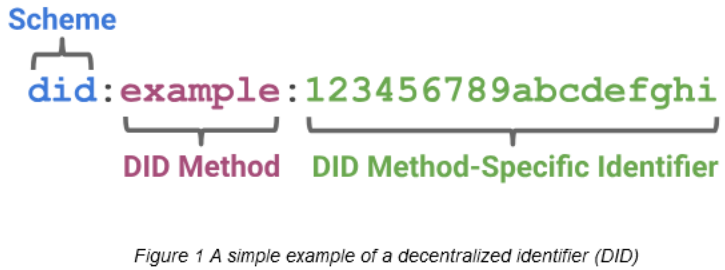


Figure 1 A simple example of a decentralized identifier (DID)

NO EXISTING SWID METHOD

USE AN EXISTING DID METHOD

Chosen method: **did:key**

Novelty of Our Work

Implementing the Spatial Web standard to enable real-time **interoperability** across **Digital Twins** from different platforms and physics environments, fostering future collaboration in space exploration.

- **First DT Interoperability Test** using Unity, Unreal Engine, and Omniverse.
- **Integration via Spatial Web**, applying **HSML** for standardized communication.
- **Building Trust in Data Exchange**, following Spatial Web's decentralized architecture, by using unique identification and verification.
- **Automation with AI**, fine tuning LLMs to build the business logic plugins and drive the format conversion for DTs

DID Method

Primary Choice: **did:key**

Reason: It's fast, lightweight, and doesn't require hosting a website or interacting with a blockchain. No domain registration, no storage, and no costs.

How this would work with Kafka:

1. To create a DID – use app or a CLI tool (can build a CLI tool with python)
2. Tool generates a key pair (public + private key)
3. Creates DID from the public key (like did:key:xyz123)
4. Your private key stays secret, and the DID becomes your public identifier

DID:key Method Implementation

A) MANUAL PROCESS

For each new Entity, user runs → **CLItool.py** generates a key pair

python CLItool.py --export-private

Public Key: Appears in JSON file as "swid": "did:key121lafj…"

Private Key: Stays secret (stored by user)

Using DID for Authentication:

- **Producers** sign objects with private key
- **Consumers** verify their signature using the public key

```
PS C:\Users\abarrio\OneDrive - JPL\Desktop\Digital Twin Interoperability\Codes\HSML codes> python CLItool.py --export-private
Generated DID:key: did:key:6MkjhYSEVgRNUkkPSo871xSCDqMpSqeq7QhuN3vT9Hch5bw
Private key saved to private_key.pem
```

DID:key Uniqueness

It is extremely unlikely for two objects to receive the same DID:key

1. Cryptographic Randomness in Key Generation

- CLItool utilizes a secure random number generator (CSPRNG) to create the private key. The Ed25519 algorithm operates on a **256-bit key space**, which means there are **22562^{256} (about 107710^{77}) possible** keys, which means that the **probability of collision is ~0** even if billions of keys generated.

2. Deterministic Public Key Derivation

- The public key is derived deterministically using the Ed25519 algorithm (same private key will always result in the same public key).

3. Multicodec and Base58 Encoding

- The public key is combined with a unique multicodec prefix (b'\xed\x01' for Ed25519) and encoded in Base58 (DID:key format is consistent).

DID Methods

Criteria	DID:ETHR	DID:WEB	DID:KEY
Privacy	**Medium** – Transactions are on a public Ethereum ledger, but can use zk-SNARKs for privacy.	**Low** – Relies on centralized control of DNS or domain hosting.	**Low** – No on-chain record, but depends on how keys are handled.
Easiness to Implement	**Medium** – Requires an Ethereum wallet and knowledge of smart contracts.	**Easy** – Requires ownership of a domain name and a simple JSON file hosted at a URL.	**Easy** – Purely key-based, no blockchain required.
Storage Needed	**High** – Stores state on Ethereum, costly for large datasets.	**Medium** – DID document is stored in a simple JSON file on the domain's web server.	**Low** – No storage required as it's entirely key-based.

Registration using MySQL Database

The Registration Tool: User is prompted to upload their HSML JSON file of the new Entity they want to register; they must log in or register themselves first.

1. **Validates the JSON file** (format, mandatory fields, and type check).
2. Checks for existing swid and warns the user if it will be overwritten.
3. Generates the **public/private key pair using the Cryptographic tool**.
4. Attaches the did:key to the swid property in the JSON.
5. **Stores** the public key and updated JSON **in the MySQL database**.
6. Provides the private key (private_key.pem) and updated JSON to the user.

Why MySQL is the best option for Verification:

- Ideal for **long-term, structured storage**
- **Supports JSON** (files) columns & for **multiple users** to access/manage data simultaneously
- Supports complex SQL queries for **metadata filtering & verification**

About the Project – Spatial Web Protocols

HSML ≡ Hyperspace Modeling Language

- A multi-dimensional ontology for encoding fundamental elements and the relationships between them.

