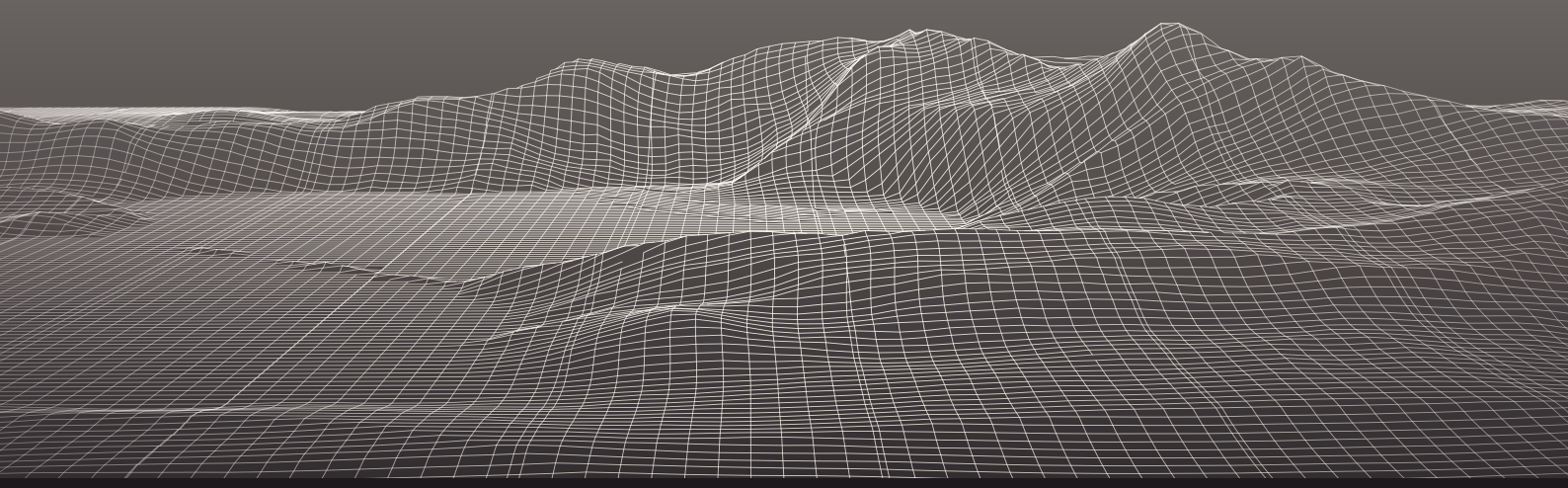
SPATIAL WEB
FOUNDATION

SWF STD-6:2025 (0.1)

The Spatial Web

Universal Domain Graph (UDG) Design Specification

November 07, 2025



Contents

List of tables

List of figures

Spatial Web Foundation
5877 Obama Blvd
Los Angeles, CA 90016
USA

E-mail: info@spatialwebfoundation.org

Abstract

The Universal Domain Graph (UDG) is a medium for collective intelligence in the Spatial Web. This specification defines how the UDG supports collective intelligence and other functions through a set of conceptual designs. Requirements arising from the designs are identified as applicable to the Spatial Web components. By this functional analysis, design synthesis and component requirements, this specification contributes to the overal engineering of the Spatial Web.

1. Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEEE Std 2874-2025, .

IEEE Std 7007™-2021, Institute of Electrical and Electronics Engineers. *IEEE Ontological Standard for Ethically Driven Robotics and Automation Systems*. 2021. https://ieeexplore.ieee.org/document/9611206.

[52] Uber Technologies, Inc. *H3: Hexagonal hierarchical geospatial indexing system*. 2024. https://h3geo.org.

W3C WD-rdf12-concepts-20251104, HARTIG, Olaf, Pierre-Antoine CHAMPIN and Andy SEABORNE (eds.). *RDF 1.2 Concepts and Abstract Data Model*. 2025. World Wide Web Consortium. https://www.w3.org/TR/2025/WD-rdf12-concepts-20251104/.

W3C did-core, World Wide Web Consortium. *Decentralized Identifiers (DIDs) v1.0*. https://www.w3.org/TR/did-core/.

W3C vc-data-model-1.1, World Wide Web Consortium. *Verifiable Credentials Data Model v1.1*. https://www.w3.org/TR/vc-data-model-1.1/.

2. Terms, definitions and abbreviated terms

For the purposes of this document, the following terms and definitions apply.

2.1. Terms and definitions

2.1.1. artificial intelligence
AI PREFERRED

The capacity of computers or other machines to exhibit or simulate intelligent behavior.

SOURCE:

2.1.2. distributed computing PREFERRED

Model of computing in which a set of nodes coordinates its activities by means of digital messages passed between the nodes.

SOURCE: ISO/IEC TR 23188:2020, Clause 3.1.1

2.1.3. dimension PREFERRED

A kind of measurable extent.

EXAMPLE: Length, breadth, depth, or height are examples of dimension.

2.1.4. DOMAIN PREFERRED
SEE ALSO: ***ENTITY***
SOURCE: IEEE Std 2874-2025, Clause 3
(ENTITY)

Sphere of knowledge, influence, or ACTIVITY.

2.1.5. domain authority PREFERRED

An ENTITY that is CREDENTIALED to have the ability to define within a DOMAIN the norms and terms under which CONTRACTS are created governing: AGENTS, ACTIVITIES and CREDENTIALS within that DOMAIN.

SOURCE: IEEE Std 2874-2025, Clause 3

2.1.6. ENTITY PREFERRED

that which is perceived, known, or inferred to exist, has existed, or is anticipated to exist.

Note 1 to entry: An ENTITY can exist whether data about it are available or not.

Note 2 to entry: ENTITY is the base item in the Spatial Web Ontology.

EXAMPLE: An event, idea, object, person, process, etc.

SOURCE: IEEE Std 2874-2025, Clause 3

2.1.7. Euclidean space PREFERRED

Real vector space or real point space for which a scalar product is defined for any two vectors.

Note 1 to entry: The usual geometrical three-dimensional space is a Euclidean point space. Four-dimensional vectors used in special relativity are elements of a non-Euclidean point space because the scalar product of a vector by itself may be negative.

Note 2 to entry: Euclidean space is a type of vector space.

SOURCE: , Clause 02-03-19, modified — Note 1 has been modified. Note 2 has been added.

2.1.8. Generalized worlds PREFERRED

ENTITIES embedded in a vector class of HYPERSPACE using a generalized coordinate reference system to define location in a high dimensional space.

Note 1 to entry: Generalized worlds are a generalization of geographic information referenced to a geographic coordinate reference system.

Note 2 to entry: see also World2Vec

Note 3 to entry: the term “generalized worlds” may be changed, but the concept as defined in [generalized-worlds] needs to be defined and given a stable term.

2.1.9. geographic coordinate reference system PREFERRED

Coordinate reference system that has a geodetic reference frame and an ellipsoidal coordinate system.

SOURCE: ISO 19111:2019, Clause 3.1.35

2.1.10. graph space PREFERRED

A set of nodes and edges such that every edge has both a source node and a target node.

Note 1 to entry: Graph space is a type of *HYPERSPACE* (2.1.11).

2.1.11. HYPERSPACE
SPACE PREFERRED

A set of 'points' such that, for every ordered pair of points, there is a possibly empty set of 'paths' from the first point to the second, and such that there is an 'identity' path from every point to itself.

2.1.12. Hyperspace Modelling Language
HSML PREFERRED

A human- and machine-readable modeling language and semantic data ontology schema that describes objects, relations, actions, activities and their permissions.

Note 1 to entry: Uses of *HSML* (2.1.12) include the expression and automated execution of legal, financial, and physical activities.

SOURCE: IEEE Std 2874-2025, Clause 3

2.1.13. Hyperspace Transaction Protocol
HSTP PREFERRED

Generic protocol designed to underwrite automated contracting transactions that are required for building a coherent, decentralized, secure, and privacy-respecting *Spatial Web* (2.1.17).

SOURCE: IEEE Std 2874-2025, Clause 3

2.1.14. register PREFERRED

Set of files containing identifiers assigned to items with descriptions of the associated items

SOURCE: ISO 19135-1:2015, Clause 4.1.9

2.1.15. relationship PREFERRED

Links, portals, membership, situations, paths

2.1.16. representation PREFERRED

Model used to describe some aspect of a world.

Note 1 to entry: Representations can be passive or active. A passive representation reflects a possible state of affairs, typically the state of a spatio-temporal local slice of an AGENT'S environment, while an active representation changes the state of a system to conform to the representation.

Note 2 to entry: Representations can be probabilistic such as conditional probability distributions or Bayesian beliefs. They can describe either state of the world that generates data or content, at a particular moment in time, or the contingencies and dynamics that could depend upon the actions of a user.

2.1.17. Spatial Web PREFERRED

Network structures supported by the protocol, architecture, and governance related to *HSTP* (2.1.13) and *HSML* (2.1.12) and their supporting components.

SOURCE: IEEE Std 2874-2025, Clause 3

2.1.18. spatial web identifier
SWID PREFERRED

A decentralized identifier used to uniquely identify an ENTITY, DOMAIN, or other item within the *Spatial Web* (2.1.17).

2.1.19. Spatial Web Domain Server PREFERRED

Networked server that provides location-based identification of HYPERSPACE assets and ownership information to Spatial Web Domains.

Note 1 to entry: A Spatial Web Domain Server handles spatial content retrieval and spatial transaction validation, whereby the rules for particular areas can be described as spatial permissions in their associated assets.

SOURCE: IEEE Std 2874-2025, Clause 3

2.1.20. Spatial Web Node PREFERRED

Computing machine connected to the internet, capable of exchanging *HSTP* (2.1.13) messages.

SOURCE: IEEE Std 2874-2025, Clause 3

2.1.21. universal domain graph
UDG PREFERRED

A distributed hypergraph which contains all relationships between all known *SWIDs* (2.1.18) in the Spatial Web.

Note 1 to entry: The universal domain graph is the complete *Spatial Web* (2.1.17) hypergraph.

SOURCE: IEEE Std 2874-2025, Clause 3

2.1.22. distributed universal domain graph system
distributed UDG system PREFERRED

The set of coordinated Spatial Web Nodes that provide services on the distributed UDG.

SOURCE: IEEE Std 2874-2025, Clause 3

2.1.23. UDG entity graph PREFERRED

All instances of SW entities and connections represented in *HSML* (2.1.12)

2.1.24. UDG node graph: PREFERRED

All distributed computing *Spatial Web Nodes* (2.1.20) and connections using *HSTP* (2.1.13)

2.1.25. UDG Node PREFERRED

A type of distributed computing *Spatial Web Node* (2.1.20) that performs UDG functions

2.1.26. vector space PREFERRED

Type of space composed of *dimensions* (2.1.3) where each *dimension* (2.1.3) is the set of real numbers.

Note 1 to entry: Vector space is a type of *HYPERSPACE* (2.1.11), and *Euclidean space* (2.1.7) is a type of vector space.

SOURCE: IEEE Std 2874-2025, Clause 3

2.2. Abbreviated terms

AI	artificial intelligence
API	application programming interface
AR	augmented reality
CRS	coordinate reference system
DGGS	Discrete Global Grid System
DLT	distributed ledger technology
IoT	Internet of Things
IRI	Internationalized Resource Identifier
RDF	Resource Description Framework
SHACL	Shapes Constraint Language
SPARQL	SPARQL Protocol and RDF Query Language
UDG	Universal Domain Graph
W3C	World Wide Web Consortium
XR	collective reference to both AR and VR

3. Overview

3.1. Scope

This Spatial Web UDG System Design Specification defines the functions performed by the UDG, the analysis of those functions, and the allocation of requirements to the Spatial Web components including HSML, HSTP, and several Spatial Web Nodes.

NOTE In IEEE 2874-2025 this document was titled the UDG Implmentation Specification.

3.2. Purpose

The purpose of this specification is to define a design of the Spatial Web UDG. The UDG design defines requirements for the Spatial Web implementation specifications and guides domain-specific Spatial Web architecture development.

3.3. Structure of document

The report addresses these themes:

- Identifies requirements and fundamental design questions
- Provides an assessment of relevant, existing technology and research
- Defines UDG system design, including foundations, conceptual models and distributed computing

This document is structured as an architecture description consistent with the approach defined in System architecture description.

This document provides these architecture views that define the UDG system design:

- Enterprise viewpoint
- UDG as a knowledge graph
- UDG hyperspace atlas
- UDG as a registry of registries
- UDG as a social network
- UDG distributed computing

3.4. Conventions used in this standard

In this document the following Spatial Web ontology entries (ENTITY, ACTIVITY, AGENT, CONTRACT, CHANNEL, CREDENTIAL, DOMAIN, HYPERSPACE, and TIME) are represented using uppercase.

4. Enterprise view

The UDG is a publicly accessible information commons that serves as a key infrastructure component of the Spatial Web. The UDG is a medium for collective intelligence in the Spatial Web. As a data commons it provides public access to information about the world. The information in the UDG allows for a previously unavailable representation of the physical and conceptual aspects of existence. The UDG will be populated by observations from sensors and semantic statements from cognitive agents. The UDG provides the solution for the grounding problem faced by Artificial Intelligence Agents. Improved reasoning by both artificial and natural intelligence will result in improved decisions for humanity's future.

The UDG provides an open, accessible representation of all ENTITIES. It enables efficient and accurate modeling and analysis of domain relationships. It allows for the organization of domains into an architecture that maintains intra-relational and inter-

relational domain coherence, allowing for the coordination and integration of domain-specific data and content. The UDG allows for nested, hierarchical and heterarchical searches and transformations within and across domains.

The UDG leverages the following concepts:

- UDG Knowledge Graph (KG) is the hypergraph containing all relationships between all known Spatial Web conceptual ENTITIES. The UDG KG is composed of nodes and links where the nodes are ENTITIES and the links are relations between the ENTITIES. Each Spatial Web ENTITY has a Spatial Web Identifier (SWID). The UDG KG is defined in Clause 5
- UDG Hyperspace is the representation of UDG relationship using the several types of Spatial Web Hyperspace including Graphs, Cellular spaces with cell identifiers, and Vector spaces with coordinates. The different types of Hyperspace allow for different methods for navigating and discoverying ENTITIES. These concpets along with the relationshps between UDG in differing hyperspaces are defined in Clause 6
- UDG Registries are a coordinated set of hierarchically organized registeries of DOMAINS. Top-level DOMAINs are registered in the Spatial Web Designated Authority Registry. Top-level domains may link to a domain-specific registry. The sysgtem of UDG Registries are defined in [section-conceptual-registry]
- UDG Social Networks is the communication between ENTITIES in the UDG. In particular AGENTS will gather information about the state of the environment from other ENTITIES in the UDG. AGENTS will exchange information with other AGENTS to form beliefs about the environment. Neighborhoods in the UDG will affect the social engagement between AGENTS. This communication between AGENTS is a social epistemological process leading to collective intelligence as described in the [section-conceputal-view-social-network]
- Distributed UDG System is the set of coordinated Spatial Web Nodes that provide services on the distributed UDG. A Spatial Web Node is a computing machine connected to the internet, capable of exchanging HSTP messages. Spatial Web Nodes particular to the UDG include: UDG Nodes and Registry Nodes. The UDG Node peforms functions related to the distributed communications needed for UDG functions, e.g., state propogation and distributed queries. UDG Registry Nodes implement UDG Registry functions. These topics are discussed in the Clause 9

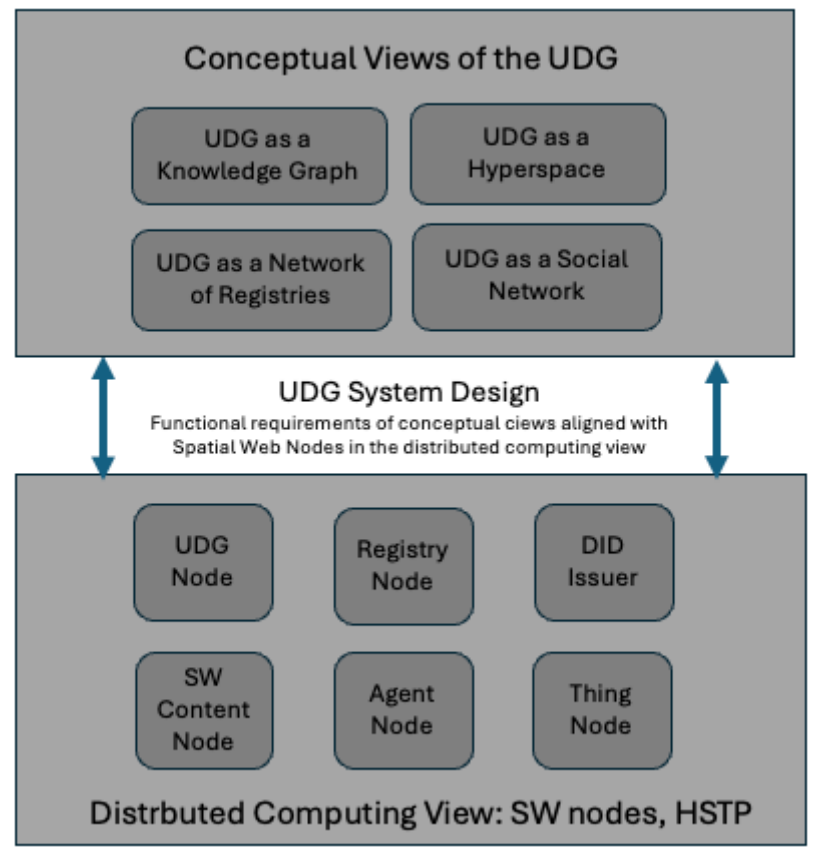
The UDG conceptual structures (KG, Hyperspace, Registries, Social network) are implemented in the Spatial Web Nodes of the distributed UDG System. There is a non-isomorphic relationship between the conceptual domain hierarchies and the SW Nodes. Presence of a Domain in a specific SW node may be dynamic and transient. This relationship between the conceptual views and the Distributed Computing view is represented in Figure 1

Each viewpoint of the UDG includes requirements that are specific to Spatial Web Components. Implementations of the Spatial Web UDG design are achieved by allocating requirements across those components. The [annex-requirements-allocation] gathers the requirements for each component. The Spatial Web Components used in this UDG Design specification are:

- Hyperspace Modeling Language (HSML) (SWF STD-02)

- Hyperspatial Transaction Protocol (HSTP) (SWF STD-03)
- SWID Documents (SWF STD-04)
- Spatial Web DID method (SWF STD-05)
- Spatial Web Registry
- UDG Node Design
- Agent Verification plan
- Spatial Web Governance

FIGURE 1: UDG Enterprise Viewpoint



SW Nodes host and exchange the content defined in the conceptual viewpoints of the is design specification. The table <sw-node-conceptual-content> provides a summary of UDG conceptual content held by the SW Nodes

TABLE 1: UDG functional content deployed in physical SW Nodes

SW Node	UDG KG	Hyperspace	Registries	Social
SW Content Node	Local ENTITIES	Embedding of local entities	DOMAIN Membership Credential	
UDG Node	Extended dynamic graph of ENTITIES	Hyperworlds and Atlases	Registry Relationships	Collective intelligence
Registry Node	Issues membership credential		DOMAIN Membership register	
Agent Node	Dynamic SITUATONS	Model uses Hyperworlds	DOMAIN Membership Credential	Social epistemology
Thing Node	Sensing and Actuation of environment	Location in cyber-physical space	DOMAIN Membership Credential	Observations of physical world
DID issuer	SWID creation and resolution		SWID creation and resolution	

5. UDG as a knowledge graph

5.1. Knowledge Graphs

This clause considers the UDG as a knowledge graph (KG). A KG is a representation of information that uses a graph-structured data model to represent and operate on data [49].

The UDG KG is a graph where the nodes are Spatial Web ENTITIES. Interlinking of ENTITIES in the UDG is accomplished using Links, portals, membership, situations, paths (see 5.3).

Knowledge in the Spatial Web will include statements of fact as well as statements with probablistic uncertainty. At any given time contradictory statements will exist unresovled across the UDG KG. Statements of knowledge will change through the dialectic activity of social agents as defined in Clause 8.

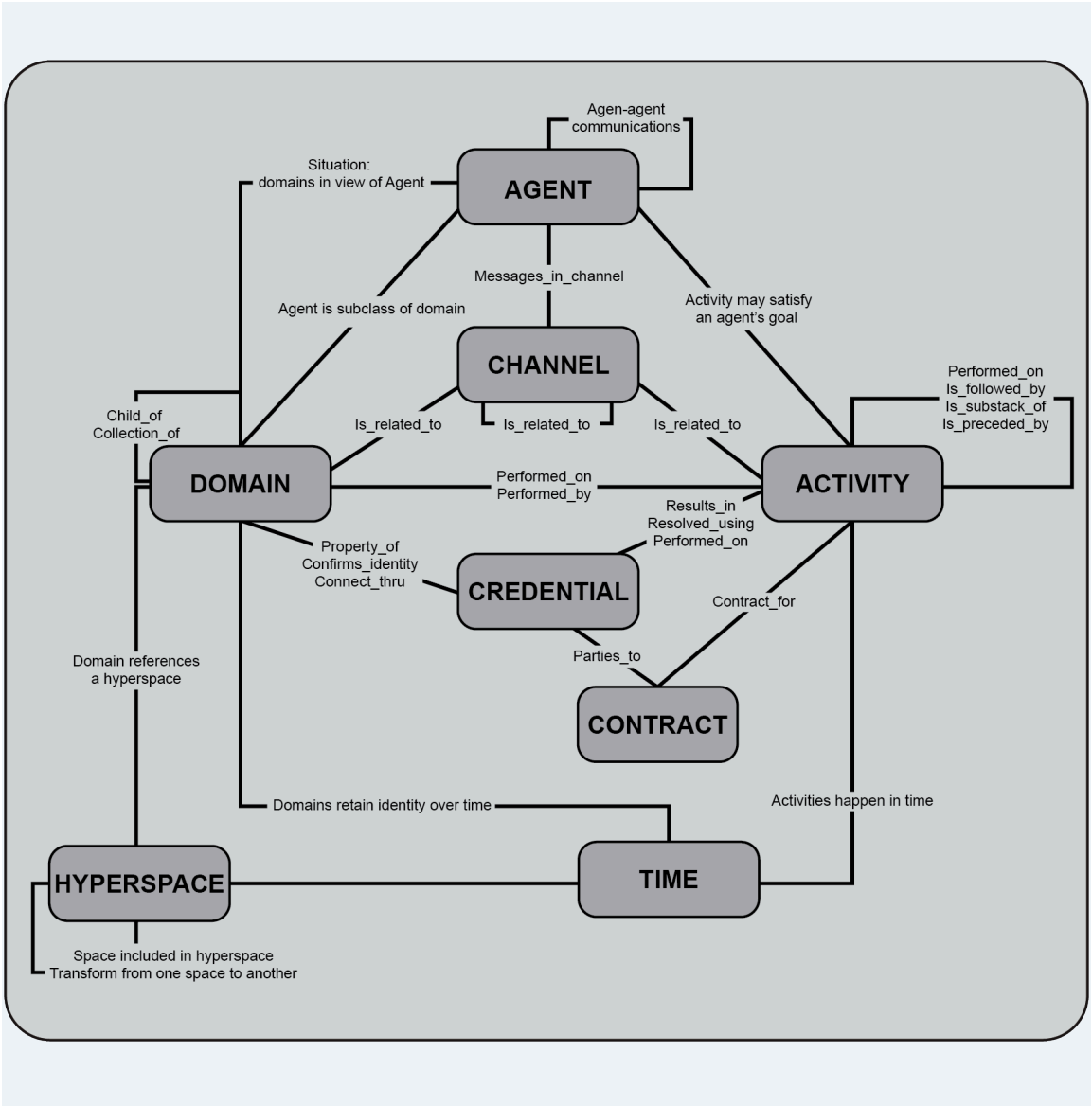
Messages between nodes in the UDG KG comply with HSML specification. The HSML builds upon the W3C Resource Description Framework (RDF) suite of standards. RDF graphs are sets of subject-predicate-object triples, where the elements may be IRIs, blank nodes, datatyped literals, or triple terms.