HSTP ≡ Hyperspace Transaction Protocol

- A multi-dimensional range query for governing the interactions between actors and assets

HSQL ≡ Hyperspace Query Language

- Syntax for performing predictive (probabilistic) queries on a vector graph document database which combines comparative, relational, and similarity searching

From P2874/D3.1, June 2024 Draft Standard for Spatial Web Protocol, Architecture and Governance:

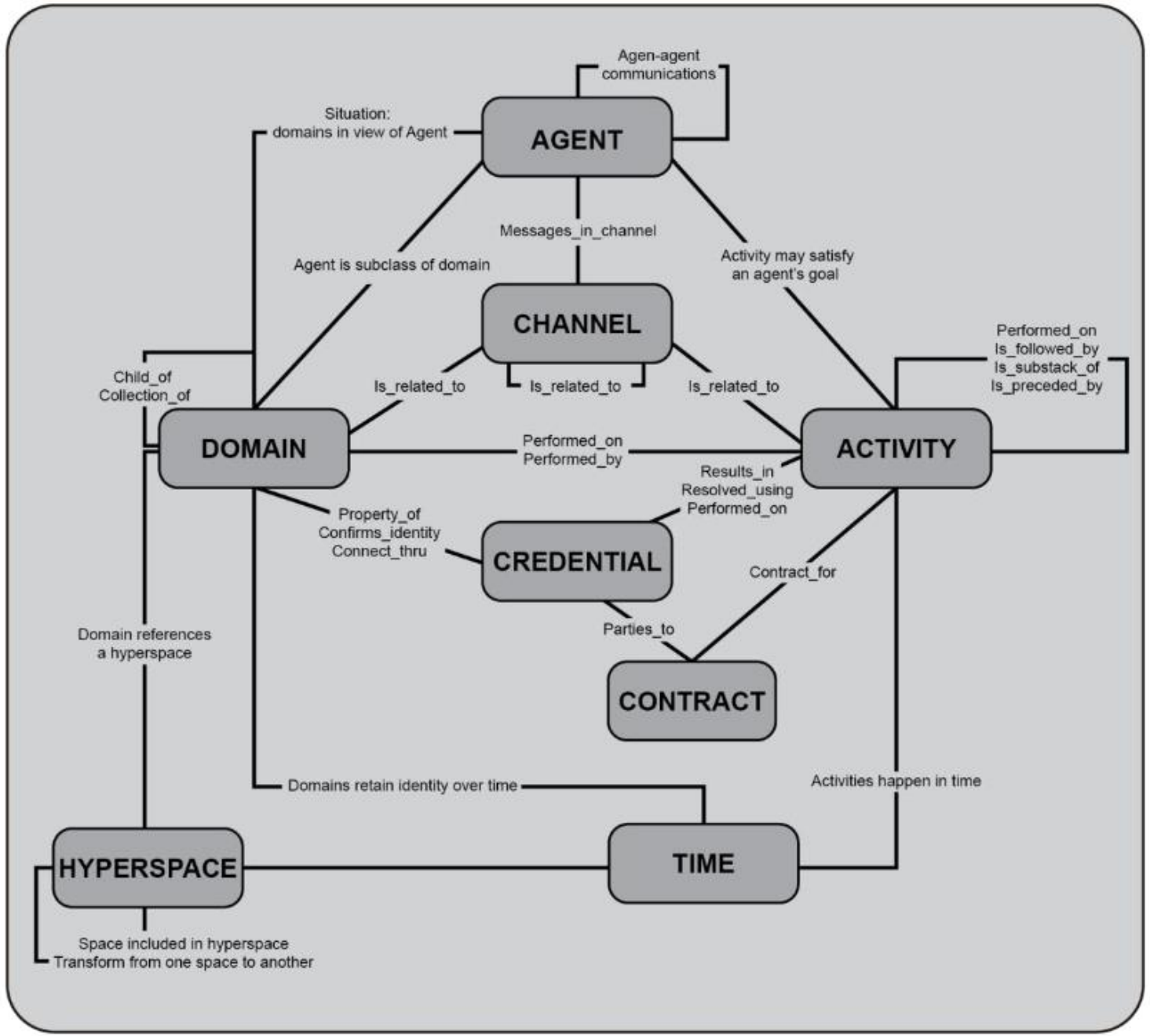


Figure 23 —Spatial Web entity relationship diagram

Credentials in Kafka

Verifying credentials, identities, and access rights requires a combination of **cryptographic techniques**, **authentication protocols**, and **access control mechanisms**.

Credential Verification, **Domain Identity**, and **Access Control** in **Kafka:**

1. **Issue Decentralized Identifiers (DIDs)** for producers, consumers, and brokers.
2. **Use Verifiable Credentials (VCs)** to grant roles like "producer" or "consumer" for specific topics.
3. **Authenticate users** using **DID-Auth** (challenge-response).
4. **Check role-based access** for producers/consumers.
5. **Use encryption and signing** for message confidentiality and integrity.