5.2. DOMAINS are the primary ENTITY in the UDG

5.2.1. ENTITIES are the basic type for all knowledge in the UDG

The UDG KG is composed of ENTITIES. An ENTITY is that which is perceived, known, or inferred to exist, has existed, or is anticipated to exist. ENTITIES are of several types as shown in the [hsml_ontology], which are the primary concepts used across the Spatial Web and in the Hyperspace Modeling Language (HSML). HSML implements the Spatial Web ontology as a set of schemas that enable increased coherence across diverse datasets without sacrificing flexibility.

FIGURE 2: Spatial Web ontology



It is certain that there will be inconsistencies in the UDG, similar to inconsistencies in the WWW. Some inconsistencies will be from relationships between DOMAINS. Other inconsistencies will be the result of the content of DOMAINS in the UDG. UDG operations must be resilient to inconsistencies in the relationships between nodes and the content of nodes.

5.2.2. Definition of DOMAIN

DOMAIN is the fundamental class within the UDG for describing the various things, agents, and locations that make up the Spatial Web. Every structure, building, planet, character, animal, mystical alien pyramid or similar construct, is ultimately a DOMAIN.

A DOMAIN is an ENTITY that may identify the context for an ACTIVITY for or by an AGENT at a specific place acting upon various things within that DOMAIN. Everything that happens within the UDG happens within a DOMAIN.

A DOMAIN is required to have these attributes:

- SWID a decentralized identifier used to uniquely identify DOMAINS within the UDG.
- DOMAIN Type indicates the particular role that the domain plays. DOMAIN may be characterized by multiple different ontologies, but it *must* have DOMAIN type.

A DOMAIN may have these other attributes:

- Location identifies the entity's position in HYPERSPACE, relative to either the containing DOMAIN, or to an externally defined DOMAIN. If a domain exists solely as a vehicle for containment, then location is not specifically required.
- ACTIVITIES provides operational instructions for the ENTITY in question.
- style resources provides guidance for rendering the DOMAIN with different content types.
- link to an external DOMAIN. Some ENTITIES provide a link that can be activated through certain interfaces or affordances. This is a simple link, meaning it changes the context (DOMAIN) of the agent initiating the link without specific constraints, which is analogous to an HTTP standard hyperlink. More information about links will be covered in the subsection Links.

5.2.3. DOMAIN types

A DOMAIN type is a specialization of a DOMAIN that incorporates additional capabilities. [IEEE_Std_2874] defines several DOMAIN types:

TABLE 2: Domain types based on their defining characteristics

Type of Domain	Description
Geographic	Implicitly or explicitly associated with a location
Concept	Intangible concepts and abstract ideas shared by a community of users
Organization	Pertaining to membership within an entity
Agent	Individual domains with active states and agency
Person	Special subtype of agent maintaining a self-sovereign identity

Type of Domain	Description
Thing	Bounded items without agency

5.2.4. DOMAIN modeling checklist

This checklist is provided to aid in modeling a DOMAIN to be included in the UDG KG.

- *Where?*. Where does the action take place?
- *How Is Space Defined?* What is the relevant hyperspace of relevance within the system.
- *How Is the DOMAIN Connected?*. How are ENTITIES within the DOMAIN connected, and how are DOMAINS connected to other DOMAINS?
- *What Happened?*. Is it possible to retrieve a history of what happened within the domain over time?
- *What Kind?*. What classifications apply to the domain?
- *Who?*. Which agents are participating within the domain?
- *When?*. When does the activity take place within the context of the domain. This becomes especially critical for asynchronous events.
- *What Happened?*. Is it possible to retrieve a history of what happened within the domain over time?
- *How?*. What are the activities that can be accomplished within the domain, and how are these activated?
- *Why?*. What are the goals or purposes of the domain, and what happens when those goals are achieved?

DOMAINS may be references using hyperspace including time. DOMAINS usually have a specific start time (or other condition) and end time (or other condition).

A DOMAIN may have a schematic representation that can be extended from a core DOMAIN type. The core identifies the relevant properties for the DOMAIN beyond the properties of the DOMAIN base class.

5.2.5. Extending Entities

While the Spatial Web makes use of the DOMAIN Types defined in [IEEE_Std_2874], those types may need to be extended to model the complexity of all use cases. A framework is defined that can identify nodes contextually using topics and state dependencies. If a use case needs to add a property, the property can be added using SHACL.

FIGURE 3

```
[] a hsml:Domain ;
    hsml:hasShape [
        a sh:NodeShape ;
```

```
            sh:targetClass hsml:Place ;
            sh:property [
                a sh:PropertyShape ;
                sh:path ex:population ;
                sh:nodeKind sh:Literal ;
                sh:datatype xsd:nonNegativeInteger;
                sh:minOccurs 0 ;
                sh:maxOccurs 1 ;
            ],[
                a sh:PropertyShape ;
                sh:path hsml:hasTopic ;
                sh:nodeKind sh:IRI ;
                sh:class hsml:Topic ;
                sh:value <#concept/Country> ;
            ]
        ] .
```

As illustrated below, the DOMAIN holds the shape definitions via the `hsml:hasShape` property, and when the DOMAIN is instantiated, this provides information about how the given property or properties are implemented.

In the example, the place (a country) is defined with a property `ex:population` as well as a second property `hsml:hasTopic`. The first is considered valid if it has a nonNegative integer (and is an optional parameter), the second is considered valid if the hsml:hasTopic property has the value <concept/Country>. If either of these are not true for the place, then the structure generates an error for the shape.

Within the graph, then, this would be applied to the Canada place node as follows:

FIGURE 4

```
[] a hsml:Place ;
    hsml:swid did:swid:0CANADA ;
    hsml:swurl <#country/Canada> ;
    hsml:hasTopic <#concept/Country> ;
    ex:population 32159219 ;
    .
```

This makes it possible to add any number of properties to the DOMAIN, as well as to set constraints that more accurately specify things such as topicality or state configurations.

5.2.6. SWIDS and Aliases

5.2.6.1. SWIDs

All ENTITIES have a Spatial Web Identifier (SWID) which may:

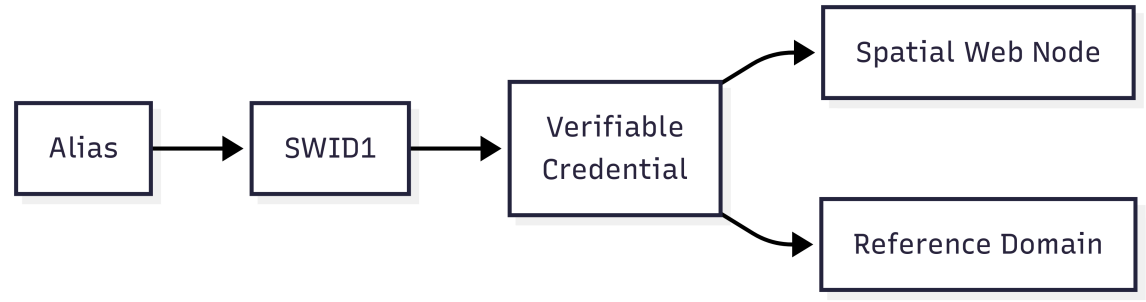
- Serve as decentralized identifier (DID) for digital identity devoid of any type of centralization (see W3C did-core)

- Identify the ENTITY uniquely within the Spatial Web. In this regard, the SWID acts as an (indirect) IRI.
- Establish links between ENTITIES.
- Resolve connections between different nodes in the UDG KG. The nodelink document identifies both authentication and address information for different nodes within the spatial web.

5.2.6.2. Aliases

In the Spatial Web, one or more aliases can be assigned to a SWID that is intended to locate a resource, but the address for the resource is contained within the SWID Document. The alias is analogous to an HTTP domain name mapping to an IPV6 address, but in this case, the "domain name" alias maps to a SWID.

FIGURE 5: SWID Alias



EDITORIAL NOTE

The specific form of aliases is still to be determined. To be consistent with other SWF specificatons, the figure may need to be revised, inserting "SWID Document" between SWID1 and "Verifiable Credential."

Internally, all resources within the Spatial Web are identied by SWIDs. However, such SWIDs exist primarily to identify identity and assertion credentials, and are usually not "human readable". In this regard, the SWID is somewhat analogous to an IPV6 address in that it identifies a specific enitity or domain in the network exclusively.

An *alias* on the other hand, is more user-friendly, and can be used to identify entities and domains SWIDs are generated by SWID Nodes directly, but aliases are more like *HTTP domain names*: they put a human readable name on the resource.

The precise mechanism for (and structure of) spatial web aliases has not yet been worked out, though it will likely be similar in scope and structure to ICANN. The specific node authority (the person or organization who owns the node) can claim a "domain name" that can be segmented based upon a common authority. For instance, SWRA might have an alias for Earth of the form:

hstp:org:swra:Place:Earth

which would then be associated with the SWID

did:swid:c124151stqwf98cjjklm

Another organization (such as the aforemention UPNA) may then create an association through their own alias to the SWRA alias:

hstp:gov:upna:Place:Earth Place:isAnalogOf hstp:org:swra:Place:Earth

Aliases can also be applied from different protocols. For instance, UPNA might use an HTTP-based URL as an allias:

http://sw.upna.gov/Place/Earth

This would map to the credential endpoint for the registry, which would then retrieve the credential to the relevant entity as a document. If this was called from a Spatial Web web client, then the retrieval of this credential can be used to determine additional actions, from retrieval of metadata to linking to that place within a given spatial web node. The specific actions are still to be determined.

5.2.7. Requirements and recommendations

TBD

5.3. Relationships between ENTITIES

5.3.1. Complexity of relationships

Spatial Web DOMAINS have a variety of relationships with one another, thereby reflecting the complexity of the real world and the interconnectedness of its many parts. The UDG's ability to represent and navigate arbitrarily complex relationships is central to its role as the foundational component of the Spatial Web and ensures that DOMAIN-specific data is not only coherent within itself but also in coordination with other DOMAINS.

There are several types of relationships between ENITITIES in the UDG. Relationships between ENTITIES are shown in the Figure 2. The following relationships are defined in this clause:

- Links represent relationships between an initiator ENTITY to a receiver ENTITY
- Portals are relationships between locations in two or more DOMAINS, providing a leap between DOMAINS
- Membership is a relationship type that signifies that a DOMAIN is a member of a DOMAIN that is of a higher order or level
- Situations are a set of DOMAINS currently in the context of a DOMAIN
- Dynamic relationships bring in the notion of paths

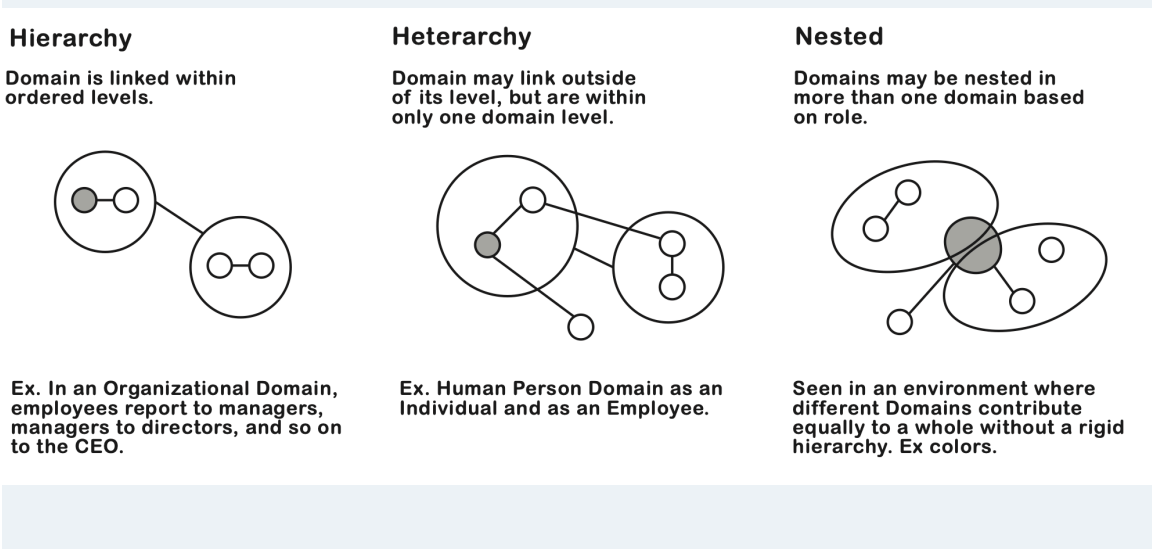
The complexity of the relationships in the UDG can be charaterized using mereology. The approach draws on ideas from mereology (the theory of parthood relations), order theory (the study of hierarchy), and category theory (the mathematics of structure). In

the Spatial Web, DOMAINS may be related in structures of arbitrary complexity. [WHO? WHERE is the clause?] discusses the complexity as hierarchy, heterarchy, and holonic as described in Table 3.

TABLE 3: Mereological relationships in domain hypergraphs

Type of Relationship	Description
Hierarchical	* DOMAINS may be a member of one and only one upper DOMAIN.
	* DOMAINS may not have links to other DOMAINS.
	* Coordination between DOMAINS occurs through the Domain Authority of the DOMAIN in which the DOMAIN is a member.
Heterarchical	* DOMAINS may be members of one and only one upper DOMAIN.
	* DOMAINS may have links to other DOMAINS.
	* Coordination between DOMAINS occurs through the Domain Authority of the upper DOMAIN and through relationships to other DOMAINS.
Holonic	* DOMAINs may be members of multiple upper DOMAINS.
	* DOMAINS may have links to other DOMAINS.
	* Coordination between DOMAINS occurs through the Domain Authority of the upper DOMAIN and through relationships to other DOMAINS.

FIGURE 6: Mereological relationships in domain hypergraphs

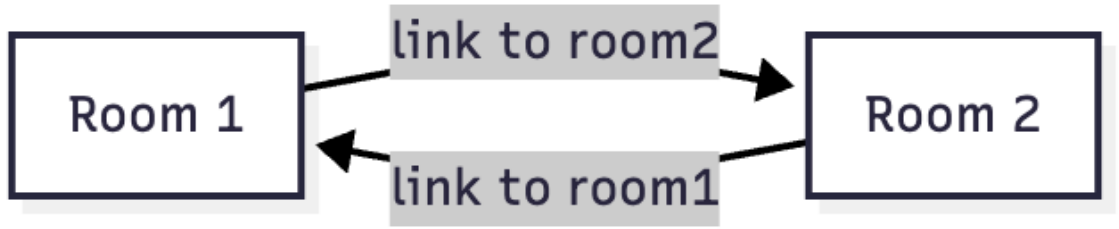


5.3.2. Links between ENTITIES

5.3.2.1. Links are relationships between ENTITIES.

A link is a relationship from an initiator ENTITY to a receiver ENTITY. The link is part of the initiator ENTITY and indicates tat the initiator ENTITY seeks to point to the receiver ENTITY. There are no bidirectional links within the Spatial Web. All links are unidirectional (one way), but there may be analogous entities (portals) for each link that provide the corresponding link back to the original place or entity.

FIGURE 7: Bidirectional links between rooms



In HSML, using RDF, a link is a subject-predicate-object triple, where the elements may be IRIs, blank nodes, datatyped literals, or triple terms.

A link implies the following information between domains:

- initiator of the link
- recipient of the link
- target of the link
- type of link (if known)
- time and location of the link
- periodicity
- duration

A link can be set to be *inactive and/or hidden_*.

An inactive link is visible, but can't be activated. Inactive links may serve the purpose of being descriptive (in a way similar to an inactive option in a select control in HTML works) or may be a divider.

Hidden Links are links that cause their containing entity to be invisible until a specific condition is met in the environment (such as the agent finding a magic scroll or having a certain power level).

It's also possible to have *nested links*. *Nested Links_* are links that point to other links

Types of links are described in the remainder of this clause

EDITORIAL NOTE

Comments about the text above: 1. a bullet mentions "target". It's unclear why a target is needed/what it provides when you have an initiator and a receiver. 2. Statement about inactive links isn't correctly stated. Statements that follow are better (if "link" is really going to be used): "A link may be active or inactive. A link may be visible or hidden."

EDITORIAL NOTE

the link types described below need to be reviewed.

5.3.2.2. Persistent links

Links to DOMAINS present in the DOMAIN description. These are persistent links, in that they are always available to the DOMAIN.

5.3.2.3. Link Challenges and Keys

A link may be associated with a challenge query. This query must be satisfied before the link is activated. Think of this as a lock on the link. The agent must have and apply an associated key for that lock before the agent can activate the link. This metaphor is so persistent that it is easy to think of the entity containing the relevant link as a portal or door of some sort.

Note that the hosting entity of the link can query the state of that link, and present a representation of that link's status to the corresponding agent. This is different from the way that HTTP links work.

5.3.2.4. Bag link: AGENT carrying a DOMAIN

One common use case in the Spatial Web occurs when an AGENT (say a truck) acts as a transport for another DOMAIN (such as a package). The package becomes linked to the truck. The carrying capacity for the AGENT domain can be determined individually for that particular location (it may be by weight, by volume, by insurability, or by some even more exotic measure)

Each carried DOMAIN in turn has a specific credential key that can serve as a key to a portal (or other linked agent). These are connected to the carrier agent through a bag link. In effect the carrier can "borrow" the key of the carried item.

A carryable agent in that case can be "picked up" by the carrier agent and thus removed from the location within the active domain into the bag location in the carrier's domain. Even if the carrier moves to a new domain, the carried object stays associated with the carrier's internal domain "bag". The carried agent can be used by the carrier to activate a portal or similar Thing agent.

Activation of a bag link *may* also cause the item to expire, in essence, being removed from the bag upon use. Additionally, a carried item may be transferred to another agent or "dropped" into the current location. That has obvious implications for both supply chain scenarios and e-commerce scenarios, where a specific virtual item is "sold" to another agent, and its use in role playing games should be self-evident.

5.3.2.5. AGENT-to-DOMAIN Links

A use case is connecting one agent that is in effect a camera (a sensor array) with another agent that is a display or monitor.

This could also be used to monitor the value of a given set of properties such as position, temperature, funds, or emotional state. Since in many cases, these values may be computed rather than intrinsic, this provides a light-weight mechanism for determining relevant state without needing to know the internal mechanisms for that agent.

5.3.3. Portals

A *portal* connects spaces as defined in Spatial Web clause 6.2.3.7.2. Connecting spaces. By identifying sets of locations in hyperspace that connect two DOMAINS, a portal between the DOMAINS is created as a relationship between DOMAINS. A portal need not be binary; it could be multiple sets of points to define an n-ary portal.

Portals are defined using sets of points in spaces. The sets of points might also be the locations of several domains. For example, a portal defined by locations on each side of a doorway also defines the connection of room DOMAINS which share the doorway.

A portal may be used by an AGENT to move between spaces. The path of the agent uses the portal to move from one location to another and between domains. Such links are topological, in that such links are not necessarily dependent upon contiguity or geometry.

Portals access may be constrained by the requirement that the initiating agent has access to a cryptographic key in order to activate the portal. Such keys may be associated with dedicated agents in a Bag relationship.

A landing place is a place within a domain that is used to indicate where a given agent is placed (lands) when entering a domain without an explicit link to a place. This can be thought of as the "home" of the domain, and is indicated as a property of the domain. This corresponds roughly with the top of an HTML page when it is rendered.

Portals will take you from a place to another place, but it is possible to link to other entities. Such links will take you to the location of that entity. For instance, if you wanted to join a party (an aggregation), then you could use the SWID of that aggregation to take you to where that party is located, even if that party moves around. See Links for more details.

> *EDITORIAL NOTE*
>
> *The OMA3 Inter-World Portaling System (IWPS) draft standard may be relevant for Spatial Web. IWPS presents a framework for digital interoperability, acting as the 3D equivalent to a web hyperlink, linking users to various virtual worlds. The IWPS standard enables users to move between applications, even on different devices, effectively serving as a generic application launcher.*

5.3.4. Membership in a DOMAIN

5.3.4.1. DOMAIN as a container of DOMAINS

A DOMAIN is a holon: it is both a unit and a composition. As a unit or system, a DOMAIN performs functions that are only achievable as a collective whole. As a composition, a DOMAIN contains parts which are subject to conditions on the parts enforced by the overall DOMAIN. An AGENT is an excellent example of a DOMAIN as a system or organism. In this clause we focus on the DOMAIN as a composition.

Membership in a DOMAIN is type of credential granted by the Domain Authority of the upper domain to the member domain. The credential provides the evidence of the claim that a DOMAIN is a member of an upper DOMAIN.

Membership in DOMAINS may be a multi-layer hierarchy: a DOMAIN maybe included in a DOMAIN which in turn is included in a DOMAIN.

As a holarchy, the membership relationship allows for a DOMAIN to be a member of more than one DOMAIN.

Movement of a DOMAIN in and out of membership in an upper DOMAIN is allowed as specified in the membership agreement of the DOMAIN.

5.3.4.2. Obligations on DOMAIN members.

DOMAINS define norms, obligations and laws which the member DOMAINS must adhere with. Enforcement of the DOMAIN obligations is a function of the Domain Authority that manages the upper DOMAIN.

5.3.4.3. DOMAIN location

A DOMAIN may have a location in hyperspace. The location may be a single point or it may be a more complex geometry, e.g., polygon, sphere, etc.

A DOMAIN location may be within an upper DOMAIN location. The DOMAIN may include sub-domains contained within the DOMAIN location. For instance, a given planet domain may have multiple locations that represent the countries of that planet.

5.3.5. Moving a DOMAIN between DOMAINS

A DOMAIN may be in more than one DOMAIN, where “in” has several meanings based on differing relationships. A DOMAIN may be a member of an upper DOMAIN and thereby holding a certificate of membership in the DOMAIN. A DOMAIN may be in a DOMAIN by virtue of a LOCATION defined for an upper DOMAIN. Movement between the DOMAINS in which a DOMAIN is in must be managed.

- Negotiate a challenge that checks to make sure that the agent can be moved.
- Identify if the agent has a corresponding swid on the new system. If not, create one.
- Copy the metadata for that agent in the graph of the new server.
- Attach the agent to the indicated place within the new domain.
- Notify the current server that the agent has been successfully replaced.

- Deactivate the agent on the current node (not remove, just deactivate) if the transfer was successful, otherwise send a note to the actor of the current agent that the link failed.

Movement between DOMAINS is different than locating a DOMAINS information through the UDG Node Network (which is discussed in 9.2.2).

5.3.6. SITUATIONS

For the purposes of defining context, the SITUATION relationship is defined. A SITUATION may be a dynamically changing relationship.

A SITUATION relationship is a relationship ENTITY comprised of all DOMAINS that can be perceived and reasoned about by an AGENT Spatial Web clause 6.6.4.. SITUATION is to be understood as defined in [IEEE_7007_2021]: a situation is an entity comprised of participating entities and relationships that represent the limited parts of reality that can be perceived and reasoned about by agents.

A DOMAIN, in particular an AGENT, can create a SITUATION at any given moment in time. This may include determining the visible, nearby DOMAINS and deleting DOMAINS no longer of interest. A SITUATION may be composed of:

- Location in Hyperspace
- Links to other DOMAINS that are currently active
- Membership in DOMAINS that are currently relevant
- Current presence of the AGENT within a DOMAIN which it may not be a member
- Awareness (e.g., via query) of nearby DOMAINS of interest to the AGENT with which it may not have any current relationship.

5.3.7. Dynamic relationships — paths

5.3.7.1. Updating relationships with time and activity

An ACTIVITY is a partially ordered set of changes effected by an AGENT. An ACTIVITY may affect the relationships for a DOMAIN, including adding or deleting links, joining or leaving membership in a DOMAIN, and/or, affecting the member ENTITIES of a SITUATION.

An AGENT may move between DOMAINS. An AGENT may be present in a DOMAIN without being a member. The AGENT may be present in a DOMAIN and then move to another DOMAIN.

5.3.7.2. Constrained paths for change

EDITORIAL NOTE

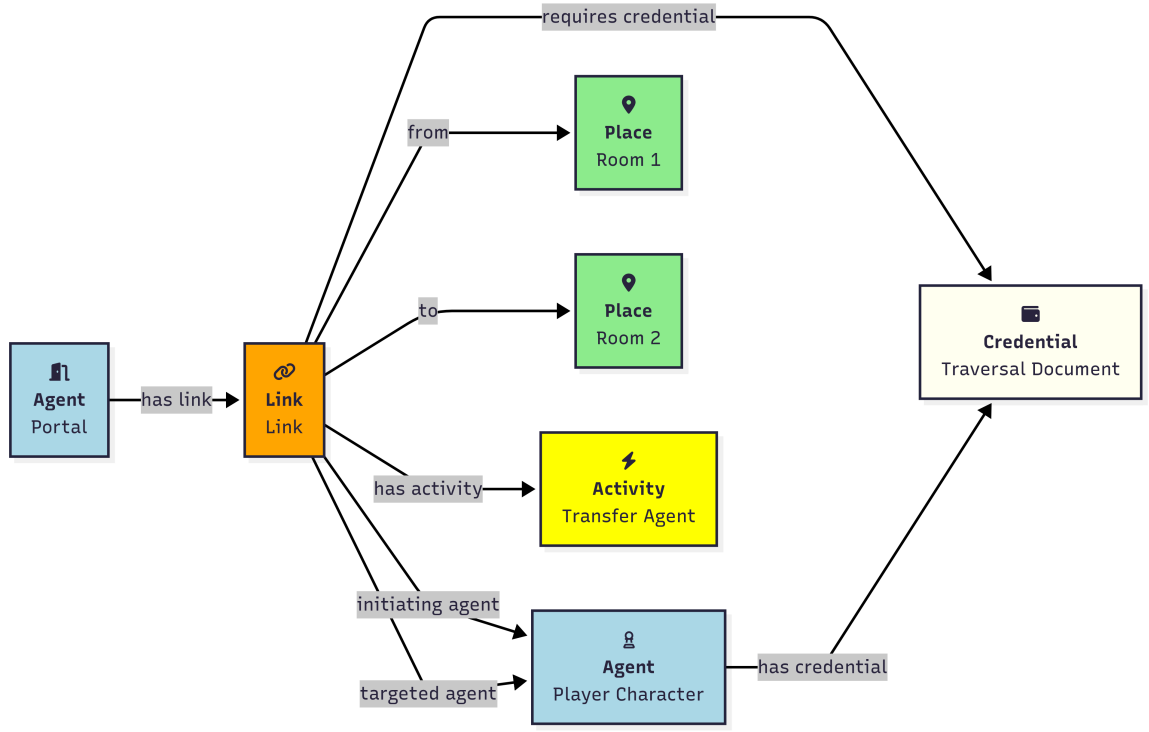
are paths between domains constrained? The following text addressing constrained movement as paths needs to be reviewed.

To go from one location to another, an AGENT has to traverse a path.

Note that there are two distinct actions that can be taken, selection and path traversal. If a location can be *selected*, it identifies that location as being part of an active set of location. If it is *activated*, then the path is traversed as described above.

A portal that is applied to a given path (styled as a door or other kind of portal), that causes the activating agent (such as a player character in a game) to move to a different, specified location:

FIGURE 8: Example of relationships between entities



For an AGENT to traverse a path means moving from one location to another following a particular path of intervening locations. This approach is straightforward and especially conducive to optimization of path traversals to minimize energy expenditure, though as the number of locations goes up, so too does the complexity of such computations.

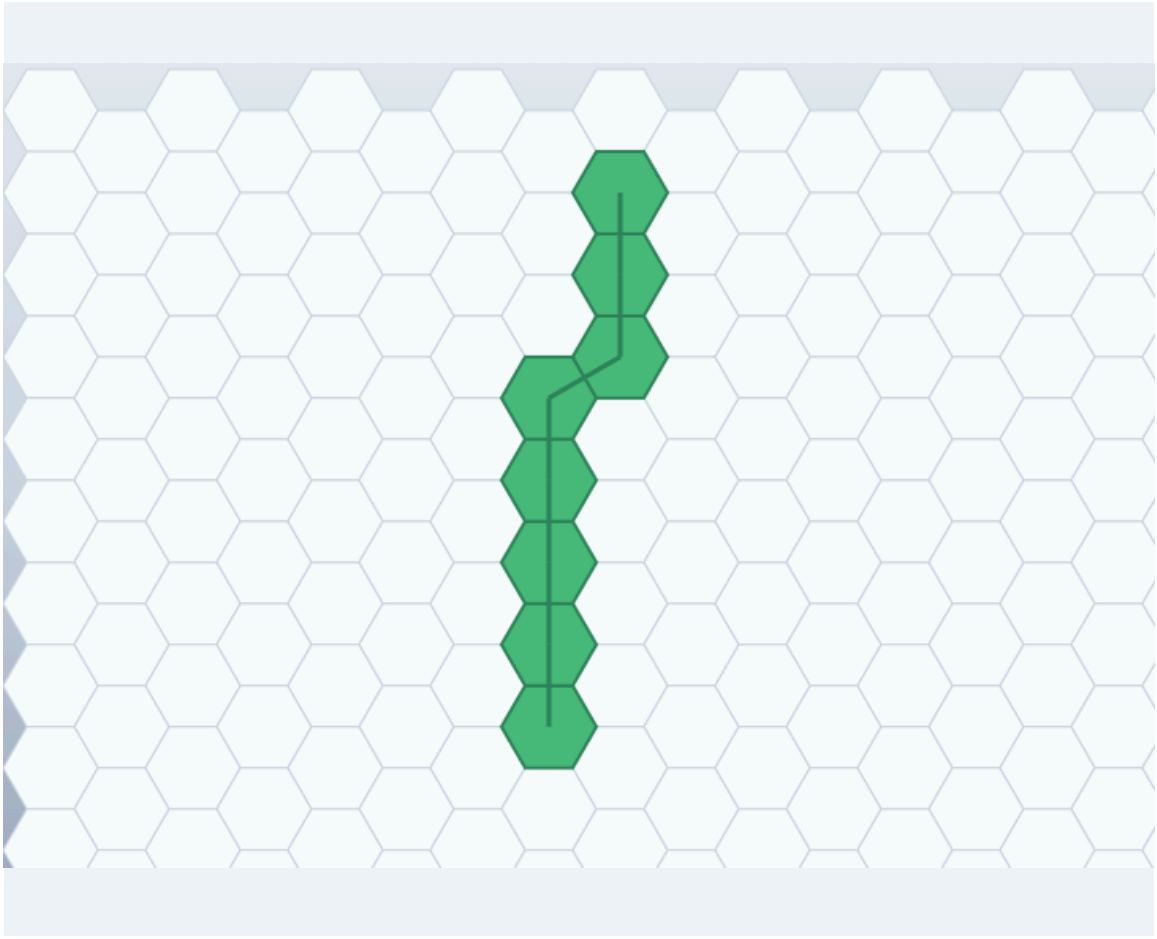
In the real world, of course, we do not hop from location to location but move in a continuous fashion, and a robot or physical twin has to determine the "how" of traversal. This process may live in the interface between the virtual and physical twin.

In general, this information may be stored in metadata that is associated with the link, but that is outside of the scope of the spatial web. For instance, a robot needs to move from the bottom of a hill to the top of a hill along a road. The link may indicate characteristics of the hill — its inclination in particular — but from the standpoint of the Spatial Web, this slope is a challenge that has to be met prior to achieving the key to allow the transition from one location (the bottom of the hill) to another (the top of the hill).

In this case, the path challenge would be to solve a physics problem — is the weight of the robot, the power of the motor, and the inclination of the slope sufficient to reach the top, and are there any routes (sequences of locations) that the robot can take if the slope is too challenging? If the problem is solved, then the robot goes ahead with the selected route, otherwise, the lock remains locked.

For a sufficient large hyperspace, the mesh of potential paths can more closely represent a curve. For instance, the road may be treated as a space with a fairly high density of hexes, and rather than trying to tackle the road head-on in a linear fashion, it ascends the road as a series of switchbacks (much like a sailboat tacking against the wind).

FIGURE 9: Tacking as movement in a cellular space



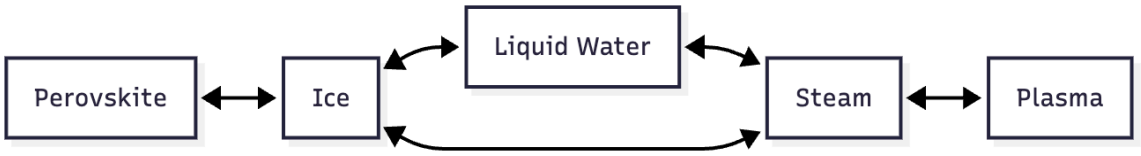
In the case where there is a physical twin bound to an agent, the path remains active until the physical twin indicates it has successfully completed the task, at which point it may update the SITUATION associated with the agent with physical coordinates that can be translated back into tiling.

This means that in general the physical location of a tile may be its centrum, unless this is specifically overrriden with a centrum property.

This analogy also corresponds with non-Hilbert spaces, such as heat/pressure state regimes. In this case, the tiles represent specific regimes of behavior for the system, as the agent (or token) moves from one such state to the next. In the real world,

these transitions are usually analog and may be subtle, but modeling these as a state diagram can be useful:

FIGURE 10: Transitions of state between ice, steam, plasma



The agent's position across the hyperspace of locations indicates what state the agent is in, where the agent can be seen as a marker for the current state.

5.3.8. Requirements and recommendations

TBD

5.4. UDG Taxonomy

The *UDG Taxonomy* is a taxonomy designed to augment discovery within the Spatial Web by providing common concepts and definitions for agents and, by extensions, domains.

> *EDITORIAL NOTE*
>
> *UDG Taxonomy is preliminary and needs to be reviewed*

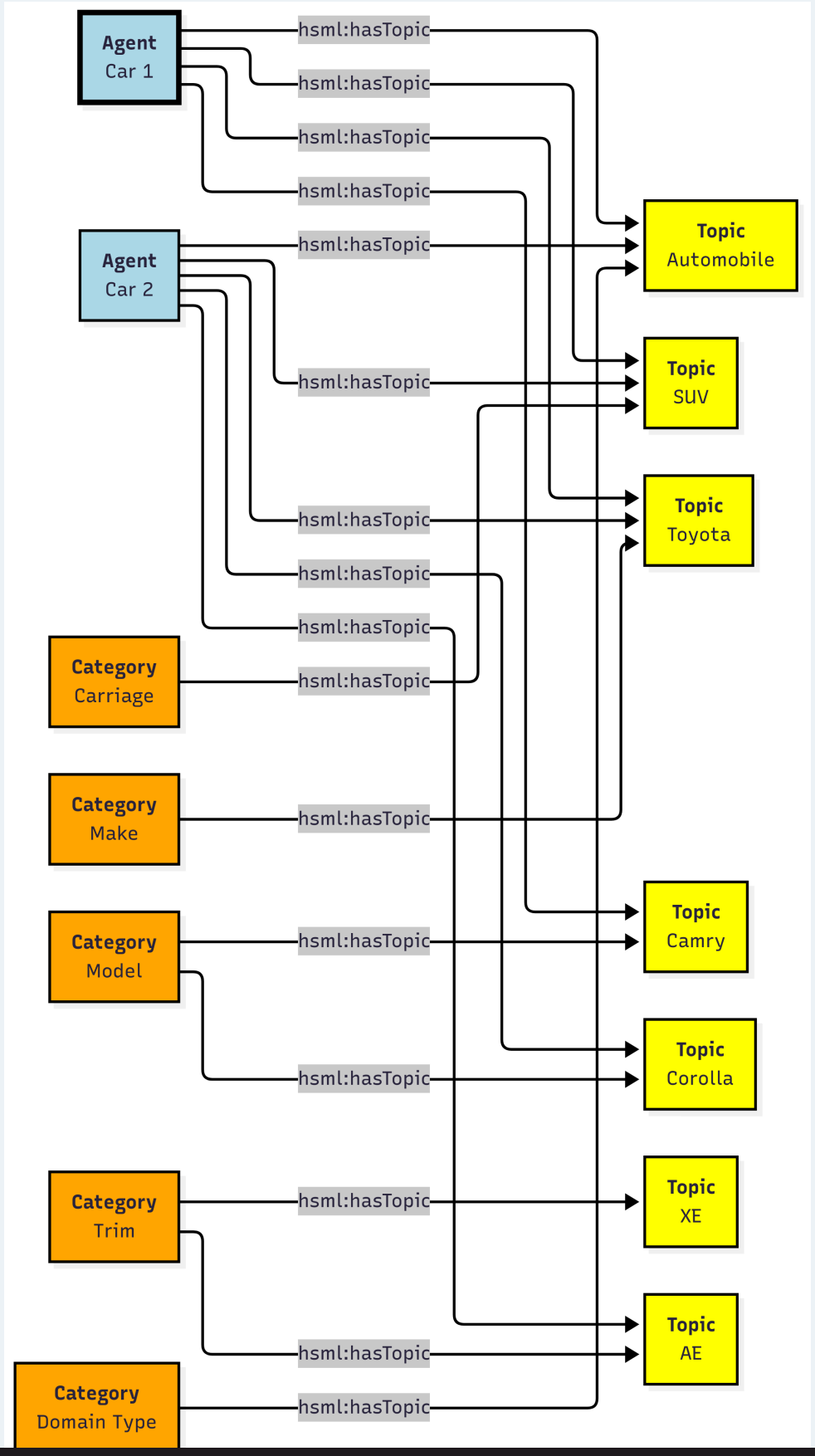
5.4.1. hsml:hasTopic property

The UDG Taxonomy can be thought of as the thesaurus for the Spatial Web. Each term in that thesaurus provides an adjective or noun that identifies some characteristic of a given agent.

For instance, an agent that is intended to be a proxy for a car in a smart city scenario may be identified by a number of such characteristics: the vehicle's *make*, *model*, and *trim*, its *carriage designation* (a sedan, sports car, SUV, light truck), its *primary and secondary external and internal colors*, its *engine type* (internal combustion engine, diesel, electric, hybrid, hydrogen-powered) and so forth. A building may be classified by *purpose*, *construction method*, *zone classification*, etc. A robot may be given by its *purpose*, *ambulatory status*, *activation level*, etc.

Each of these terms are used primarily as mechanisms for classification, and are considered as *categories*, with each particular enumeration in turn considered a *topic*. The `hsml:hasTopic`, `Place:hasTopic` and `hsml:hasTopic` properties in HSML takes zero or more topics as arguments.

FIGURE 11: UDG Taxonomy



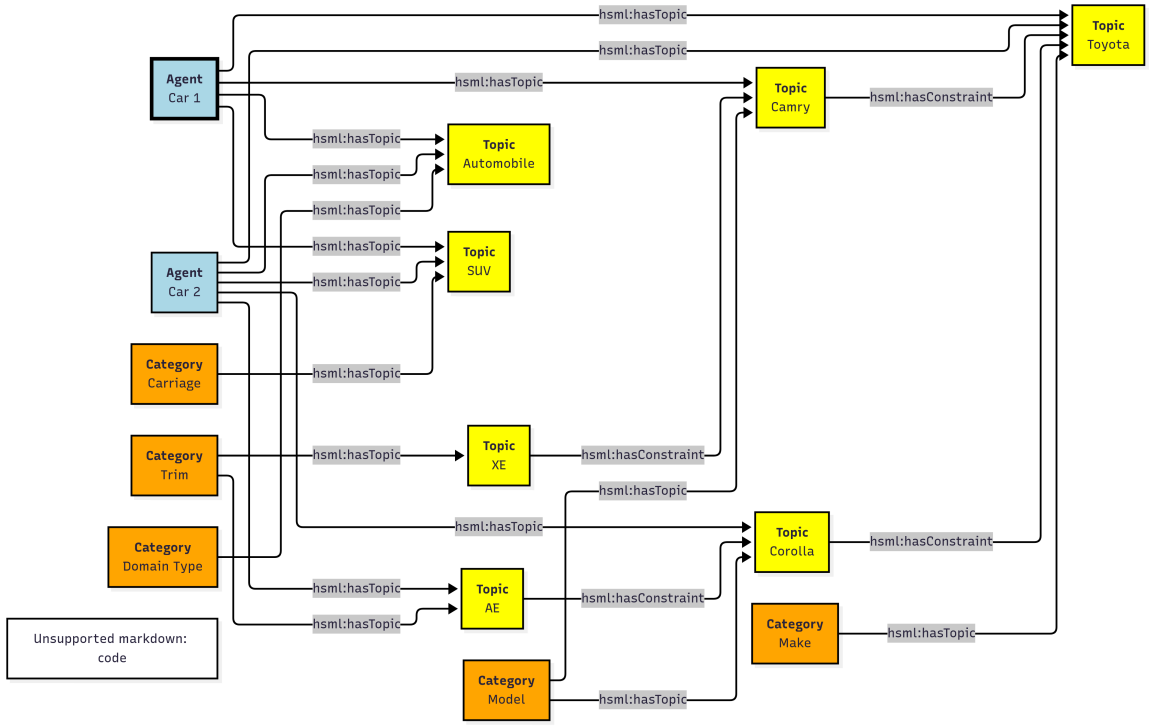
The power of the UDG taxonomy is in its ability to cluster agents by topic, mediated by category. For instance, car 1 and car 2 are both of the same make (Toyota) but of different models (Camry vs. Corolla) and trims. They are also of the same "domain Type" of automobile. Note that domain type here is not privileged, it is simply one more category that agents can be in, though a fairly broad category.

5.4.2. hsml:hasConstraint property

Some times there are interdependencies between topics. For instance, the Corolla and the Camry are two different models produced by Toyota, and another car company will not produce those same models. Similarly trim provides variants for a given car model.

These relationships are called *constraints*, which is a relationship indicating that one topic is dependent upon another. This changes the diagram somewhat:

FIGURE 12: An example of the UDG Taxonomy with constraints

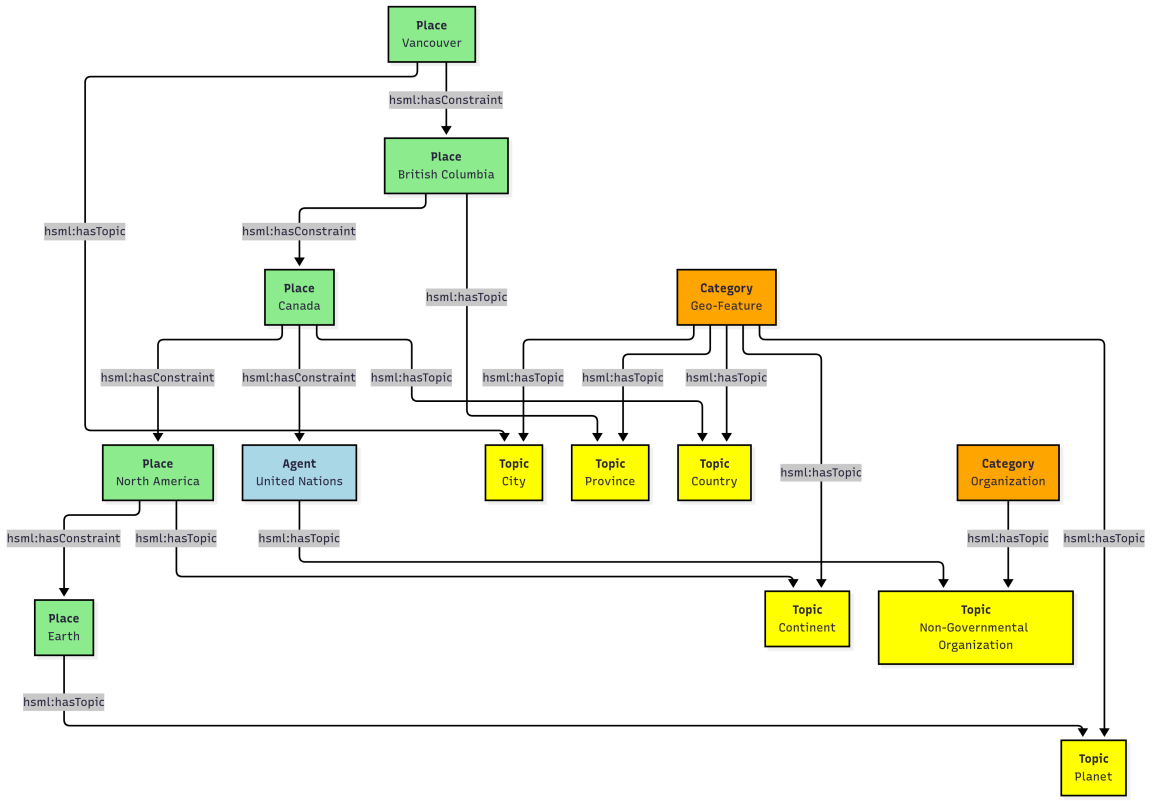


5.4.3. Geographic DOMAINS with `hsml:hasTopic` and `hsml:hasConstraint`

Geographic DOMAINS can be defined in a similar manner. For instance, a place may be a country, city, planet, river, lake, sea, township, etc., Each of these are `geoFeature` topics, though these may be subclassed.

For instance, Vancouver, British Columbia, Canada, and North America are all places, they are connected as follows:

FIGURE 13: UDG Taxonomy example with a GeoFeature



In this case, while these are all geoFeatures, Vancouver is a city while Canada is a country. Significantly, the implicit structuring (Vancouver is a part of Canada) becomes simply a constraint relationship here, albeit one that can be exploited for reasoning purposes. Furthermore, Canada might also be in another constraint relationships with an organization of countries (such as NAFTA or the United Nations), so the hierarchy here is a hierarchy of topics, and is actually more holonic than strictly hierarchical.

5.4.4. Topics vs. States

At first glance, topics and state properties would appear to be similar — one could express topics as states, though they serve somewhat different purposes. A state typically associates a facet value with a normalized value indicating the strength of that value, while a topic typically is a binary relationship used for classification exclusively (you could say that a state property is a topic with a value of either 0 or 1, not something in between).