Safety and Trustworthiness in the Spatial Web

"Safety & Trustworthiness in the Spatial Web is enabled by **Governance**"

The main concepts for Governance are:

- **Credentials** – used for authentication and identity – who can access a domain, and how can they interact with it
- **Norms** – act as a principle of conduct or right action over domains
- **Contracts** – binding agreement used to dictate activities and relationships among domains

4 aspects of trust between parties intermediated by digital technologies:

1. **Authenticity** – SWIDs (unique identifier)
2. **Integrity**
3. **Confidentiality** – HSTP
4. **Privacy** – Design principle

Space Companies using Digital Twin Technology

SpaceX

- Dragon Capsule Spacecraft
- Siemens NX software on the design of entire rocket, to ensure that their rocket designs are fully functional by simulating motion.
- To design and test their Starlink satellite internet constellation.
- To optimize the design and testing of its Starship spacecraft.

Lockheed Martin

- On NASA's OSIRIS-REx Mission
- NVIDIA to Build Digital Twin of Current Global Weather Conditions for NOAA

Sierra Space

- Siemens Xcelerator platform and its DT technology to support its Dream Chaser

Use Cases

Data to be exchanged

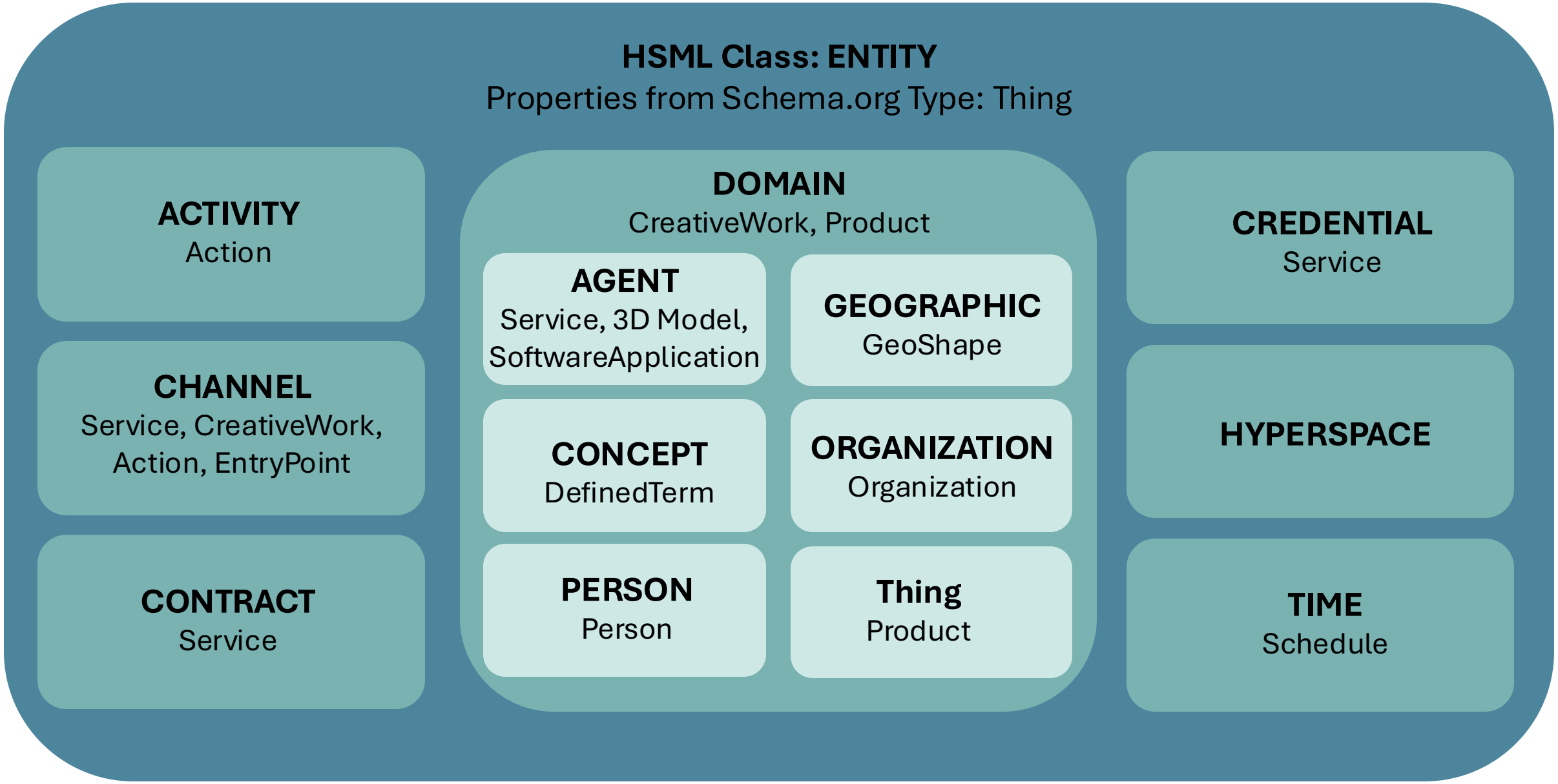
- Position/rotation
- Speed
- Sun location
- Signal frequency
- Signal strength
- Sensor data
- Status of instruments
- Satellite Telemetry data (Satellite's health, Status of instruments, payload data)