Moreover, topics tend to be relatively stable once assigned to an agent or place. This makes them useful for classification, and discovery. For instance, while it is possible to identify the state of a traffic light as being red or green, this value is likely to change regularly between queries. However, it's designation as a traffic light is very much unlikely to change. meaning that if you look for traffic lights on a given node, you will likely get all such agents.

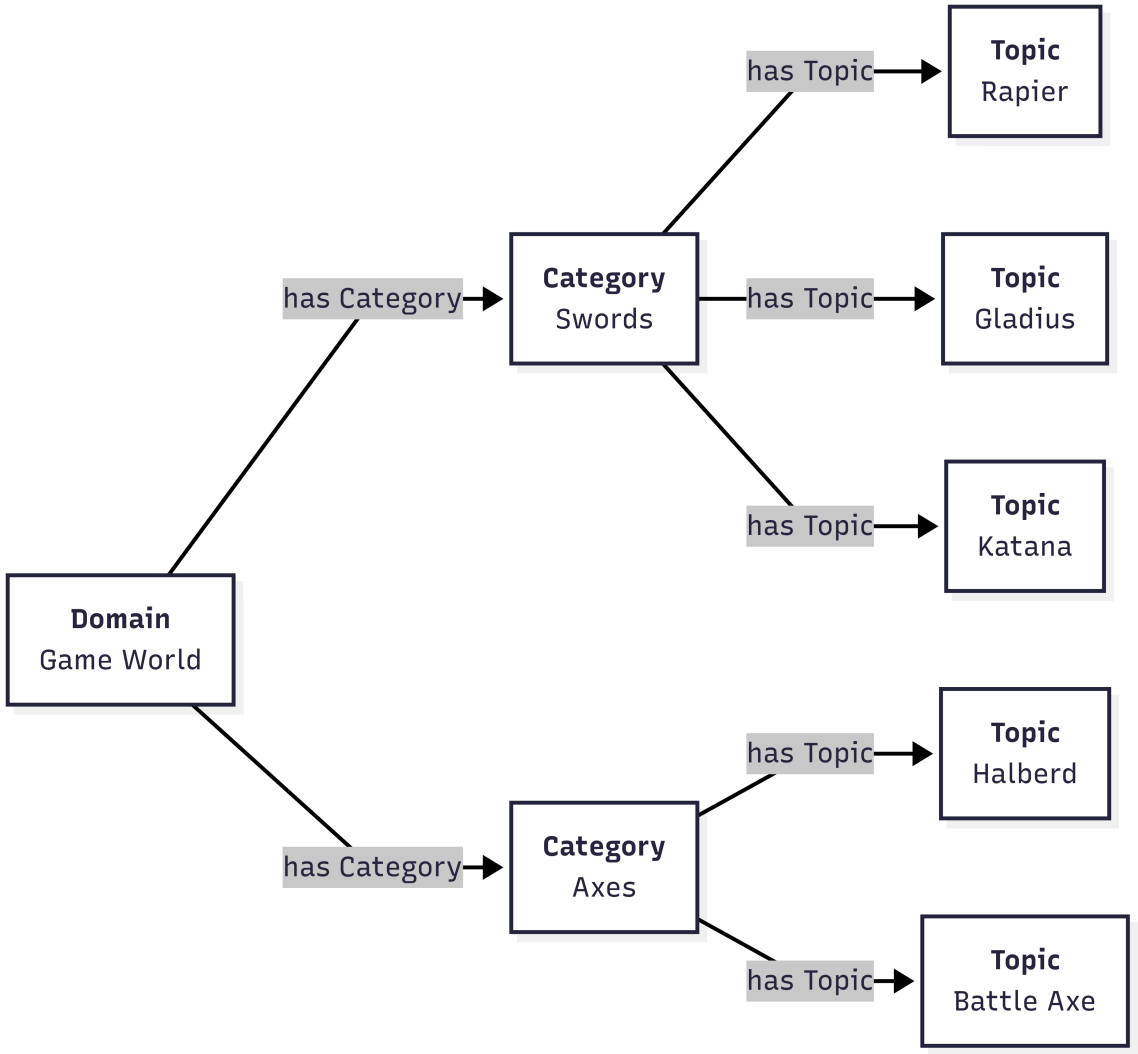
Discovery on a given spatial web node then becomes a matter of querying the node for desired topics. Note that the topics can include synonyms (analogous to skos:altLabel) that can be compared to the base topic labels for mapping to the respective node. Moreover, multiple language versions of the same topic can be provided in order to match in different languages.

Note that topics can be used for state variables. In that particular case, however, they won't necessarily participate in search unless they are also incorporated as hsml:hasTopic objects.

5.4.5. Taxonomies and Schemas with Domains

A taxonomy is a data structure that defines the topics that are relevant to that domain. The predicate `hsml:hasCategory` identifies the categories that are defined within the domain (and is a property of the hsml:Domain class). Each category in turn identifies one or more topics that are associated with that category. Because domains are named graphs, the categories defined are local to that domain.

FIGURE 14: Game World



5.4.6. Importing Taxonomies and Schemas

The predicate hsml:includeDomain is an instruction to add the graph of the indicated domain as part of the graph search, and is applied to the hsml:Domain object. This makes it possible to import external taxonomies and schemas into an existing domain. This has a lot of utility, in that it means that a domain can be defined that contains common taxonomy and schema definitions which can then be used within another domain.

Typically, a spatial web node will contain a primary domain that contains many of the core concepts, structures, and places and common agents that may be used within the majority of domains on that node. This can be imported into any given domain, providing a common framework for terms. In general, this is like a link in that the

SWURL for the resource is passed. This is then interpreted by HSML (through the graph.d engine) to add this as resource into the active graph for the domain.

This can also be done across node boundaries. A *resource repository* is a domain server that contains various entity resources that may be used across the entirety of the spatial web. By working from these common repositories, entities such as common places, frequently defined agents, taxonomy terms, and so forth can be referenced within a domain, while staying up to date.

Note that because of latency considerations, there are times where it may be more advantageous to autoload an external domain's contents permanently onto a given spatial web node. The hsml:importDomain is similar to the hsml:includeDomain` but copies the imported domain content to the server directly, rather than referencing them from an external server. This creates an internal domain, and requires that you specify both the external SWURL and the internal name:

FIGURE 15

```
[] a hsml:Domain ;
    hsml:swurl <domain/ExternalTaxonomy>
    hsml:importDomain <https://myExternalResources.
com#domain/externalTaxonomy> .
    .
```

When this is interpreted by the hsml parser, it will retrieve the subgraph from the external domain and load it into the graph as a named graph with associated local-name SWURL. This may frequently be done from packages that are loaded in initially, and that may be periodicallly refreshed.

The primary difference between `hsml:includeDomain` and `hsml:importDomain` is that `hsml:includeDomain` creates a domain extension from the external system that is always up to date but that may have higher latency (and is not actually stored on the same node as the including domain), while the hsml:importDomain creates a local copy of the external graph in the calling graph. This graph may be out of date but that has much lower latency.

An example of an included domain would be a hotel that had a number of guest rooms, each of which were in their own domain (which may or may not be on the same spatial web node). An example of an imported domain would be one that incorporated a taxonomy that is commonly used by other domains but that also does not change frequently and may be heavily referenced.

Another way of thinking about imports vs. includes is that an import is essentially a cache of one domain within another, while, an include is a temporary reference.

Note also that in both cases, the node server MUST have the relevant credentials to load in the external domain. Otherwise this statement will fail and an error message will be sent to the error channel.

5.4.7. Requirements and recommendations

TBD

5.5. Use cases

5.5.1. Cultural Location Tourism

Annex C.1 provides a mutipart example of applying the UDG to the Spatial Web Application Scenario for Cultural Location Tourism

5.5.2. The Light Bulb Room

> *EDITORIAL NOTE*
>
> *This use case is under review and may be updated or deleted in the future based on implementation experience.*

5.5.2.1. Overview

This is a simple example of a DOMAIN. The Light Bulb room is a room with a single switch. The switch can be on or off. When the switch is on, the light is on. When the switch is off, the light is off.

5.5.2.2. Where

The domain is in a *Place* that we can call `Light Bulb Room =1`. Note that for the Domain, there was a template (or base class) called `<Light Bulb Room>`, specified via a schema language (for the moment, SHACL), that can both be used to create multiple instances, and to limit the number of instances so created.

This handles the particular situation in which a given instance is tied to a digital twin as well as the situation where a single long-running domain may exist. For the light bulb room class (LBR), if the instance was tied to a physical room, then LBR=1 would need to persist between sessions, which would mean that the SWID for the room would be persistent for all agents that had permissions to access the domain.

Note that Place in this case need only be a single value — the Room itself. The domain is the conceptual room, and there is no real reason to subdivide it into component places in this very simple model.

5.5.2.3. How Is Space Defined

The operational definition of a hyperspace is the set of all valid places within a domain. The spatial web (as currently defined) is a discrete spatial system. What this means in practice is that things are located in specific discrete Places, and within a domain, an agent moves from one such discrete Place to another through a link. A Place can describe the specific extent in other terms (H3, Geometric Tiles, ESRI geometries andso forth) but the domain determines which of those places are considered valid. This in turn reduces a potentially intractable geometric description into a graph-oriented topological description.

5.5.2.4. How Is Space Connected

In a *domain*, two or more *places* are connected by *links*. A link is analogous to a hypertext link in HTTP. In each domain, there is typically at least one link from a source place to the *home place* of the domain. When you "go to" a domain, you're agent is actually moving to the home place for that domain, unless another place is explicitly stated.

In the Light Bulb Room, there is only one place defined for that domain, so if you are coming from the directory domain for the SW Node, then the directory will contain a link to the LBR=1 place. Unless there is a conditional lock on the link (you have to satisfy a test condition), you (or more specifically your agent) can generatlly backtrack across links through the client

5.5.2.5. What

This indicates the things that are bound to the room that are controllable from within the domain. In this case, there are two distinct things — a light switch and a lamp. By activating the light switch, you enable the lamp. By deactivating the light switch, you disable the lamp. In an analog system, of course, what the light switch does is turn power off to an electrical outlet, but this is an operational detail that is unimportant to the model.

Note that there are a number of low level Things that will be generally subclassed. For instance, a lamp is a Meter that can take a value from a range of values (here [0,1]) A Toggle is a Thing that can take a Boolean value, and switch from one value to the other when activated. In short, many of these have a direct correspondance to HTML form components. These are detailed as part of the Activity specification, which is out of scope for this specification.

5.5.2.6. What Kind

A domain can be classified based upon a conceptual facet value tied to a specific classication facet (known as the Domain taxonomy). The specific facet can be given as a subproperty of this depending on the definition given within the associated shape.

Everything is shape based rather than class based. This means that you can use combinations of facets to determine which property shapes apply to a given entity, which in turn means that you are not as dependent upon RDFS based supclass/ subproperty inheritance.

In the case of the Light Room =1,onw such classification might be IoTDevice, while another may be Purpose:Illumination or something similar.

5.5.2.7. Who

This indicates the agent(s) that are currently within the context of the room. There may be zero or more agents in the room at any given point, though the domain model could be set up to limit the number of agents that can occupy a given place at a certain time. This creates a crude physics.

Note that in this model as well, there is no indication about the agents are, or what priorities they have. In general, if one agent turns the light on and the other turns it off, then the system will reflect the current state from the last activity that occurred.

Agents can move from one place to another (see Places for more information).

5.5.2.8. When

Each domain has a clock. Typically, such clocks can be defined in terms of a Spatial Web Node chronometer that is specific to the host (to the extent that in many cases, the domain can refer to a specific "System Clock", which is the default chronometer when not otherwise supplied). Note that this is used primarily to control timing and action within system on the part of autonomous entities, and in general is NOT synched from one node to the next. A chronometer is of type Entity:Thing.

Also please note that the chronometer is not technically part of hyperspace. If, for instance, you had a relativity simulation, then the time component of such a transformation would be treated as a coordinate in the hyperspace system (if you are doing Lorenz Transformations, for instance), but this is only peripherally related to the domain chronometer. The chronometer is, however, a key part of maintaining a domain history (see What Happened?.

5.5.2.9. What Happened

Each domain manages its own queue indicating relevant state change reports that are updated as part of the activity. This becomes the history of the domain. In this case, every time that the switch is flipped, the context of the domain for those things maintaining a history get written to the queue, indicating who initiated the action and what the state of the light (the meter) was at the time. This effectively creates a recording of the session, and in theory should be transformable to reproduce the state transitions of the system.

> *EDITORIAL NOTE*
>
> *The depth of the queue will obviously be dependent upon system resources, and may be in a condensed serialized format. The exact mechanism for how this works is still TBD.*

5.5.2.10. How

One of the roles of the chronometer is to indicate when a given domain should check to see if an expressed contextual configuration is in place (typically by querying the graph) and if it is, to then cause some activity within the domain. These are domain specific, such as expressing representations of the domain to an external channel.

5.5.2.11. Why

Most domains have objectives and goals. A remote drone domain, for instance, exists to get the drone to a target, perform a function, and hopefully return safely. These objectives typically will put the domain into a different state (Reset, Archive, Delete, etc.) In a game, these are the conditions that end the game and determine the winner. In a story, this is The End. In a device controller, this the termination of the updates to the devices in question. When the domain is instantiated, the why is set up as an end condition and is evaluated as part of the processing cycle for the domain.

5.5.3. Wiki Data as a conceptual domain

> *EDITORIAL NOTE*
>
> *This example is under development current in SWF and will be updated based on the results.*

5.5.3.1. Overview

Wikidata is a collaboratively edited multilingual knowledge graph hosted by the Wikimedia Foundation. It is a common source of open data that Wikimedia projects such as Wikipedia, and anyone else, is able to use under the CC0public domain license. Wikidata is a wiki powered by the software MediaWiki, including its extension for semi-structured data, the Wikibase. As of early-2025, Wikidata had 1.65 billion item statements (semantic triple).

The terminology for Wikidata is: - Entity is an item, property or lexeme - Item refers to a real-world object, concept, or event - 12,366 properties in Wikidata:List of properties - Lexeme is an entity of Lexicographical data.

Wikidata can serve as a source of data for a Spatial Web Conceptual Domain for common terms.

5.5.3.2. Where

The DOMAIN is Spatial Web Common Concepts. The Common Concepts DOMAIN is a conceptual DOMAIN type.

5.5.3.3. How Is Space Defined

Spatial in WikiData is a graph. The use case in the Spatial Web is to

- define the WikiData in terms of the Spatial Web ontology, e.g., to define WikiData as a conceptual domain
- create a WikiData spatial embedding that can be used to locate the DOMAINS in hyperspace.

5.5.3.4. How Is Space Connected

Triples

5.5.3.5. What

Concepts

5.5.3.6. What Kind

TBD

5.5.3.7. Who

An AGENT uses the Common Terms DOMAIN to define a new DOMAIN.

5.5.3.8. When

An ACTIVITY of checking consistency of terms over timee.

5.5.3.9. What Happened

TBD

5.5.3.10. How

TBD

5.5.3.11. Why

TBD

5.5.4. Requirements and recommendations

TBD

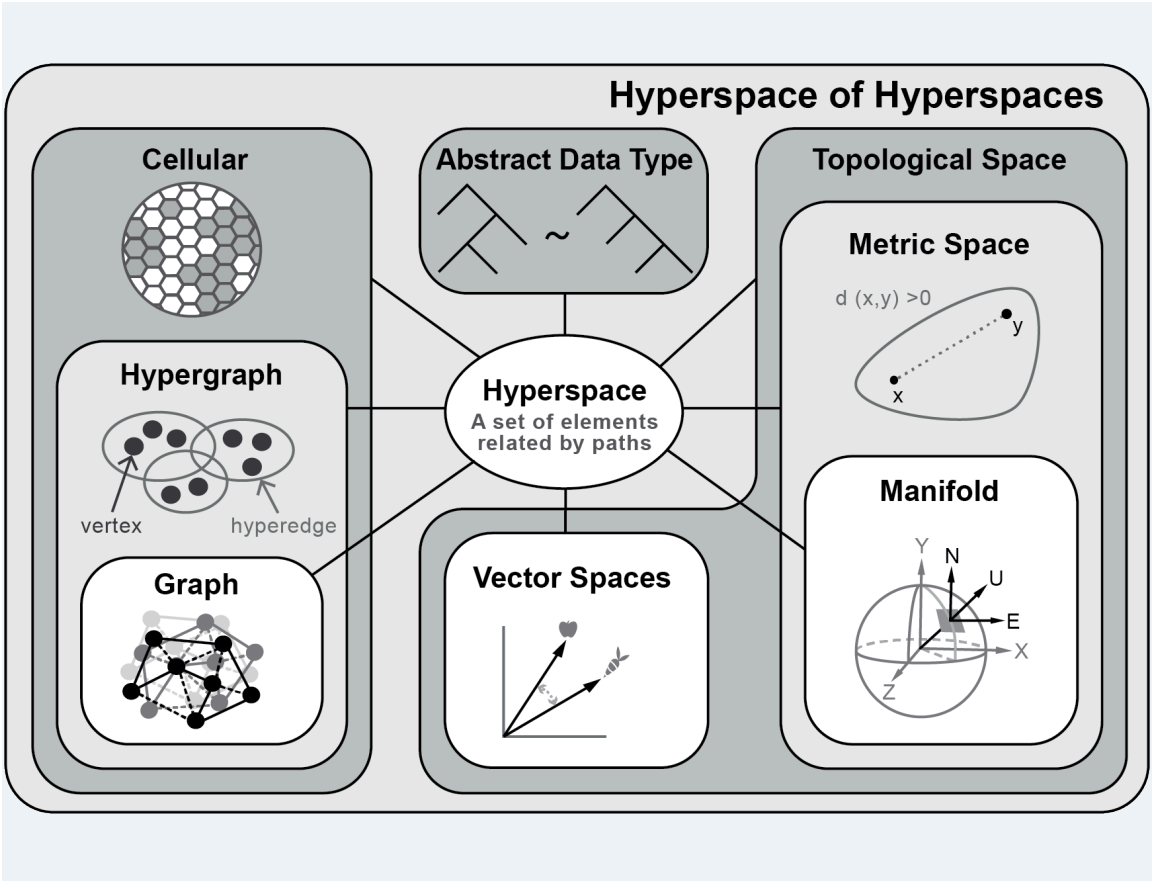
6. UDG and hyperspace

6.1. Spatial Web Hyperspace

6.1.1. Classes of hyperspace

The Spatial Web hyperspace ENTITY provides the fundamental organizing method for entities in the UDG. To enable this role, Spatial Web clause 6.2.3.7 provides a formal definition of hyperspace, as any set whose elements are related by a formal notion of *path*. The basic classes of hyperspace include: Graphs including Cellular spaces and Vector spaces with coordinate systems.

FIGURE 16: Basic classes of hyperspace



The Spatial Web Protocol Architecture and Governance standard defines concepts for hyperspace summarized here for convienence:

- Classes of hyperspace: graphs, vector spaces, hypergraphs, abstract data types.
- Hyperspaces of spaces: where a point of a space is a space of the corresponding type
- Cellular spaces: in which cells of a chosen type are connected according to a schema.
- Geographic spaces: locations relative to the Earth
- Time in hyperspace: as a dimension, as a duration, as a trajectory
- Structures: origin, dimension, metric, similarity, norm and inner product
- Operations: products, sums, subspaces
- Connecting spaces: points in common to multiple subspaces, i.e., Portal.

6.1.2. Hyperspace attributes

DOMAINS are represented in hyperspace by identifying the type of hyperspace and the location of the DOMAIN in the hyperspace.

Location is a key attribute of HYPERSPACE. Location identifies a specific coordinate or index value in a hyperspace. The Location attribute takes on a specific form based on the class of hyperspace: cell index for a cellular space, coordinate value for a coordinate reference system. See the HSML Implementation Specification for the controlled attributes for location.

Geometries are mathematical objects in the vector space class of hyperspace. ENTITIES in a vector space may have a hyperspace:spatialProperty of geo:hasGeometry. A geometry may be a single point location or it may be a geometric structure like a polygon or sphere in any number of dimensions.

ENTITIES as features may have attributes of location hyperspace as well as other attributes not associated with hyperspace. This is similar to a Geographic Feature where location is one of several types of attributes associated with a geographic feature. The existence of a feature is independent of any specific geometric representation associated with the feature. A feature may have multiple geometries as attributes depending for example on scale it may be associated with a point or a complex polygon. For example, a city may be have a point attribute and it may also have the city's boundary as a location in hyperspace.

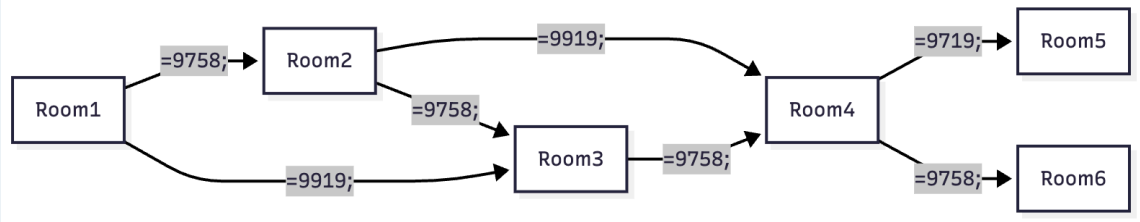
6.1.3. Topological vs geometric representations

Geometry is concerned with properties of space such as the distance, shape, size, and relative position of entities. Topology is concerned with properties that are invariant when an entity is deformed, e.g., stretching, twisting, crumpling, and bending.

For a topological space, DOMAINS consists of a set of places, each of which is a conceptual node connected by links. The set of all places that are traversable within the graph makeup the hyperspace for that domain, with the links in turn controlling access from one place to another within the domain. Relationships between entities in topological space and geometric space are expressed in the Poincaré duality.

Figure 17 is an example of a topological representation of a domain: a building with several rooms. It consists of six DOMAINS, but each DOMAIN does not necessarily have to represent a physical location in the real world. Instead, the DOMAIN has a geometry that is the boundary of the room. It could represent stations in an assembly line, steps in a process, a detailed internal represntation of a given subsystem, and so forth.

FIGURE 17: Topological relationships of rooms in a building



In this case, the hyperspace for the domain consists of six "rooms", each connected by links of various types:

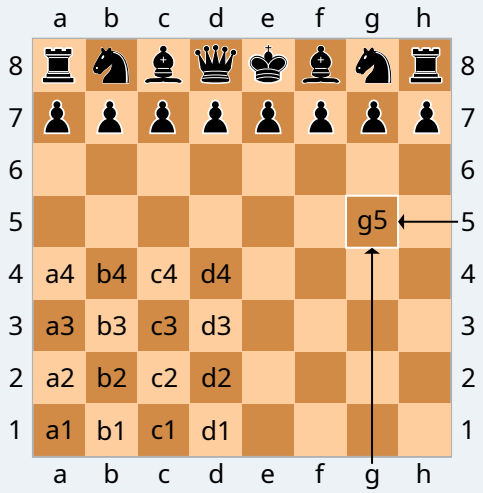
- Pointers (&=9758;) represent open links — an agent can move from one room to the next freely.
- Keys (&=9919;) represent locked links — the agent needs some form of key to open the link and move to the next room.
- Finally, clocks (&=9719;) represents conditional locks — an external condition (such as a store being closed for the night) must be met before traversal can happen.

Access to various domains can e controlled as they are within the overall domain that are constrained by the links that connect them. Because links are contextual, only certain subsystem can be accessed if the AGENT has the relevant key or some external condition is in force.

The hyperspace of the domain then becomes the set of all domains within that domain. This solves another problem that a more physical realization introduces — determining whether the AGENT is at the edge of, or out of the boundaries of, a physical space. In a topological model, if the place is not in the domain, then it is not accessible by ANY agent.

Another example of a topological representation is to use a cellular space. For example, consider a domain that represents a game (an instance) of *chess*. There are 66 potential cells where a given piece can be located — the 64 squares that make up an 8×8 chessboard, and two cache areas, one for each player, where pieces that are removed are located.

FIGURE 18: Chess board with algebraic notation



Each of these cells has a distinct identifier that makes up its address, and, within the scope of a particular game of chess, is considered unique. Note that the places here are not globally unique — the same position may be used across multiple games across multiple servers — but within the context of its domain, it IS unique.

Notations for each of the two caches are defined — White Cache (WC) and Black Cache (BC) — for pieces that are outside the formal scope of the board but nonetheless represent the location of a specific piece. If I move the White Queen (WQ) to e4, then that piece is now located in the cell 4 units up from the White queen's side of the board and five units (e) from the left edge of the board.

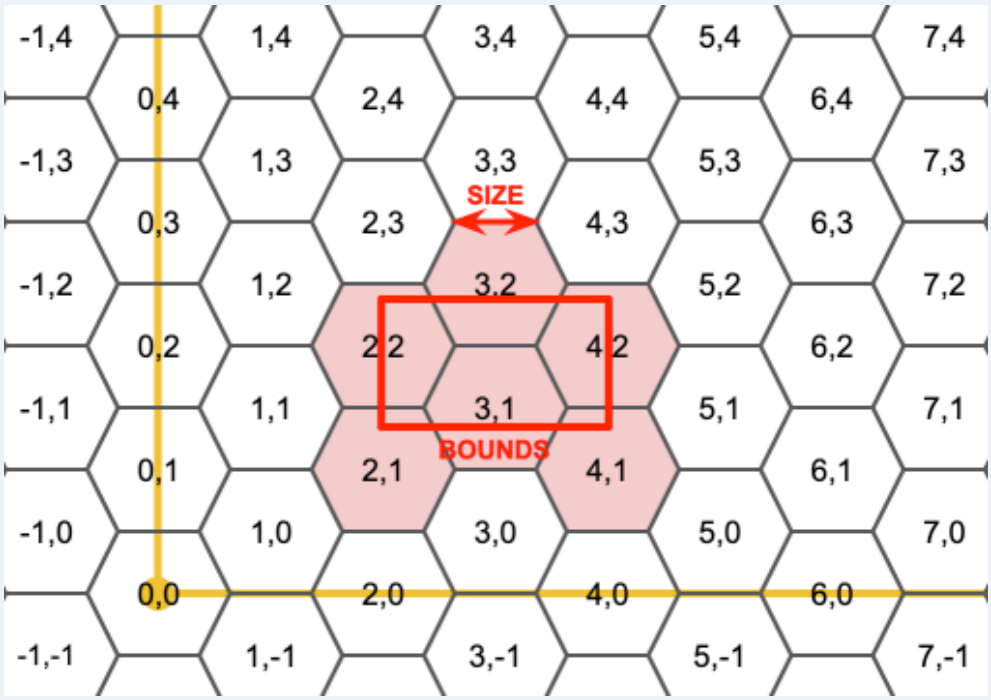
The hyperspace here is ONLY these particular squares (or caches). Position j17 cannot be occupied because such a position, while plausable, does not exist within the set of all permissible places in the hyperspace.

Movement from cell to cell within a domain does not have to follow contiguity rules. The knight in particular is very constrained in where it can go (if the knight is on e4, than it can go to d6, f6, c3, c5, g3, g5, d2 or f2, but it cannot go to d4 or f4, which are contiguous to it), but can also move over other pieces so long as the target place is not occupied by a piece of their own color.

A Spatial Web Domain is not a perfect reproduction of the real world. The knight cannot move within its cell in this particular domain (though a representation could move within that square, it's not something that is considered in the model).

Consider a hexagonal tiling of a space, as may be considered by a table top war game. The addressing in this space is more complex, because each hexagon has a particular index (address) that can be derived from some particular algorithm. The H3 System system used by Uber is an illustration of such a tiling system on a sphere.

FIGURE 19: Hexagonal tiling



6.1.4. Requirements and recommendations

TBD

6.2. Spatial embedding of entities

6.2.1. Spatial embedding technology

The Spatial Web applies spatial embedding to Spatial Web ENTITIES to create the vector space version of Hyperspace. The Spatial Web defines space as hyperspace in order to represent many entity properties using a spatial embedding approach. Spatial embedding: mathematical methods for representing data in a continuous, multi-dimensional vector space.

Spatial embedding was a key technology to the recent advances in LLM AI. A survey of the development of hyperdimensional representation technology presents the main ideas behind the models ([14]). One of the fundamental papers showed an efficient method for learning high-quality distributed vector representations that capture a large number of precise syntactic and semantic word relationships ([15]). LLMs use spatial embedding for linguistic concepts, encoding words as vectors. The ([16]) code represents a word as a high-dimension vector of numbers which capture relationships between words. In particular, words which appear in similar contexts are mapped to vectors which are nearby.

Spatial embedding enables useful functions based on syntactic and semantic regularities in language, and that each relationship is characterized by a relation-specific vector offset. This allows vector-oriented reasoning based on the offsets between words. For example, the male/female relationship is automatically learned, and with the induced vector representations, "King — Man + Woman" results in a vector very close to "Queen."([15])

(World2Vec) was an internal name for the PyTorch-BigGraph: A Large-scale Graph Embedding System project ([PBG]).

Scalability of embedding needs to be assessed for the Spatial Web. ([25]) showed there are limits of embedding models under a certain vector paradigm and calls for future research to develop methods that can resolve this fundamental limitation.

6.2.2. Cartographic visualization of concept mapping

Spatial embeddings can be visualized using cartographic methods. ([13]) has developed methods for applying cartographic metaphors and methods for rendering a spatial embedding of non-geographic concepts in to a map-like form. Vizualization tools can be used for analysis of the conceptual space. The visualization also serves as a locatting mechanism, e.g., "GPS" for locating knowledge. The visualization requires that the knowledge system works based on a known reference system. This calls for the definition of Hyperspace Reference Systems.

FIGURE 20: Cartographic visualization of spatial embedded concepts

6.2.3. Hyperspace Reference Systems (HRSs)

Vector emdeddings occur in a vector space. Vector spaces are indexed using coordinate systems. The coordinates are the index.

Defining hypespace vector coordinate systems are essential for interoperability of Hyperspace vector embeddings. Geographic spatial reference systems can provide the basis for defining Hyperspace Reference System (HRSs).

Geographic space is indexed using a Coordinate Reference Systems (CRS). A CRS includes both a Coordinate System (CS) and a Datum. The CS provides the dimensions and units. The Datum anchors the abstract CS to the physical world. Typically a Datum includes a benchmark plate at a specific location to which coordinates are assigned. The assigned coordinates for that location allow for transformation to a CRS with a different datum. (See ISO 19111 Referencing by coordinates) A coordinate set such as

latitude and longitude is ambigous unless a datum is specified. The GPS positioning system led to the most widely used family of datums known as WGS-84.

By analogy, investigation is needed to determine if HRSs can be defined for hyperspace coordinates systems using for Spatial Web embedding. CS systems for spatial embedding already exist. Datums for Spatial Web hyperspace HRSs are needed. The datums most likely will be based on real-world semantics. Real-world semantic datums may allow for transformation between HRSs based on differing semantic datums.

With HRSs based on semantic datums defined, Hyperspace Entity Locators (HELs) can be defined. Much like URLs locate based on hypertext, HELs would location an ENTITY ased on an HRS. HELs would locate ENTITIES in hyperspace.

With HRS and HEL defined, the ability to go from the UDG KG to Hyperspace can be done. ENTITIES in UDG KG would include an HEL to located it in hyperspace. Thereby providing every KG ENTITY a locating in an HRS; and the locations could be converted between HRSs.

It will be important to define HRSs that are useful across many DOMAINS.

6.2.4. Requirements and recommendations

TBD

6.3. Hyperspace geography

6.3.1. Hyperspace geography defined

Hyperspace geography is the extension of traditional geography using hyperspace. Traditional geography is the study of objects and phenomena that are directly or indirectly associated with a location relative to the Earth. Hyperspace geography is an ecology of humans and autonomous agents interacting as complex adaptive systems in physical space and extended to hyperspace.

The First Law of Geography states that “everything is related to everything else, but near things are more related than distant things” Miller [B89]. Traditional geography calculates this distance in physical space. Hyperspace geography extends the First Law by calculating distance in hyperspace. In hyperspace, entities may be physically distant while being strongly coupled in other dimensions.

By identifying strong relations between entities that are close in hyperspace but may be physically distant, the Spatial Web defines hyperspace geography. Such a hyper-geography provides new perspectives.

6.3.2. DOMAINS in hyperspace geography

Features are the key concept for modeling objects and phenomena in traditional geography. In the Spatial Web, DOMAINS play the role that features play in traditional geography. DOMAINS like features are abstractions of phenomena of interest.

Features in traditional geography include continents, mountains, rivers, countries, cities, roads, buildings, houses, and many others. DOMAINS in the Spatial Web include

these traditional features and include entities not considered features in traditional geography: concepts, organizations, agents.

Geographic information uses Spatial Referencing Systems (SRS) for location referencing including Coordinate Reference Systems, Discrete Global Grids, and civic addressing. The Spatial Web extends SRSs to include graph referencing systems

Features in traditional geography are conceptual and have geometric attributes in a physical space, e.g., the boundary of a parcel of ground. As there may be multiple geometries associated with a feature, the feature exists independent of the geometric attribute. Determining the geometry of an attribute may not be easy or unique.

DOMAINS in the Spatial Web exist independently of having a hyperspace represenation. A DOMAIN may have multiple hyperspace representations. A DOMAIN may have hyperspace representations of differing types: graphic, cellular, coordinate. A DOMAIN may have multiple represetnations of the same type: several coordinate geometries.

6.3.3. Features in LLM spatial embedding

Recent research has shown that features in LLM spatial embeddings may have universal applicability. Such universal applicability provides the basis for building interoperability methods between hyperspace models in the Spatial Web.

The ([18]) Claude 3 LLM was found to include a diversity of highly abstract features. Examples include features for famous people, features for countries and cities. Many features are multilingual (responding to the same concept across languages) and multimodal (responding to the same concept in both text and images), as well as encompassing both abstract and concrete instantiations of the same idea.

Representations in AI models, particularly deep networks, are converging ([20]). Different neural networks representations are becoming more aligned. Convergence is occurring across modalities: vision models and language. This convergence is driving toward a shared statistical model of reality, akin to Plato's concept of an ideal reality.

Harnessing the universal geometry of embeddings provides a method for translating text embeddings from one vector space to another ([19]).

6.3.4. Hyperworlds

Spatial embedding of any ENTITY using HYPERSPACE results in generalized features populating Hyperworlds. Spatial analysis of features in Hyperworlds can provide insights not previously available. To support the analsysis Hyperworlds there must be development of well-known (6.2.3, HRSs), i.e., HRSs with as much generality in Hyperspace as WGS-84 has in geographic space. Hyperworlds make visible relationships between generalized features leading to new insights.

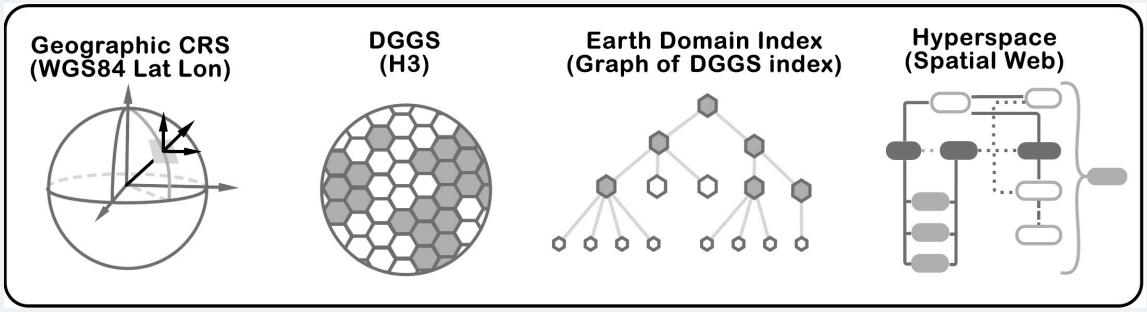
Hyperworld models with the ability to go between graph and coordinate spaces provide a basis for new forms of semantic interoperability. The UDG KG contains a graph of ENTITIES which could be alternatively mapped using a vector space and Hyperworld. Traditional geography is useful in supporting the development of alternative representations. Using traditional geography helps users map (conceptually) current reference systems with which they may be familiar/use to hyperspace

reference systems. This ability to go between UDG KG graph and UDG Hyperspace coordinates may provide a basis for semantic interoperability.

6.3.5. Geography as both geometry and topology

For Geographic DOMAINS, the hyperspace is defined using a geographic Coordinate Reference System (CRS) or a cellular space with an associated graph. Geographic space can be modelled as a particular kind of cellular space, called a *Discrete Global Grid System* (DGGS), which typically models both three-dimensional physical space plus one time dimension. DGGSs therefore provide the Spatial Web with an approach for embedding geographic space into cellular or graphy hyperspace. The figure below illustrates a framework for the coordination of geographic DGGS and CRS with Spatial Web hyperspace. Latitude and Longitude are examples of location in a CRS. Alternatively, the ECEF Geographic CRS uses 3 orthogonal axis located at the center of the Earth.

FIGURE 21: Embedding geographic space in hyperspace



For Concept DOMAINS, the use of coordinates and graphs can be extended from geographic locations to conceptual locations. The motivation for using a coordinate location for conceptual DOMAIN will be become apparent through spatial embedding.

6.3.6. Generalize interfaces to hyperspaces

It would be beneficial to coordinate interfaces that access data represented in differing classes of hyperspace. Category theory defines a set of concepts to uniformly treat the multiple concepts of hyperspace: coordinate spaces, cellular spaces, conceptual spaces, and vector spaces.

Category theory is a system of interfaces that are learned once, and then reliably applied across scientific fields. Originating in abstract mathematics, specifically algebraic topology it thoroughly permeates, these fields: systems theory, bayesian learning, and information theory and probability. The theory can express functions that process structured data from computer science (e.g. lists and trees) and behave in stateful ways like automata. [21]

The UDG will link a variety of data types: vector databases, digital twins with geometric models in coordinate spaces, the .Earth index for all locations on the globe, activities

and contracts performed by agents, etc. These items in the UDG each have different computer science representations. The UDG needs a common approach to managing the various data structures. Category Theory — categories, functors and natural transformations — can be a basis for the UDG.

6.3.7. Hyperspace geography atlas

A traditional atlas is a collection of maps, that serves as a comprehensive geographical reference. Atlases are used to study geography by displaying physical, political, and thematic maps that show landscapes, country borders, and other data like population density, climate, or economic activity. This allows for the study of patterns and distributions across different regions or the entire world.

A UDG Atlas will define regions of the UDG that have semantic and/or spatial similarity. A UDG Atlas then becomes a basis for the distributed replication of the large knowledge base that is the UDG.

Many DOMAINS (such as the Planet Earth) will be used commonly by many DOMAINS that have global operations. A Registry of commonly used DOMAINS is created similar to a traditional atlas.

A DOMAIN can create a relationship to a publicly registered entity internally, such as the government of The United Provinces of North America (a fictional entity) creating a reference to Planet:Earth called Planet:Earth:UPNA that can then be used by any specific subnetwork under the UPNA umbrella. Planet:Earth may contain some specific metadata (how large the planet is, for instance), but this metadata will be fairly limited. Each Node administrator would then create local definitions relevant to the node in question. This makes it possible to query for specific public entities across nodes while decentralizing definition.

DOMAINS may be registered in the Spatial Web Registry as Atlas term for use by any DOMAIN.

If a DOMAIN already exists, then the registrants could “attach” their entity to the existing canonical one so that when searches are made over a range of nodes by querants, the synonym searches can identify which nodes have some version of Planet:Earth.

Inversely, if a local synonym is known on a given node, this can be used to determine what the canonical entity SWID is for that node.

EDITORIAL NOTE

This serves two purposes. The first is, as mentioned, discovery; finding a way to find a particular resource then extracting the relevant metadata from multiple nodes that each have that resource. The second is more political — toponyms and extents of places in particular tend to be contentious. This system makes it possible to provide the minimal definition identifying a resource, but then to allow different SW Node claimants to make their own

claims about that resource that can then be used to establish provenance.

6.3.8. Requirements and recommendations

TBD

6.4. Hyperspace dynamics

6.4.1. Concept: Movement in UDG

- Represented in hyperspace
- Trajectories and activities
- Pose, geopose

6.4.2. Design: thing states

1. Energetic physical systems
2. State machines
3. Trajectory of an individual
4. Group hamiltonian

6.4.3. Design: agent locations

1. Route
2. Agent based population

6.4.4. Design: agent Ideas, concepts

1. Etymology: changed in meaning over time
2. Worlds of ideas and how they evolve

6.4.5. Spatial embedding of motion

- BEHAVIOR-1K? BEHAVIOR-1K is a comprehensive simulation benchmark for human-centered robotics. Compared to its predecessor, BEHAVIOR-100, this new benchmark is more grounded on actual human needs: the 1,000 activities come from the results of an extensive survey on "what do you want robots to do for you?". It is more diverse in the type of scenes, objects, and activities. Powered by NVIDIA's Omniverse, BEHAVIOR-1K also achieves a new level of realism in rendering and

physics simulation. We hope that BEHAVIOR-1K's human-grounded nature, diversity, and realism make it valuable for embodied AI and robot learning research.

- Behavior Domain Definition Language BDDL is a predicate logic-based language inspired by, but distinct from, the Planning Domain Definition Language [1]. It defines each BEHAVIOR activity definition as a BDDL problem, consisting of of a categorized object list (:objects), an initial condition that has only ground literals (:init), and a goal condition that is a logical expression (:goal).
- Catalyzing a virtuous cycle of seeing, learning and doing
- Spatial intelligence models the world and reason about objects, places, and events in 3D space and time
- Spatial intelligence is catalyzing robotic learning
- Key component for any embodied intelligence systems that needs to understand and interact with 3D world.
- HAI BEHAVIOR project will do for video activity labeling what ImageNet was to image feature labeling

6.4.6. Requirements and Recommendations

6.5. Hyperspace knowledge examples

6.5.1. Hyperspace geography applications

This clause describes applications use of Hyperspace Geography to various types of knowledge.

The main aim is for the UDG to serve as a medium for collective intelligence in the Spatial Web. As a medium it is the channels and methods through which ENTITIES in the Spatial Web send and receive HSTP messages. It plays a crucial role in how the message is perceived, interpreted, and understood using HSML.

6.5.2. Geospatial and graph domains for urban energy grid

The Urban digital twin / Smart city applciation scenario (Spatial Web clause 5.3.7) shows how the Spatial Web enables digital twin technology for addressing urban sustainability with a focus on energy. Using urban energy system modeling and analysis developed in the Spatial Web multi-scale cognitive computing ecosystem will benefit next generation cities and the globe. Considering climate change it is vital that more cities deploy energy UDTs to address energy consumption and climate sustainability.

(Spatial Web clause 6.3.7) defines DOMAINS required for the energy scenario.

- The Metropolis Digital Twin DOMAIN provides the geometry for the Metropolis using vector space. The Metropolis Digital Twin domain is a geographic Domain. Domain [metropolis] is all regions near the surface of the Earth that are managed by the Metropolis government.

- The Regional Energy Grid DOMAIN defines the connectivity of nodes in the energy grid using a graph space. The Regional Energy Grid Domain is a conceptual graph or knowledge base for the concepts and operations of a regional energy grid.
- The Metropolis Digital Twin—Energy Model DOMAIN aims to provide a common interface on the Metropolis space model and the Energy grid graph model. The Metropolis Digital Twin—Energy Model Domain is holonically related to the Metropolis Digital Twin Domain and the Regional Energy Grid domain.

6.5.3. Improving KGs using geospatial features

Integration of geospatial features into knowledge graphs can enable more precise characterization of knowledge dynamics across temporal and geographic dimensions.. The absence of geospatial information significantly undermines the value of such knowledge representations. Spatial embedding can coordinate relational semantics with spatial-temporal geography models. A geometry-aware embedding space that dynamically fuses spatial grids and temporal slices through spatio-temporal states, enables robust reasoning on heterogeneous graphs ([8]).

6.5.4. Global Earth observation models

Massive Earth Observation (EO) archives containing hundreds of petabytes of dataoffer a huge potential for understanding the Earth's resources and potential hazards tosecurity while also addressing societal challenges ([element_84_2025]). There is a growing need for efficient vector representations of the underlying raw data. The approach of extracting feature representations from pretrained deep neural networks is a powerful approach that can provide semantic abstractions of the input data. Several projects are creating global and dense embedding datasets released openly and for free. ([10]), ([11]).

6.5.5. Telecoupling

Telecoupling implicilty uses hyperspace for identifying connections between physically distant systems. Telecoupling exposes socioeconomic and environmental interactions between distant coupled human and natural systems. The integrated framework of telecoupling examines flows of information, energy, matter, people, organisms, and other things such as financial capital and goods and products around the globe. It pinpoints causes and effects arising from engagement of diverse agents in the global sphere. It enables recognizing trade-offs between local and global sustainability and the need for multi-level management and governance solutions. Telecoupling can help systematically expand from a focus on specific places separately to human-nature interactions across distant places. (see ([22]) and ([23]))

6.5.6. Emboddied cyber-physical agents

Living systems balance energetic efficiency with the capacity for path-dependent effects. ([blattner]) defines a geometric framework that models embodied agents as hierarchies of manifolds linked by projections from physical states to cognitive representations and onward to intentions. The framework consists of three manifolds/

subspaces — Physical, Cognitive, Intentional — along with functions for mapping between the spaces; mappings which constrain the paths in each space. The framework provides a geometric language linking embodiment, memory, and energetic cost, yielding testable predictions and design guidelines for biological and robotic systems. This framework might provide necessary rules for structures in the UDG which could be utilized for desired UDG-based behaviors of cyber-physical agents,

6.5.7. Active inference for collective intelligence

Ecosystems of intelligence are core to the Spatial Web. Intelligence can be understood as the capacity to accumulate evidence for a generative model of a sensed world. Formally, this corresponds to maximizing model evidence, via belief updating over several scales: i.e., inference, learning, and model selection Friston et al. ([7]). Operationally, this self-evidencing can be realized via message passing or belief propagation on a factor graph. This same imperative underwrites belief sharing in ensembles of agents, in which certain aspects of each agent's generative world model provide a common ground or frame of reference. Active inference addresses this ecology of belief sharing—leading to a formal account of collective intelligence that rests on shared narratives and goals. Based on this perspective, levels of multi-scale cognitive computing in a Spatial Web ecosystem of Agents can be defined. The ensuing levels can be used for governance in the Spatial Web.