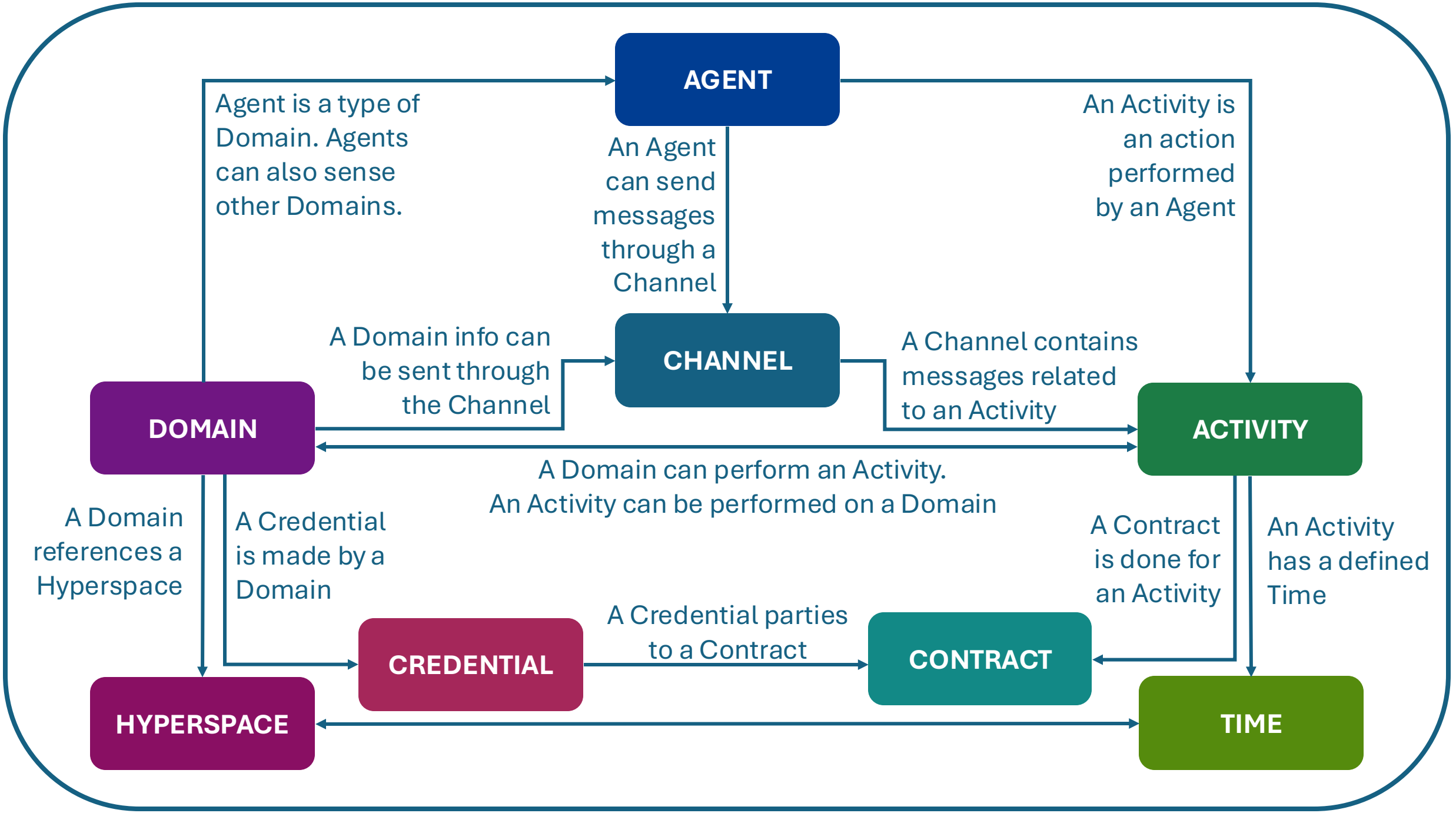
Can our protocol cover it?