6.5.8. Requirements and recommendations

TBD

6.6. UDG size estimate

6.6.1. Internet scale

(<<IEEE_2874_2025>)> requires the Spatial Web be designed for internet scale. Internet scale has changed as the Internet has evolved and grown. Internet Protocol version 4 (IPv4) address size was found to be inadequate. IPv6 address space was greatly increased to meet future demands.

The Spatial Web size estimates listed below are currently less than the IPv6 scale. While the Spatial Web estimates below are justified, the need for extensibility of the Spatial Web should be anticipated.

6.6.2. Summary of Spatial Web estimate

Estimates of the UDG size are given below. Here is a summary:

- UDG is a hypergraph which contains all relationships between all known SWIDs in the Spatial Web.
- How many SWIDs might there be in the Spatial Web: approximately 10^{14}
  - Current large knowledge graphs (Microsoft): is on the order of 10^{12} entities
  - Wikipedia is $6*10^{7}$

  - Embedding H3 DGGS into UDG that would be 10^{14} hexagons
- Query performance
  - H3 grid/hash queries by Databricks
  - Geographic query bigger than a set of buildings in the US is too big. $2.5*10^{8}$

6.6.3. Existing large knowledge graphs

The UDG is similar to existing, proprietary knowledge graphs.

(Industry-scale knowledge graphs: lessons and challenges) provides a summary of large existing knowledge graphs. These knowledge graphs may have grown a handful of orders of magnitude since 2019 when the size of each was:

- Microsoft's knowledge graph: approximately 2 billion primary entities, $2*10^{9}$ entities as of 2019.
- 2023 estimate of large knowledge graphs 10^{12} entities

As of September 2025, the contents of Wikipedia were:

- English articles 7,063,252 (roughly $7*10^{7}$)
- Total wiki pages: 64,128,394

6.6.4. Geographic H3 index

The H3 geospatial indexing system is a discrete global grid system (see Sahr et al., 2003) consisting of a multi-precision hexagonal tiling of the sphere with hierarchical indexes. H3 was developed to address the challenges of Uber's data science needs. H3 is now open source under the Apache 2 license.

H3 is a hierarchical geospatial index. Hexagonal cells at level 12 have an area of 90 dm2 and there are $5.7*10^{14}$ hexagons

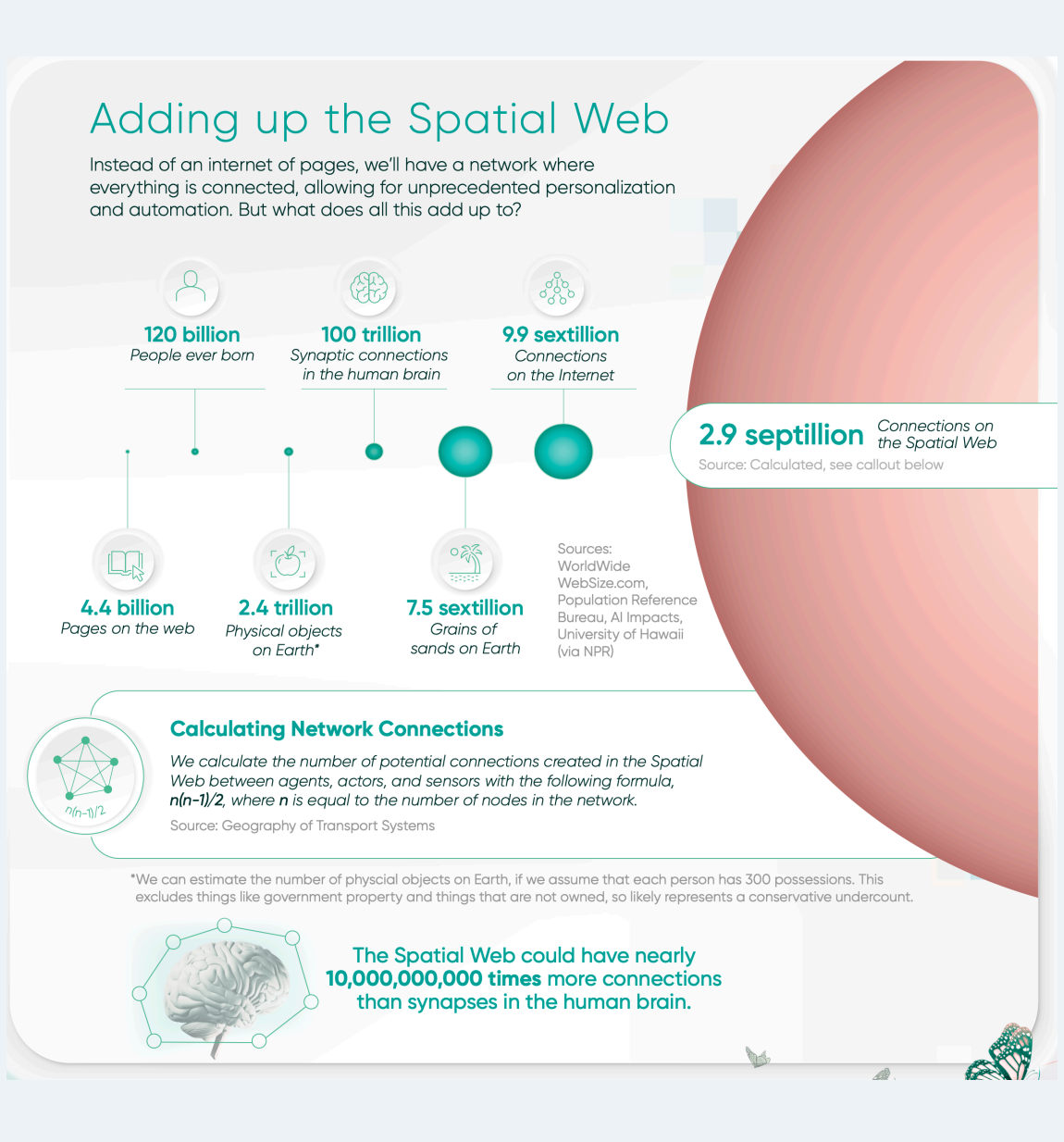
6.6.5. VERSES estimate of the Spatial Web

- 2.4 trillion objects on the Earth. $2.4*10^{12}$
- 9.9 sextillion connections on the internet $9.9*10^{21}$
- 2.9 septillion connections on the spatial web $2.9*10^{24}$
- Square root of a septillion = approximately 10^{12}

EDITORIAL NOTE

this figure needs to be simplified.

FIGURE 22: Estimate of the number of entities in the Spatial Web UDG



6.6.6. Requirements and recommendations

TBD

7. UDG as a network of registries

7.1. Spatial Web registries

7.1.1. Domain membership, identity and registries

Registries in the Spatial Web are of two types, performing two distinct functions:

- A Domain Registry is used to record the membership of an ENTITY in a DOMAIN. (See (5.3.4))
- A Verifiable Data Registry (VDR) provides the information for a SWID document. (VDR is as defined in (W3C did-core))

The two functions for Spatial Web registries are distinct, i.e., a Domain registry is not the source of SWIDs.

Because DOMAINS may be members of multiple upper domains, and DOMAINS may move between upper domains; DOMAIN identity may proceed registration.

Registration of a DOMAIN in an upper DOMAIN results in the membership being recorded in the Registry and a credential being issued to the registered ENTITY.

The Spatial Web registries follow these design goals and properties:

- All ENTITIES (including CREDENTIALS) have SWIDs.
- Uniqueness of SWIDs using the did:swid method is ensured by a VDR. The VDR may be part of the Spatial Web Registry operated by the SWA or in a VDR separate from the Spatial Web Registry
- SWIDs may be created by a DID Issuer at behest of the ENTITY and not recorded in a centralized registry. o The Spatial Web will include ENTITIES with DIDs created without interaction with any particular authority.

These properties are elaborated in the Spatial Web Identifiers (SWIDs) and SWID Documents Implementation Specification (SWF STD-4:2025) and the DID Method for the Spatial Web (did:swid) Implementation Specification (SWF STD-5:2025).

7.1.2. Spatial Web Registry

The Spatial Web Registry contains both a Domain Registry for Top and Public DOMAIN registration and a VDR associated with issuing SWIDs using the did:swid method. The Spatial Web Registry is managed by the Spatial Web Registration Authority.

EDITORIAL NOTE

In IEEE 2874-2025, SWA is the acronym used for the Spatial Web Registration Authority. In this document, both SWA and SWRA are used for the Spatial Web Registration Authority, but SWRA is preferred.

The composite Spatial Web registry process is in two parts: SWID creation; Domain registration. Each part is shown in a figure.

- A registrant request to create a SWID for an ENTITY using the Spatial Web Registry; The Registry responds with a SWID created using the did:swid method
- A registrant request to add the ENTITY as a member of a DOMAIN; The Registry responds with a credential

FIGURE 23: Creation of a did:swid by the Spatial Web Registry

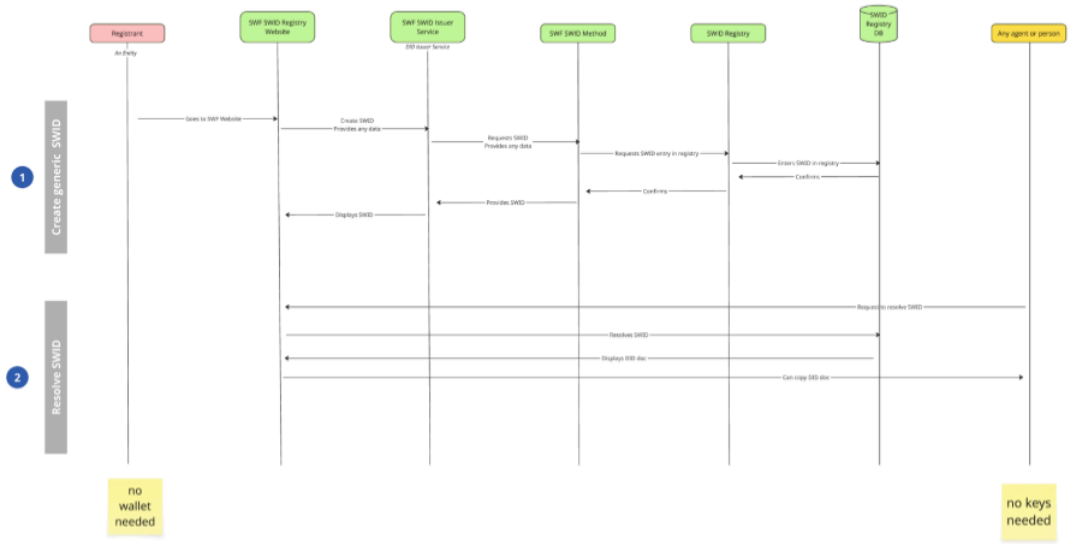
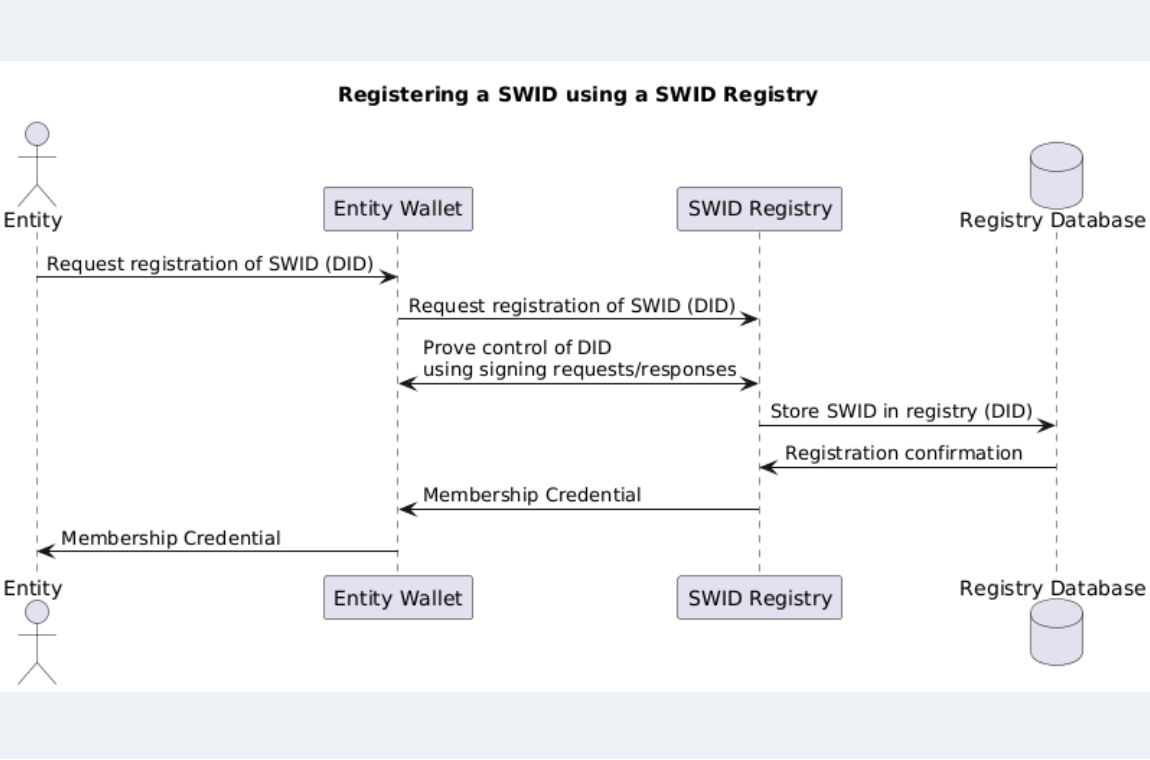


FIGURE 24: Registering an ENTITY as a member of a DOMAIN



7.1.3. Governance

The Spatial Web Registration Authority (SWRA or SWA) is the organization which administers the Spatial Web Registry. The SWRA may designate one more entities to manage the creation of SWIDs and registry of Domain membership. The SWA, on behalf of the public, shall establish general rules for changes to the UDG, such as Domain Authority, allowable Domain names, Domain claim dispute resolution, cost of registration, and restrictions on the addition or deletion of names.

7.1.4. User functions

The Spatial Web Registry shall provide user facing functions including:

- Resolving SWIDs
- Disclosure of ENITITIES registered as members of a DOMAIN

7.2. Public and Top Domains

7.2.1. DOMAINS in general

DOMAINS provide structure which matches the structure of people, places, concepts, and things as nested in nature as parent, child, and sibling relationships.

The purpose of a DOMAIN is to offer a conceptual area in which to add definitions or behavior with HSML-compliant software that allows reality to be described from multiple perspectives and correlated hierarchies. It can be used to provide a single distinct container for things such as a geolocated physical object or a concept without having to be the thing itself.

In the inhabitable world, a domain refers to the geopolitical territory governed by a single entity or government or a geophysical region characterized by a specific feature, type of growth or wildlife, etc. E.g. The domain of the United States, the State, or a person's home.

In the human world, a domain refers to a cultural field of knowledge or influence or range of personal knowledge, or responsibility. E.g. The domain of science, art, or morals.

In the digital world of the Internet, a domain is a set of addresses that refers to the category or geographical area that an Internet address belongs to. E.g. Internet or web domains.

Although each definition of a domain has a slightly different meaning, they all refer to a type of location or "hyperspace" that defines a range of knowledge, activity, or interest, over which someone has control, influence, or rights.

Domains, as defined in IEEE P2874 clause 6.6 about the Spatial Web ontology, are Entities with identity maintained through time, endowed with rights and credentials as defined in the Hyperspatial Modeling Language (HSML), a framework for describing objects, relationships, and governance rules in the Spatial Web and how it links domain structures to policies, credentials, or other metadata.

Functionality: Domains provide structure, often parallel structures, for the existence of people, places, and things, including concepts, nested in nature, that allow for parent, child, and sibling relationships.

Purpose: A purpose of a DOMAIN is to offer a conceptual area in which to add definitions or behavior with HSML-compliant software that allows reality to be described from multiple perspectives and correlated hierarchies. It can be used to provide a single distinct container for things such as a geolocation, physical object, or a concept without having to be the thing itself.

The Spatial Web maintaining core repositories, taxonomic concepts, activity components and schemas. These items are avaialble for any DOMAIN to reuse, while the Domain may create their own definitions, they can use the public Spatial Web concepts to provide core provenance and structure.

The Spatial Web registry provides a clearinghouse for identifying and classifying public domains, using the Spatial Web UDG Taxonomy (and the corresponding hsml:hasTopic and related predicates) to help to identify relevant content.

7.2.1.1. Public and private domains

Public Domains have no owner and are for the public good like a wiki commons. SWF acts as the steward for all public domains. Examples: knowledge concepts like baseball, traffic laws, lidar, weather, applied physics.

Private Domains are domains registered by individuals, organizations and their agents. Examples: Los Angeles Dodgers, California DMV, an autonomous car, a weather sensor, Johns Hopkins University Applied Physics Laboratory.

Top Domains are the highest Private domain of a domain hierarchy and managed by the Domain Authority, which governs the entire domain. Examples: Major League Baseball, United States, Coca-Cola Company, NASA, Johns Hopkins University.

All DOMAINS are conrolled by a Domain Authority, which is a Spatial Web ENTITY that manages membership in the domain and defines norms for the domain.

7.2.1.2. Top Domains & UDG

A top domain registers just one domain but can be linked to all relevant Public Domains in the UDG. This ensures discoverability, governance inheritance, and interoperability without the need for multiple SWIDs.

7.3. Hierarchical domain registries

7.3.1. Domain Authority role

The Domain Authority plays five roles simultaneously in the Spatial Web:

- As a Domain Authority (DA), it governs related subdomains, controls its Domain Governance, and may issue and govern SWIDs.
- As Registrant, it registers its Domain
- As a Registry, it maintains all subdomains under its Domain,
- As a Registrar, it issues SWIDs for products, operations, personnel, agents, geolocations, etc.
- As an Index, it organizes and makes domain-related information discoverable in the UDG.

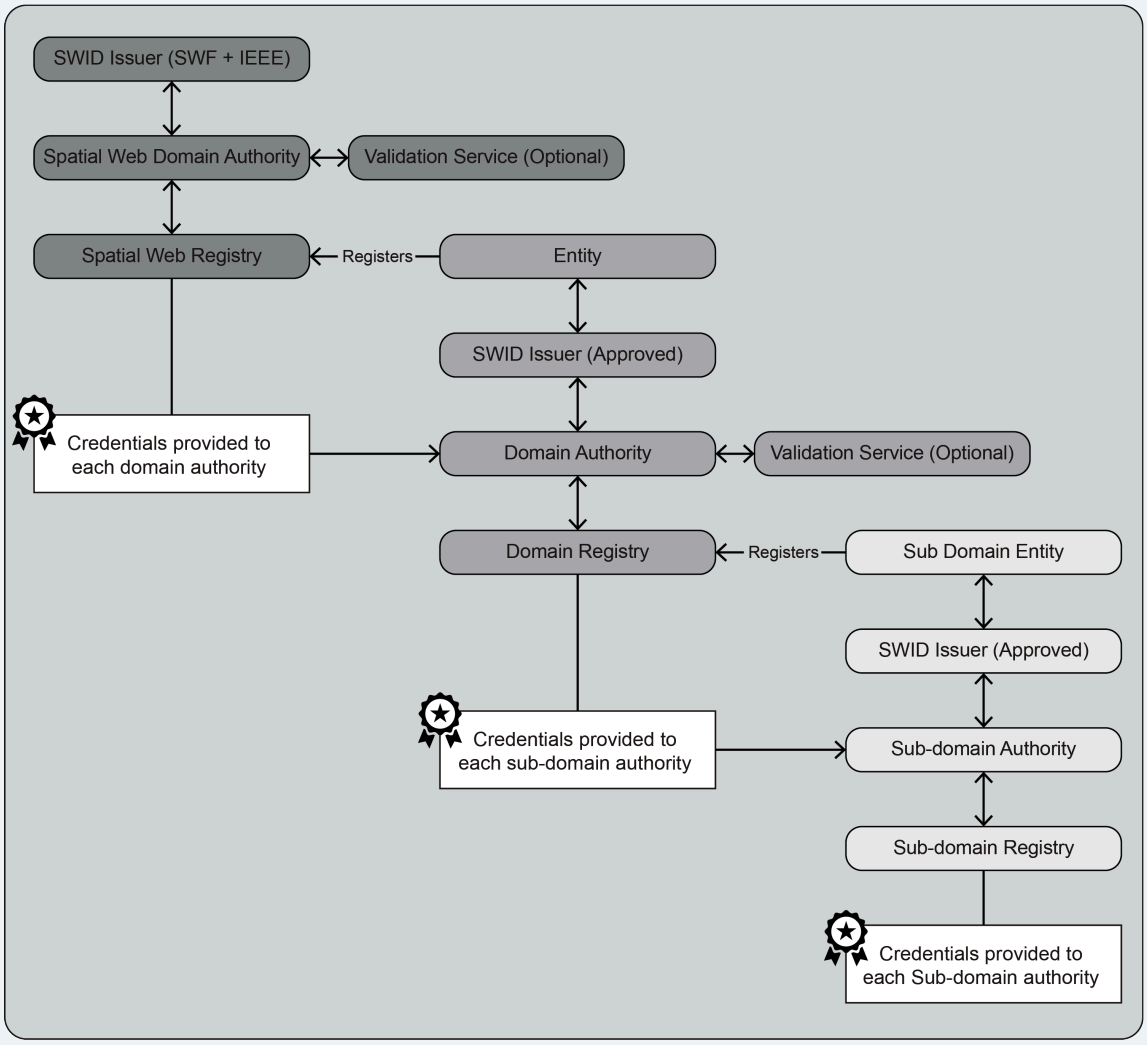
7.3.2. Domain-specific registries

Domain Authorities may establish a domain-specific registry, i.e., a domain registry that is defined specifically for the entities of interest to the Domain Authority. A domain-specific registry may have the same functions as the Spatial Web Registry (creation of did:swids and registering membership of domains). The difference is that the scope of a domain-specific registry is for a specific domain whereas the Spatial Web Registry is the registry for all registered DOMAINS.

7.3.3. Registry of Registries

A domain-specific registry may be registered in the Spatial Web Registry. As this registration may propagate through several layers of domain hierarchy, a hierarchy of registries is created.

FIGURE 25: Hierarchy of registries in the UDG



7.4. Baseball league domain example

EDITORIAL NOTE

STD-1 generalized the MLB example to a generic baseball league.

7.4.1. MLB example

Major League Baseball is an American sport that has 30 professional teams organized in several sub and regional leagues. The domain of baseball has many public domains such as bat, pitcher, uniform, strike out, base hit, umpire, baseball diamond, baseball stadium, etc. These public domains can be linked to and used by any top-level domain baseball organization anywhere in the world, who can then use the domain model as

the base structure domain model, modifying any of the rules or attributes for their local domain. For example, in the public domain there are several kinds of baseball bats (aluminum, wood, composite) that can be used to play the game or as a weapon. However, in MLB, there are very specific attributes that can be used for a baseball bat.

Below is an example of some of the relationships between the Public Domain of baseball, the MLB Top Domain and its various Sub Domains. MLB registers one top domain (mlb.spatialweb) linking it to all relevant Public Domains in the UDG. This ensures discoverability, governance inheritance, and interoperability without needing multiple SWIDs.

FIGURE 26

```
* Public Domains
  * America
    * Sports
     * Baseball
       * Bat
       * Pitcher
       * Uniform
* Top Domains
  * Major League Baseball
    * Sub Domain
      * American League
      * National League
    * Sub Sub Domain
      * East
      * Central
      * West
      * East
      * Central
      * West
    * Sub Sub Sub Domain
      * Baltimore Orioles
      * Chicago White Sox
      * Athletics
      * Atlanta Braves
      * Chicago Cubs
      * Arizona Diamondbacks
      * Boston Red Sox
      * Cleveland Guardians
      * Houston Astros
      * Miami Marlins
      * Cincinnati Reds
      * Colorado Rockies
      * New York Yankees
      * Detroit Tigers
      * Los Angeles Angels
      * New York Mets
      * Milwaukee Brewers
      * Los Angeles Dodgers
      * Tampa Bay Rays
      * Kansas City Royals
      * Seattle Mariners
      * Philadelphia Phillies
```

```
            * Pittsburgh Pirates
            * San Diego Padres
            * Toronto Blue Jays
            * Minnesota Twins
            * Texas Rangers
            * Washington Nationals
            * St. Louis Cardinals
            * San Francisco Giants
```

7.4.2. MLB's Multi-Role Responsibilities in the UDG

TABLE 4: MLB roles in the UDG

Role	Definition	MLB's Function
Domain Authority (DA)	A governing body that sets rules for its domain and subdomains.	MLB acts as the DA of mlb.spatialweb, defining governance, credentialing, and policies for baseball-related entities.
Registrant	An entity that registers a domain and owns its SWID.	MLB is the registrant of mlb.spatialweb, meaning it officially holds the domain.
Subdomain Registry	A system that maintains records of registered subdomains and SWIDs.	MLB acts as the registry for all baseball-related subdomains registered to it, maintaining an authoritative list.
Subdomain Registrar	A service that registers subdomains and issues SWIDs	MLB serves as the registrar for subdomains like yankees.mlb.spatialweb, dodgers.mlb.spatialweb, stadiums.mlb.spatialweb, etc.
Index	A searchable index of a domains and their subdomains within the UDG.	MLB maintains an index (or assigns the right to its subdomains) of its subdomains i.e. all teams, players, stadiums, and equipment, making them searchable.

7.4.3. How It Works: MLB's Registration & Linking Process

1. MLB registers its SWID mlb.spatialweb.
2. A single SWID is assigned SWID:web:swf.com:domain:mlb
3. MLB becomes a registrant of mlb.spatialweb.
4. MLB selects relevant Public Domains (Linking).
5. MLB is given a "Select All That Apply" option to associate with relevant parent domains.
6. Linking occurs to connect MLB's SWID to:
   - baseball.spatialweb (General Baseball)
   - baseball-pitcher.spatialweb (Baseball pitcher)
   - bat.spatialweb (Bats as it relates to baseball)
   - american-sports.spatialweb (American Sports)
   - uniform.spatialweb (Uniforms as it relates to baseball)

7. The UDG establishes these connections via graph-based relationships.
8. MLB's SWID is indexed under all selected domains.
9. Queries from any relevant domain will correctly return MLB and its Sub Domain.
10. Domain model inheritance
    - MLB inherits the attributes, schema and model of the parent Public Domain and can then make any necessary modifications required for use in the MLB.
    - This ensures a common data model and interoperability that can be understood by any outside entity.
11. Governance Inheritance & Discovery
    - MLB inherits governance and rules constraints from each linked Public Domain.
12. Search engines & AI agents can traverse the UDG graph to find MLB and its Sub Domain through any relevant path.

7.4.4. Example: How UDG Queries Would Work

Because MLB registers one top domain (mlb.spatialweb) and links it to all relevant Public Domains in the UDG, MLB can be discovered by searching adjacent public domains.

- Query 1: "Show all professional baseball leagues"
- Searches in: sports-leagues.spatialweb → Finds MLB (via smart link)
- Query 2: "Find all governing bodies in baseball"
- Searches in: baseball-organizations.spatialweb → Finds MLB (via smart link)
- Query 3: "Show all baseball businesses"
- Searches in: sports_business.spatialweb → Finds MLB (via smart link)

8. UDG as a social network

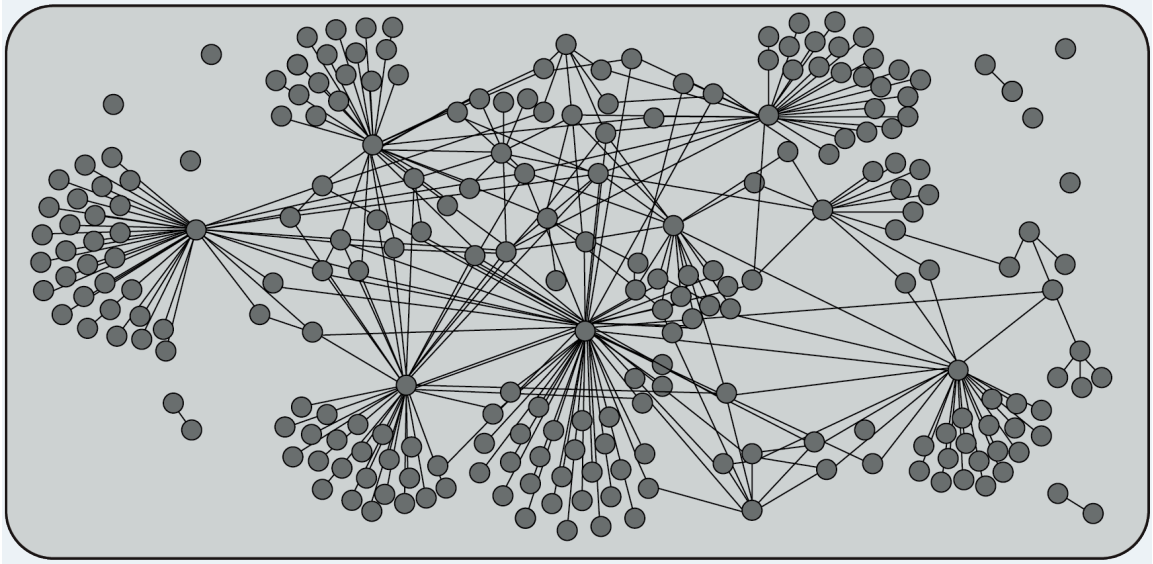
8.1. Idea flow in social networks

8.1.1. Social intelligence

Groups have a collective intelligence that is mostly independent of the intelligence of the individual participants. The UDG a universal digital commons is a form of universally distributed intelligence, constantly enhanced, coordinated in real time, and resulting in the effective mobilization of skills.

The UDG is a social network of ENTITIES in the Spatial Web. In particular AGENTS will interact with other ENTITIES including other AGENTS to determine the state of the environment, to determine knowledge about other ENTITES, to coordinate ACTIVITIES. The structure of relationships in the UDG Social viewpoint are in many cases more important than the knowledge content of ENTITES in the UDG KG.

FIGURE 27: Social clusters in the UDG



Social intelligence is composed of individual intelligence, but social intelligence quality is distinct from individuals ideas. Social intelligence is necessary as individual intelligence is more limited.

"The knowledge of the circumstances of which we must make use never exists in concentrated or integrated form, but solely as the dispersed bits of incomplete and frequently contradictory knowledge which all the separate individuals possess" ([27]).

The design approach for social intelligence in the UDG does not guarantee results or solutions to all problems, but it creates an institutional infrastructure that unleashes more local knowledge and individual initiative, and therefore does in fact produce better results and more solutions.

The UDG employs social networks to improve idea flow between AGENTS. Social physics provides the science for the design. The design choices take into account social issues like autonomy, privacy, self-determination. These design choices are crtical to avoid creating a virtual Panopticon.

8.1.2. Social idea flow

Idea flow is the engagement and belief propagation which spreads new behaviors through a social network. Idea flow may be conceptualized as exploration to harvest new ideas followed by engagement with peers to sift through those ideas and convert the good ideas into habits.

Social physics is a quantitative social science that describes reliable, mathematical connections between information and idea flow on the one hand and people's behavior on the other ([26]). Social physics seeks to understand how the flow of ideas and information translates into changes in behavior. It provides a causal theory of how social structures confer evolutionary advantage. An alternative term for social physics is computational social science.

An aspect of idea flow is that of pooling ideas a "wisdom of the crowd" judgment will be better than individual AGENT judgements. The effectiveness of idea-pooling for estimation problems works as long as there is no initial social interaction. It assumes that all the people in the crowd will act independently. When social interaction occurs, AGENTS begin influencing other AGENTS, and that may result in panics, bubbles, and fads. In humans, the social learning strategy of feeding back the best current idea—that is, a constrained, artificial sort of social interaction that interleaves periods of idea harvesting with periods when experts evaluate the ideas—produces a wisdom of the crowd effect that works even for small groups. This can be effective in the Spatial Web as well.

Social Learning is measure of idea flow. It is the chance that a person's behavior will change when a new idea has appeared in their extended social network.

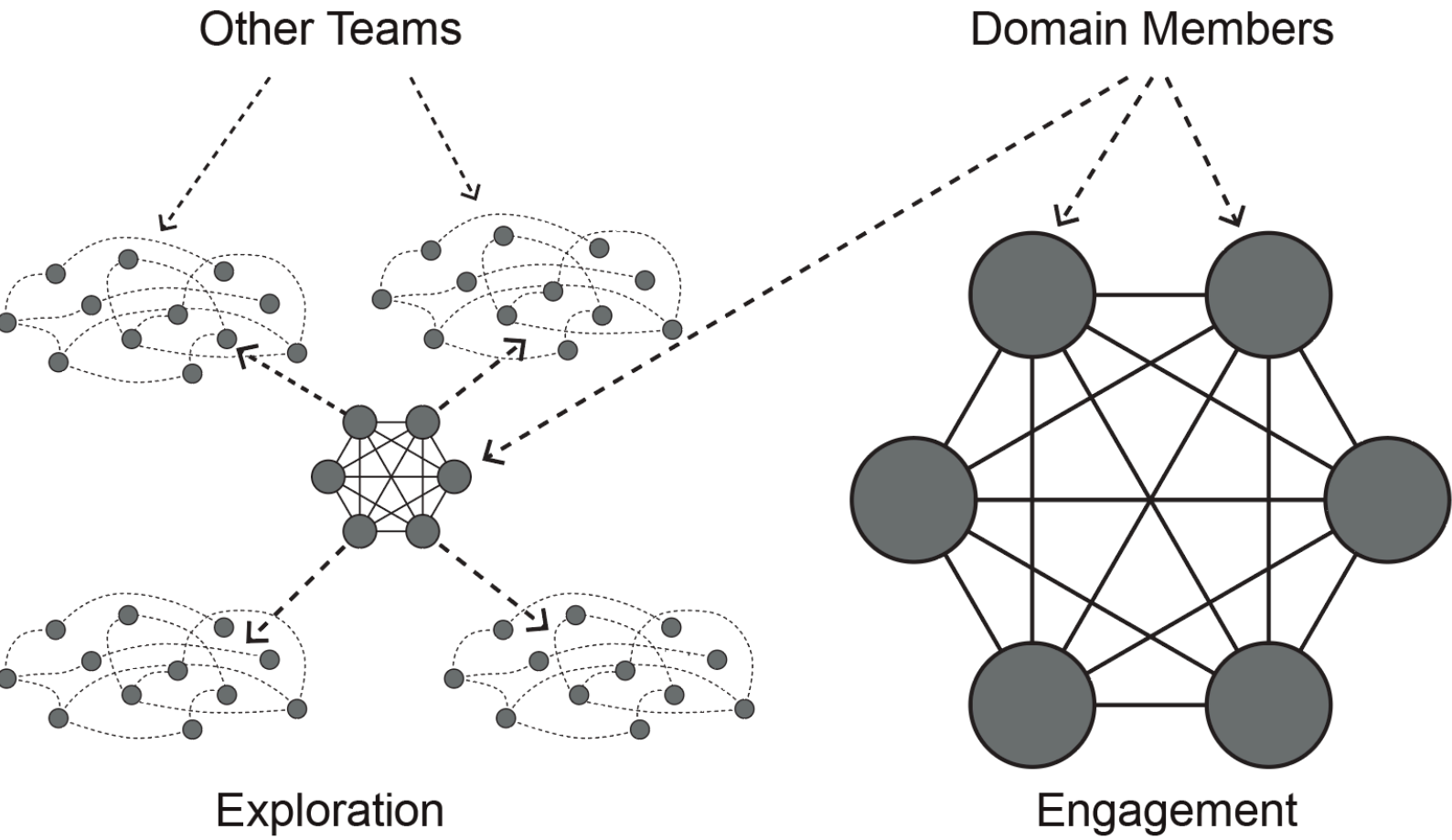
Exploration refers to the use of social networks in harvesting ideas and information. Exploration is the part of idea flow that brings new ideas into a work group or community. Social learning, Diversity and Contrarians are important for effectie exploration.

In order to acheive Enagement in the UDG, AGENTS must interact, cooperate and build trust.

One measure of effectiveness of idea flow is creativity. Creative output depends strongly on two processes: idea discovery (exploration) and the integration of those ideas into new behaviors (engagement). Exploration outside of the group, together with their engagement within the group. Exploration and Engagement networks are shown in (Figure 28).

The UDG and AGENTS are designed to improve Idea Flow. With increased engagement comes an increase in opportunities for social learning, for sharing vital resources such as tacit operational knowledge and successful work habits. Thois results in improved collective intelligence peformance. Collective intelligence is not strongly correlated with the average or maximum individual intelligence of group members but is correlated with the average social sensitivity of group members, the equality in distribution of conversational turn-taking, and for human groups, the proportion of females in the group. ([48])

FIGURE 28: Network structures for exploration vs engagement (Source: ([26]))



8.1.3. The Mathematics of Social Influence

An "influence model" can assit in understanding social influence. The model estimates how much the state of one AGENT affects the state of another AGENT in the Spatial Web. The model can recover estimates of influence, generate results that are consistent with other measures of social networks, and uncovers important shifts in the way states may be transmitted between actors at different points in time. ([29])

- Observable signals: O
- Agent hidden states: s
- Influence matrix: R

FIGURE 29: Inflence model equation (Source: ([29]))

$$\text{Prob}(h_t^{(c')} | h_{t-1}^{(1)}, \ldots, h_{t-1}^{(C)}) = \sum_{c \in \{1,\ldots,C\}} \underbrace{\mathbf{R}_{c',c}}_{\text{tie strength}} \times \underbrace{\text{Infl}(h_t^{(c')} | h_{t-1}^{(c)})}_{\text{influence } c \to c'},$$

This model takes raw observations of behavior and provides social network parameters needed to get a numerical estimate of idea flow, which is the proportion of users who are likely to adopt a new idea introduced into the social network.

Idea flow takes into account all the elements of the influence model: network structure, social influence strength, and individual susceptibility to new ideas.

The influence model allows for examination of social roles: protagonist, attacker, supporter, neutral, and so on, in small groups, and in organizations,

8.1.4. Groups, Organizations and Bureaucracy

Social networks are a set of relationships connecting individual AGENTS, organization DOMAIN, or groups of AGENTS and/or Organizational DOMAINS (perhaps modeled as a DOMAIN)

Networks can come in multiple forms:

- Primary groups: Small, intimate groups that are central to an individual's identity and provide emotional and intellectual support. Examples: Family, close friends, and long-term work or church groups. Characteristics: Long-term and emotionally vital.
- Secondary groups: Larger, more impersonal groups that are often formed for a specific purpose or goal. Examples: A class, a workplace team, or a large organization. Characteristics: Often temporary and task-oriented.
- Formal organizations: Large, structured groups that are designed to achieve specific objectives. Characteristics: Usually have a hierarchy, a professional leadership, and written records.
- Informal organizations: Looser, less structured groups that emerge within formal organizations, such as social friendships between coworkers.
- Bureaucracy: A formal system of organization characterized by a clear hierarchy, division of labor, and rules.

8.1.5. Requirements and Recommendations

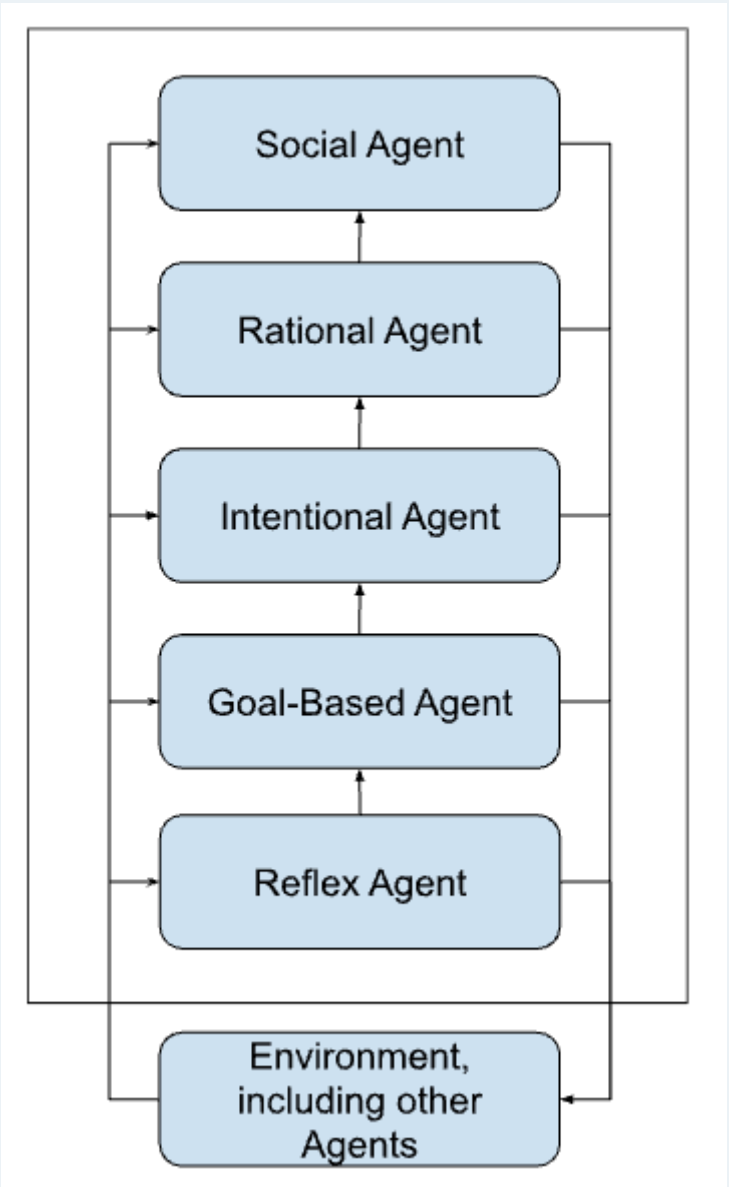
TBD

8.2. Social Agents; shared goals and activities

8.2.1. Agent type: social

The Spatial Web Agent Framework Design Specification ([swf_std_6]) defines several AGENT types for the purpose of explaining examples of agency used in the Spatial Web. The agent types are distinguished by the functions they perform which are visible through behavior or statements of the agent's developer. The canonical Agent Types are informative as the following:

FIGURE 30: Canonical Agent Types as a hierarchy.



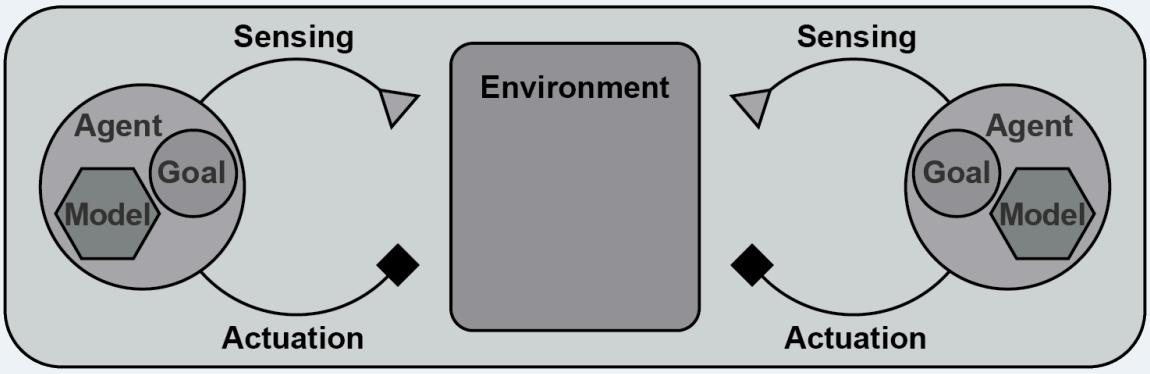
Natural social agents provide a basis for describing AGENTS in the Spaital Web. ([32]) describes the traits of socially normative agents. Even early human individuals formed

with other individuals a joint agency. They engaged in socially normative self-regulation which obliges individuals to direct and control their actions not just individually but also to comport with the normative standards of the shared agency in which they are participating. Individuals came together to form joint goal, coordinating roles and collaboratively self-regulating the collaboration. Individuals closest to those with whom they shared the most experiences; derived through joint attention and common ground, in collaborative activities.

Social agents learn the shared habits, norms, and expectations of their culture through immersive participation in patterned cultural practices that selectively pattern attention and behaviour. ([31]) calls this process "thinking through other minds" (TTOM) – in effect, the process of inferring other agents' expectations about the world and how to behave in social context.

Figure 31 depicts the loop between action, sensations, and niche construction that lead to the acquisition and production of cultural habits, and to the inference and learning about other minds. In the Spatial Web, this "thinking through other minds" (TTOM) depends on the exchange of HSML content using HSTP to share cultural practices and associated artefacts that guide the attention (and learning) of AGENTS.

FIGURE 31: Social agent interaction: thinking through other minds (Source: ([31]))



Natural agents and some artificial agents exhibit behavior that implies curiosity.

Curiosity is an intrinsic desire for knowledge. Recent work in reinforcement learning suggests that surprise functions as a reward signal for the curious animal. Reward for surprise can then be shown to amount to a desire for knowledge gain, where knowledge is a cognitive adaptation to reality. This adaptation results in a mental state whose stable existence depends essentially on the truth of its contents; that is, a factive mental state. Curious creatures benefit from an interaction between the prediction-error correction processes of basic learning and the active surprise-seeking force of their curiosity. This internally adversarial interaction accelerates knowledge gain in ways that are helpful for biological creatures in a complex natural world.([33])

Learning, inference, and decision making are processes that resolve uncertainty about the world. Active Inference simulations used to illustrate the emergence of curiosity and insight follow from a single principle: the minimization of free energy (i.e., surprise) or maximization of model evidence. ([34])

8.2.2. Social epistemology

Curiosity causes AGENTS to aquire additional information. Agents interact with other agents in order to gain this information and improve the internal models the agents holds. Social epistemology is how AGENTS learn in interaction with other AGENTS. This social interaction is affected by Network epistemology: how social network structure affects beliefs and knowledge ([28]).

Communities of AGENTS can develop collective intelligence that is greater than the members' individual intelligence. There is a complex relations between individual agent knowledge and collective intellgence. ([28]) defined the Independence Thesis that norms of social epistemology are independent of norms of individual epistemology. Two divergent "folk" notions about epistemic groups are 1. Wisdom of the Crowds: groups can be far wiser than its members vs. 2. The opposite: group think, pluralistic ignorance, echo chambers, information cascades. Some epistemic problems are caused by social structure, not by individually bad behavior. These include group pathologies, like pluralistic ignorance, herding, and polarization, Blindly increasing connectivity can make the group worse rather than better the individuals sometimes less communication is better.