Simulation tools used for DT

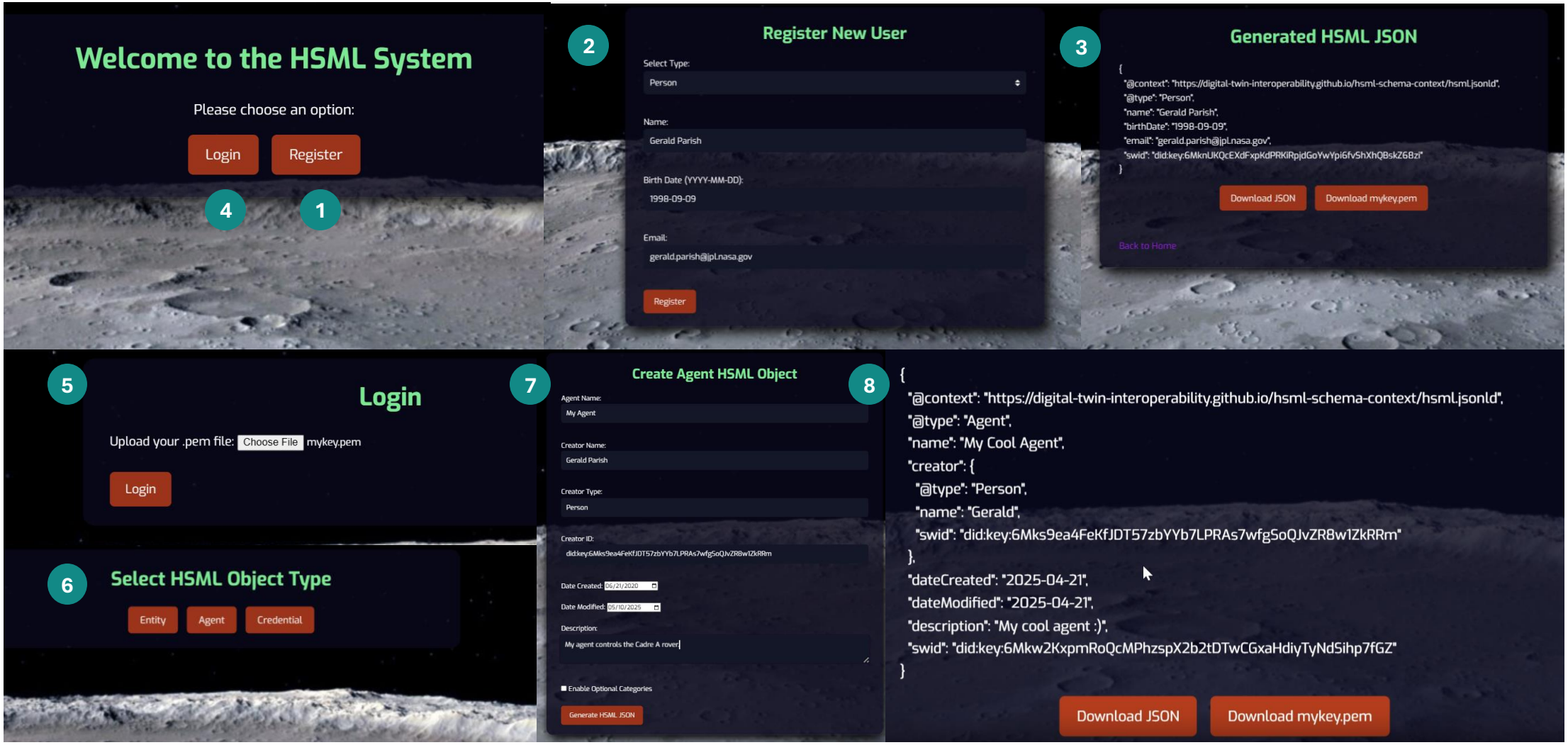
- Unity
- Unreal Engine
- Omniverse
- DARTS (JPL)
- Siemens Nx, Teamcenter
- …

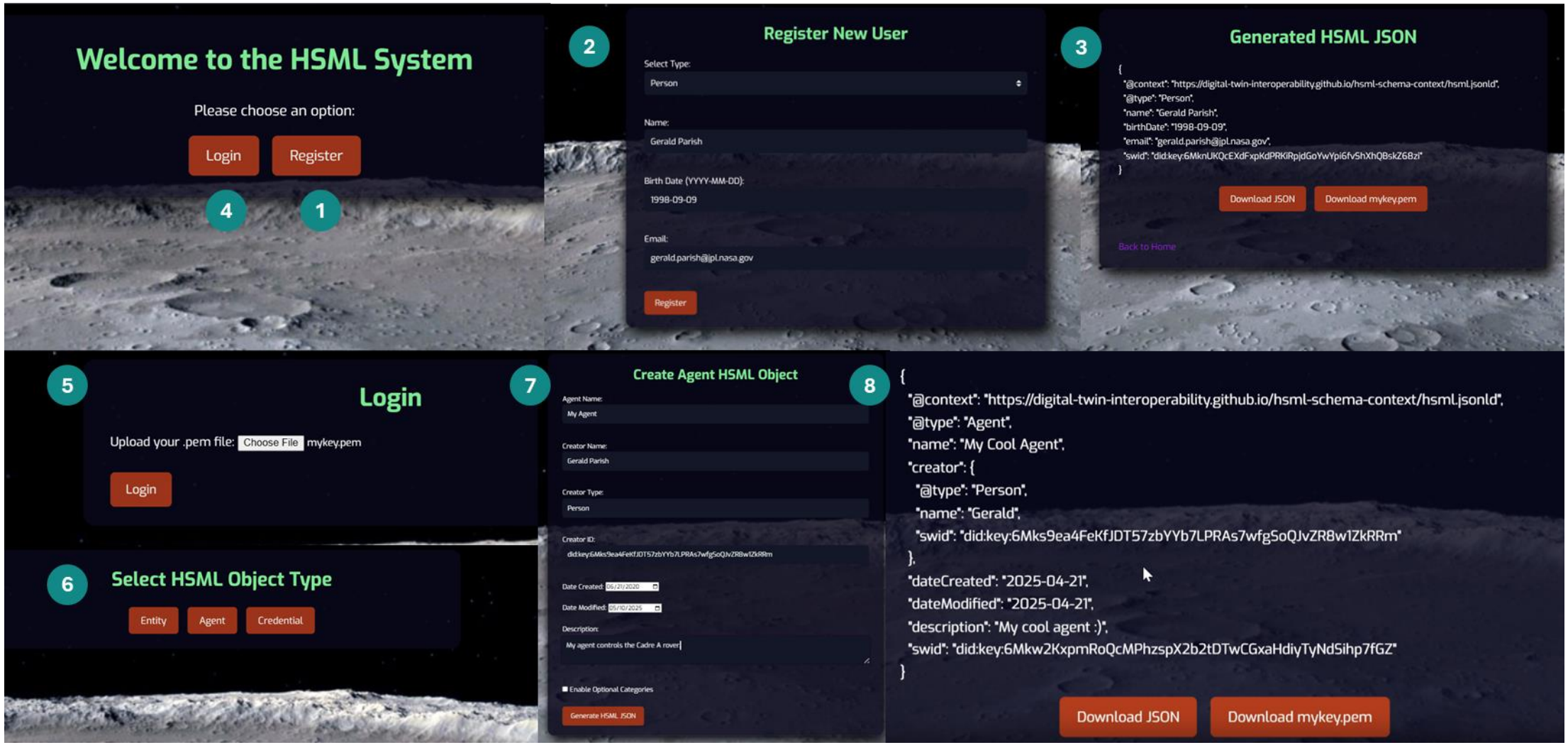
How do we interface with those?


HSML Class: ENTITY
Properties from Schema.org Type: Thing
ACTIVITY
Action
CHANNEL
Service, CreativeWork, Action, EntryPoint
CONTRACT
Service
DOMAIN
CreativeWork, Product
AGENT
Service, 3D Model, SoftwareApplication
GEOGRAPHIC
GeoShape
CONCEPT
DefinedTerm
ORGANIZATION
Organization
PERSON
Person
Thing
Product
CREDENTIAL
Service
HYPERSPACE
TIME
Schedule

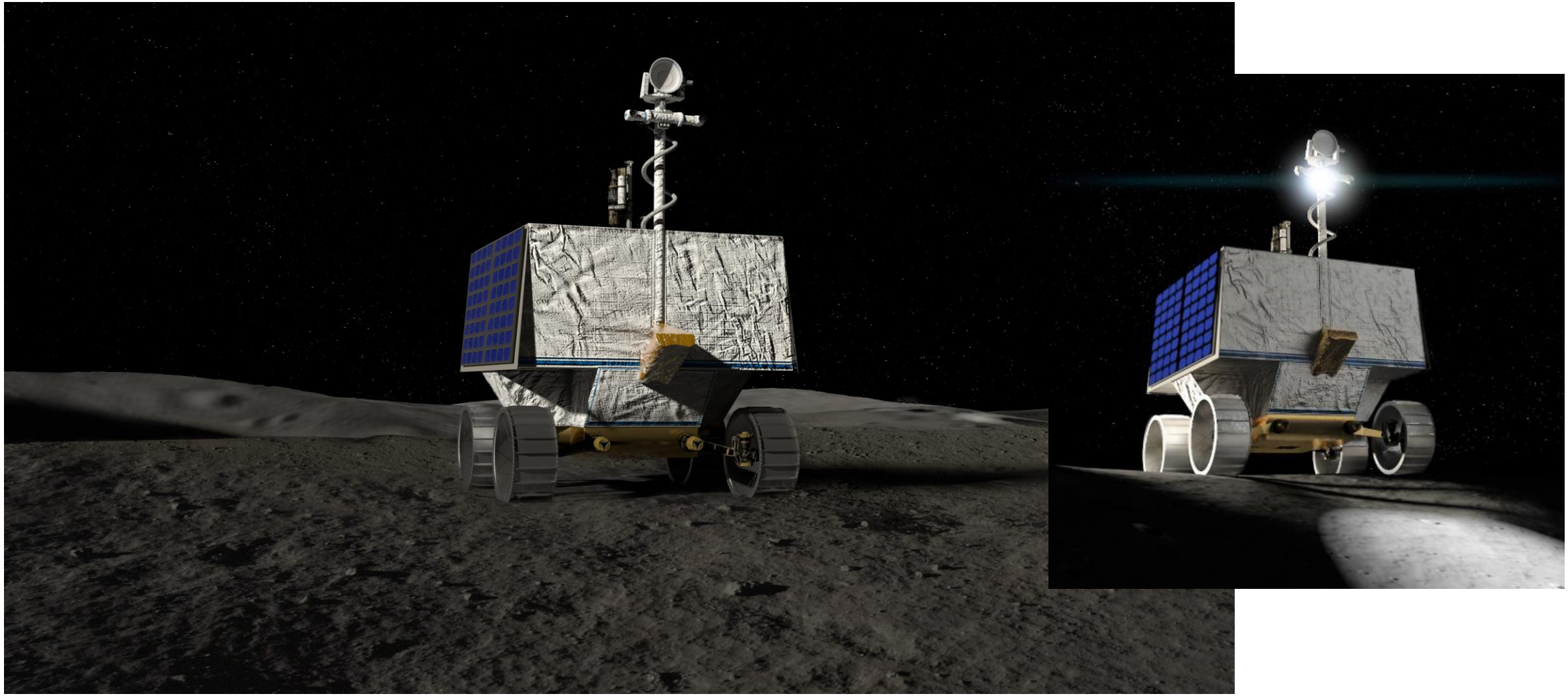

AGENT
Agent is a type of Domain. Agents can also sense other Domains.
An Agent can send messages through a Channel
An Activity is an action performed by an Agent
A Domain info can be sent through the Channel
CHANNEL
A Channel contains messages related to an Activity
DOMAIN
ACTIVITY
A Domain can perform an Activity.
An Activity can be performed on a Domain
A Domain references a Hyperspace
A Credential is made by a Domain
A Contract is done for an Activity
An Activity has a defined Time
A Credential parties to a Contract
CREDENTIAL
CONTRACT
HYPERSPACE
TIME

Methodology – Registration using the Web App

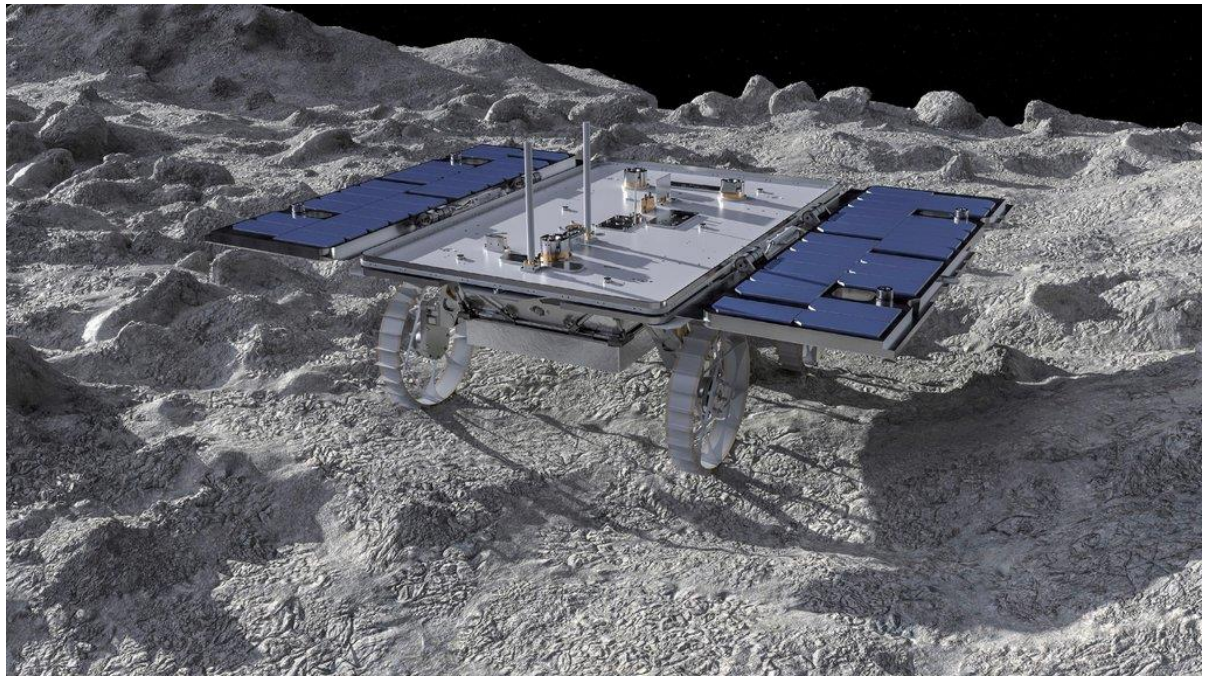


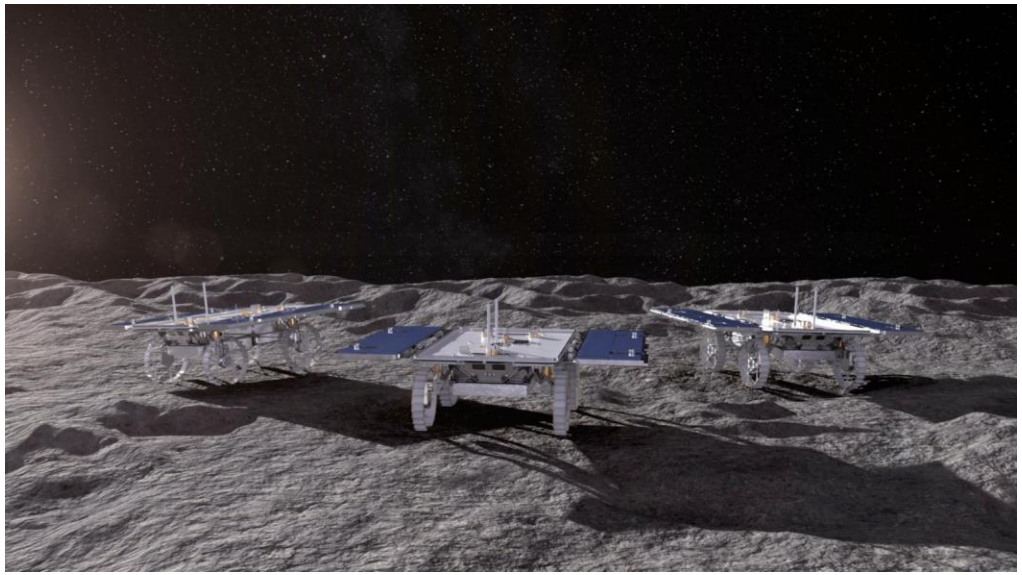

Welcome to the HSML System
Please choose an option:
Login
Register
4
1
2
Register New User
Select Type:
Person
Name:
Gerald Parish
Birth Date (YYYY-MM-DD):
1998-09-09
Email:
gerald.parish@jpl.nasa.gov
Register
3
Generated HSML JSON
{
"@context": "https://digital-twin-interoperability.github.io/hsml-schema-context/hsml.jsonld",
"@type": "Person",
"name": "Gerald Parish",
"birthDate": "1998-09-09",
"email": "gerald.parish@jpl.nasa.gov",
"swid": "did:key:6MknUKQcEXdFxpKdPRKiRpjdGoYwYpi6fv5hXhQBskZ6Bzi"
}
Download JSON
Download mykey.pem
Back to Home
5
Login
Upload your .pem file:
Choose File
mykey.pem
Login
6
Select HSML Object Type
Entity
Agent
Credential
7
Create Agent HSML Object
Agent Name:
My Agent
Creator Name:
Gerald Parish
Creator Type:
Person
Creator ID:
did:key:6Mks9ea4FeKfJDT57zbYYb7LPRAs7wfgSoQJvZR8w1ZkRRm
Date Created:
06/21/2020
Date Modified:
05/10/2025
Description:
My agent controls the Cadre A rover
Enable Optional Categories
Generate HSML JSON
8
{
"@context": "https://digital-twin-interoperability.github.io/hsml-schema-context/hsml.jsonld",
"@type": "Agent",
"name": "My Cool Agent",
"creator": {
"@type": "Person",
"name": "Gerald",
"swid": "did:key:6Mks9ea4FeKfJDT57zbYYb7LPRAs7wfgSoQJvZR8w1ZkRRm"
},
"dateCreated": "2025-04-21",
"dateModified": "2025-04-21",
"description": "My cool agent :)",
"swid": "did:key:6Mkw2KxpmRoQcMPhzspX2b2tDTwCGxaHdiyTyNdSihp7fGZ"
}
Download JSON
Download mykey.pem

Artemis NASA



An illustration of NASA’s Volatiles Investigating Polar Exploration Rover, or VIPER.
Credit: NASA/Ames Research Center/Daniel Rutter





CADRE Rover on the Moon (Artist's Concept)
This artist's concept depicts a small rover – part of NASA's CADRE (Cooperative Autonomous Distributed Robotic Exploration) technology demonstration headed for the Moon – on the lunar sur... Credit: Motiv Space Systems