Application of Landscape search problems to benefit social epistemology. The knowledge landscape can be modelled using HYPERSPACE model of conceptual domains. The AGENTS must 'spread out' over the space of possibilities in order to find the best solution. This can be approached using the notions of NK-Landscape to model landscape search' problem. N is the number of propositions that the agent must assign truth values to; K represents how complex are the relevant composite propositions. Harder problems are those with both higher N and higher K.' These models suggest that infrequent social learning is optimal. Imitation reduces the diversity of the search; If every agent imitates on every round, the group ends up being very homogeneous. This is another example of the independence thesis: individuals would prefer to engage in social learning far more frequently than would benefit the group. ([28])

A central feature of many real world social networks is that they feature very low mean path length, the distance between two randomly chosen individuals in the network. The familiar 'six-degrees of separation' is an instantiation of this idea. Mean path length is related to eccentricity, since both are a measure of average distance. Small world property characterizes this network: That two people are separated by a relatively small number of friends, while each person also has relatively few friends. The weak ties those' that tend to connect two different people who are otherwise quite distant in the social network were incredibly important in understanding diffusion processes ([35]).

- If the diffusion is simple, clustering slows down the spread of new ideas,
- if the diffusion is complex, it makes the diffusion possible to begin with.

No single network provides optimal social epistemology in all cases. The UDG needs multiple adaptive methods to situtational networks and objectives for specific communities of AGENTS.

8.2.3. Requirements and Recommendations

TBD

8.3. Objective-normative society

8.3.1. Construction of social reality

Social agents operate in objective-normative worlds: what an agent should do in part is based on objective statements established through social mechanisms. Norms enable shared agencies that comprise both shared experience on a common focus along with different perspectives. Modern human agency operates in a world of objective facts and objective moral values ([32]). The Spatial Web integrates artificial social agents into our shared objective-normative world.

The Spatial Web provides the infrastrcuture to construct a social reality including AI where objective facts exist in the world, that are only facts by human agreement. The Spatial Web enables a virtual reality, a world of agency, intentionality, and other conceptual phenomena, to fit alongsie a world consisting of physical particles in fields of force ([36]). In the UDG an AGENT or DOMAIN acquires status based in part on deontic norms, laws, contracts, and certifications.

8.3.2. Social entities: norms, laws, contracts

The Spatial Web Ontology ([STD-1]) defines ENTITIES essential to construction of social reality: ACTIVITY, CREDENTIAL, CONTRACT. HSML adds NORM as a ENTITY.

- ACTIVITY: A partially ordered set of changes effected by an AGENT. Activities are are performed on, by, in, or with, DOMAINS, including other AGENTS
- CREDENTIAL: A set of one or more claims made by a DOMAIN. Credentials are used for several claims including Identity, DOMAIN relationships, an AGENT'S ability to perform an activity.
- CONTRACT: A binding agreement between two parties, especially enforceable by law, or a similar internal agreement wholly within an organization.
- NORM: A socially or organizationally defined expectation governing behavior within a DOMAIN or ACTIVITY. Defined in IEEE Norms specify conventions, standards, or best practices that guide interactions among ENTITIES. NORMS may be a standard, or principle of right action, binding upon the members of a group and serving to guide, control, or regulate proper and acceptable behavior. (Includes statutory law)

CREDENTIALS in the UDG consist of Verifiable Credentials issued by an issuing authority that can be presented and verified during an HSTP transaction. Verifiable credentials consist of a DID credential document stored wrapped in a HSML document — referenced by the document's SWID. Credentials are presentable and verifiable in compliance with the W3C Verifiable Credentials specification (W3C vc-data-model-1.1).

Types of credentials:

- Contract credential — when a contract is posted for execution, it should specify the credential required to sign and execute that contract.
- Peer-to-peer credential — In the case of automatic peer-to-peer discovery and connection, a proper credential is required.
- Master credential — in the special case of domain ownership that involved a real person, a verifiable credential is used a master credential that can override other credentials. This credential is used to encrypt a personal data store owned by the user.
- Exchange credential — controls the exchange of knowledge between nodes. In the case of peer-to-peer, the scope of the exchange is gated by this credential, where peer-to-peer copies.

8.3.3. Certification of agent credentials

CREDENTIALS are used by AGENTS to make claims about abilities to perform an AGENT function. The claim may be specific to a DOMAIN or to an ACTIVITY, i.e., ability to stably control the air temperature in a room using a residential HVAC system.

To obtain a CREDENTIAL the AGENT must be certified. Besides issuing the CREDENTIAL artifact, certification includes the process of giving official or legal approval to an AGENT that has reached a particular standard.

A framework for AGENT CREDENTIAL certification can be built based on previous systems for certification, e.g., medical devices, robots, automotive, railroad, avionics, etc. ([39])

8.3.4. Law as code

Law in HSML

8.3.5. Decisions based on norms and laws

The actions of an Agent in multi-agent social systems must include decisions that consider norms and laws.

In Multi-Agent Systems (MASs), multiple AGENTS operate in an environment and interact with some others. For an individual AGENT in a multi-agent system, their function is not only to interact with the environment but also to interact with other agents, perceive the states of other agents, and make decisions on how to respond to other agents' actions.

The Spatial Web provides mechanisms and methods that enable agents to understand and interact with other entities in the system as well as humans. This goal is centered around agents' decision making about when and how to interact with whom for pursuing specific goals or performing specific task. An important objective is to ensure that agents make right and good decisions — based on norms and laws — typically the best decision that they can do given what is known. Therefore, decision making is at the heart of building multi-agent systems.

Decision theory applies mathematical and statistical methodologies to help provide information on which decisions can be made. It is based on the axioms of probability and utility. To some degree, it is a combination of probability theory and utility theory. Decision Theory in the multi-agents must be accomodated by the UDG ([40])

Explainability refers to the ability of an AGENT to provide understandable reasons for its decisions and actions. One approach is to provide a framework for transparent introspection and decision-making by designing explainability with active inference ([41]).

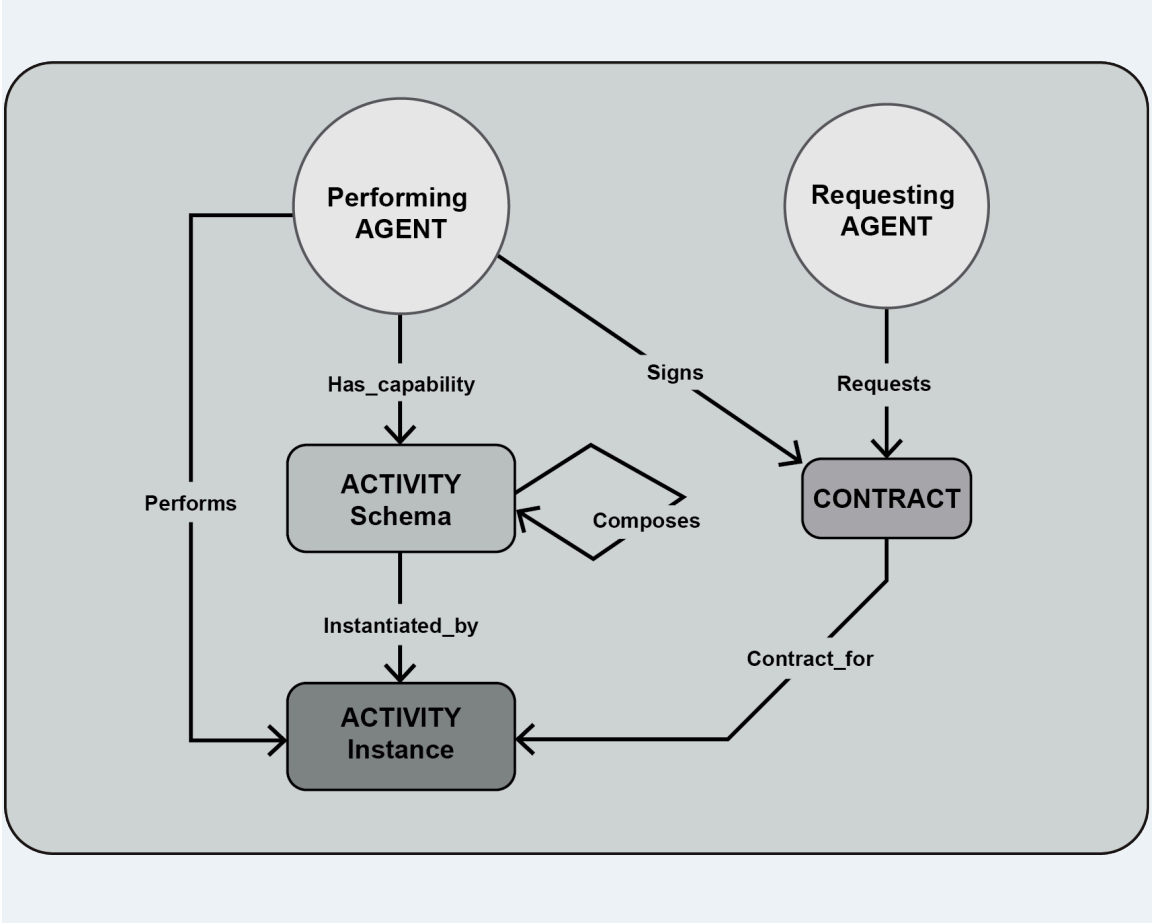
Decision Theory and decision explainability are key aspects of social agents exhibiting safe and trusted behavior. Safety and trustworthiness on the Spatial Web are enable by credentials, norms, and contracts.

8.3.6. Agents forming a contract

AGENTS will form contracts using the UDG. Contract must be consistent with laws, may be consistent with norms.

Relationships between entities AGENT, ACTIVITY, and CONTRACT relationships are shown in IEEE 2874 clause 6.6.11 including this Figure (Figure 32):

FIGURE 32: Agent-Contract-Activity relationship diagram



(Figure 32) shows the core relationships in the HSML ontology among, a) AGENTs, b) ACTIVITY Schemas, c) ACTIVITIES (aka ACTIVITY Instances), d) CONTRACTs. A

Requester is an AGENT requesting performance of a task or other CONTRACT. The loop connection on ACTIVITY Schema represents the composition of Complex ACTIVITY Schemas.

For additional information on AGENTS forming contracts for ACTIVITIES, see SWF Agent Framework Design Specification.

8.3.7. Normative agent functions

All social AGENTS will need the ability to recognize and acquire social norms. To achieve collective action in the UDG, social agents need functions that categorize observed behaviors as approved or disapproved, and a motivation to drive behavior in accord with the norms. Social norms emerge in multi-agent systems containing this agent function and the UDG must provide the conditions under which this results in socially beneficial outcomes. ([38])

Some AGENTS will need to perform specialized functions. These agents peform functions that maintain and enable the normative multiagent system. These functions include ([37]):

1. agents supporting communities in their task of recognizing, creating, and communicating norms to agents
2. agents to simplify normative systems, recognize when norms have become redundant, and to remove norms
3. agents to enforce norms.
4. agents to construct organizations
5. agents to create intermediate concepts and normative ontology, for example to decide about normative gaps
6. agents to decide about norm conflicts
7. agents to voluntarily give up some norm autonomy by allowing automated norm processing in agent acting and decision making
8. legal responsibility of the agents and their principals

8.3.8. Governance including privacy

Sharing of protected information between AGENTS

8.3.9. Requirements and Recommendations

TBD

8.4. Promoting collective intelligence

8.4.1. Collective intelligence as a public good

Collective intelligence is “a form of universally distributed intelligence, constantly enhanced, coordinated in real time, and resulting in the effective mobilization of skills.” ([44])

Ensembles of individual units, in general, have the capacity to perform better in a variety of ways than individuals on their own, in part because individuals can share information and other resources, they can coordinate or collaborate on activities, and/or they have the potential for differentiation of function, any or all of which can be leveraged in the production or maintenance of a public good. However, for these opportunities to be leveraged in the service of a public good, the associated trade-offs that come from competing demands, their costs, and the limitations of individuals, must be well managed. ([42])

Collective, cultural intelligence has evovled in humans with many cognitive skills not possessed by their nearest primate relatives. The cultural intelligence hypothesis argues that this is mainly due to a species-specific set of social- cognitive skills, emerging early in ontogeny, for participating and exchanging knowledge in cultural groups. Herrmann_2007 The Spatial Web extends these skills to artificial social agents as described in (8.3).

The Spatial Web plays the role of a collective intelligence for societal infrastructure. It hosts cognitive computing, i.e., inference (perception, learning, action selection). By combining and abstracting distributed information, the Spatial Web enables the formation of complex ideas and facilitates decision-making. The benefits of collective action can be numerous, from estimating hard-to-observe environmental cues, efficiently searching for resource, and mediating toward common understandings and shared norms.

8.4.2. Exploration and engagement

Each cohesive community has its own stream of idea flow that allows the members to incorporate innovations from other people within it, and even to create a separate culture. As a consequence of these shared habits, human communities can develop a sort of collective intelligence that is greater than the members’ individual intelligence ([26]). As described in ([exploration-engagement-network]), engagement with and learning from other AGENTS, along with the mutual sharing and vetting of ideas in the UDG, generate the collective intelligence.

The UDG provides the structure for enagement and exposure. Exposure is more important for driving idea flow than all the other factors combined highlights the overarching importance of automatic social learning ([26]). The UDG provides for automatic social learning. The UDG provides social network incentives to change idea flow that are distinct from using individual incentives.

UDG shall promote exploration outside the DOMAIN UDG shall promote engagement within the DOMAIN

8.4.3. UDG librarian AGENT

Maintenance and continous improvement of the UDG is essential. The UDG librarian AGENT provides the knowledge maintenance function. UDG Librarian AGENT functions are defined based on traditional librarian functions as extended with consideration of artificial intelligence:

- Automated Cataloguing and Classification
- Enhanced Search and Information Retrieval
- Virtual Assistants and Chabot's
- Data Analysis and Usage Insights
- Content Digitization and Preservation
- Accessibility Services
- Security and Fraud Detection
- Content Curation and Personalization

8.4.4. Constructive engagement AGENT

AGENTS that maximize the joy people get out of discussing things with others. Such algorithms alread exist: for emaple, Spotify and Apple Music's algorihtms do a pretty good ov of electing music that listenres will enjory. Avoid damage that social neworks are causing when trained for maximizing attention. Such agents would mizimize quality of engagement not time-on-device. ([45])

8.4.5. Nudging AGENTS

"Nudges" as exhibited by robotic, intelligent or autonomous systems are defined as overt or hidden suggestions or manipulations designed to influence the behavior or emotions of a user. This standard establishes a delineation of typical nudges (currently in use or that could be created). It contains concepts, functions and benefits necessary to establish and ensure ethically driven methodologies for the design of the robotic, intelligent and autonomous systems that incorporate them. (Nudging for Autonomous Systems)

8.4.6. Decentralization AGENTS

Design methodologies for decentralized groups provide rules that individual agents should follow to make their own choices on-the-fly that account for environmental cues and the reward, state, and/or action of other agents, in the service of a system-level goal. A successful approach to be used in the Spatial Web is called Decentralized setting with networked agents where there is no centralized controller and agents are connected via a communication network, so that the local information can spread across the network, by information exchange with only each agent's neighbors ([Zhang_2021])

The UDG communication and strategy must account for costs to individual AGENTS, such as in attention or energy, and trade-offs for the ensemble, such as the flexibility

to respond to an important change in the environment versus stability that is robust to unimportant variability. When there is a tension between the interests of the individual and those of the group, game-theoretic considerations may affect the level of collective intelligence that can be achieved. Models of individual rules that yield collective dynamics with multi-stable solutions provide a means to examine and shape collective intelligence in evolved and designed systems. ([42])

8.4.7. Collectively rational

The UDG Collective commons is desiged to be collectively rational. Whereas AGENTS may not be individually rational. AGENT behavior is determined as much by social context as by rational thinking or individual desires. Each AGENT has a level of intelligence that can be described as Bounded Rationality. Using the terminology of economics, in most things AGENTS are collectively rational, and only in some areas are they individually rational ([26]).

8.4.8. UDG is a commons

The UDG is a universal, digital commons. The commons are resources accessible to all members of a society. These resources are held in common even when owned privately or publicly. Commons can also be understood as resources that groups of people (communities, user groups) manage for individual and collective benefit.

A digital commons is "information and knowledge resources that are collectively created and owned or shared between or among a community and that tend to be non-exclusive, that is, be (generally freely) available to third parties Examples of digital commons are Wikipedia, free software and open-source hardware projects.

While the original work on the tragedy of the commons concept suggested that all commons were doomed to failure, they remain important in the modern world. Work by later economists has found many examples of successful commons, and Elinor Ostrom won the Nobel Prize for analysing situations where they operate successfully. Tragedy of the commons in the Wiki-Commons is avoided by community control by individual authors within the Wikipedia community.[26]

Cooperation in a large society of self-interested individuals is notoriously difficult to achieve when the externality of one individual's action is spread thin and wide on the whole society. This leads to the 'tragedyof the commons' in which rational action will ultimately make everyone worse-off.

Achieving global cooperation can be encouraged by localizing externalities to peers in a social network, thus leveraging the power of peer-pressure to regulate behavior. The mechanism relies on a joint model of externalities and peer-pressure. ([47])

8.4.9. polycentric governance

The Spatial Web employs polycentric governance which is a form of governance with multiple overlapping polycentric nodes of nested decision making, each of which operates with a degree of autonomy.

The Spatial Web is an information infrastructure for society. As such, societal issues will need to be addressed in the governance of the Spatial Web, e.g., privacy, location information, trust, and self-sovereign identity.

Polycentric governance of the UDG must address the concerns identified in ([STD-1]):

- Governance by legal authorities
- Governance for ethical development
- Governance for privacy.
- Governance of location information.
- Governance for identity management

While the UDG provides the computing and communication system consistent with these polycentric governance concerns, legal and orgnizational aspects are addressed in the Spatil Web Governance Recommended Practices (currently under development).

8.4.10. Requirements and Recommendations

TBD

- udg computing node shall promote exploration and engagement
- udg agent entities shall promote exploration and engagement
- UDG and agents promote peer-pressure to promote cooperation
- UDG and agents promote nudging consistent with IEEE P7008

9. UDG Distributed Computing

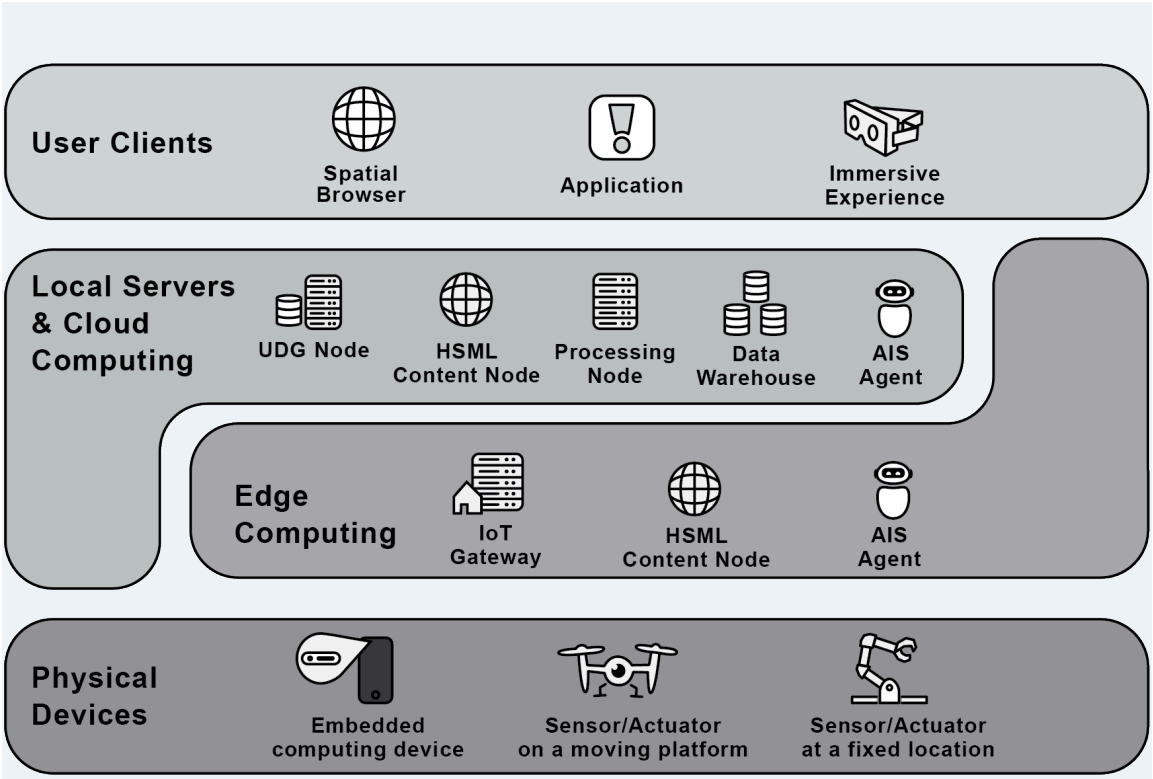
9.1. Computing continuum

9.1.1. Pantheon of SW Nodes

A Spatial Web Node (or SW Node) is a computing machine connected to the internet, capable of exchanging HSTP messages. SW Nodes include softrware components, e.g., daemons, that exchange Spatial Web entities as HSML. Each SW Node holds a portion of the UDG conceptual content. Messages between SW Nodes update the state of ENTITIES in real time.

A broad view of the distributed computing architecture of the Spatial Web node system is depicted in the (Spatial Web computing continuum).

FIGURE 33: Distributed computing continuum



9.1.2. UDG content in SW Nodes

SW Nodes host and exchange the content defined in the conceptual viewpoints of the is design specification. The table <sw-node-conceptual-content> provides a summary of UDG conceptual content held by the SW Nodes

TABLE 5: UDG conceptual content held in SW Nodes

SW Node	UDG KG	Hyperspace	Registries	Social
SW Content Node	Local ENTITIES	Embedding of local entities	DOMAIN Membership Credential	
UDG Node	Extended dynamic graph of ENTITIES	Hyperworlds and Atlases	Registry Relationships	Collective intelligence
Registry Node	Issues membership credential		DOMAIN Membership register	
Agent Node	Dynamic SITUATONS	Model uses Hyperworlds	DOMAIN Membership Credential	Social epistemology
Thing Node	Sensing and Actuation of environment	Location in cyber-physical space	DOMAIN Membership Credential	Observations of physical world

SW Node	UDG KG	Hyperspace	Registries	Social
DID issuer	SWID creation and resolution		SWID creation and resolution	

9.1.3. Dynamic content between SW Nodes

Content in the Spatial Web will change across a range of timeframes.

1. Agent speed: AGENTS interacting with moving THINGS with no HUMAN interaction. Latency: less than sub-millisecond range
2. Human reacting: HUMANS and AGENTS interacting with SW Content Nodes and moving THINGS. Bandwidth: 30 frames per second
3. Human thinking: HUMANS and AGENTS interacting with static SW Nodes. Latency: 250 msec
4. UDG synchronization: UDG nodes updating content across UDG node network. Latency: seconds

As shown in (Spatial Web continuum), the Spatial Web distributed computing environment spans across a continuum of communication networks, e.g., internet, mobile, LAN/WAN. This continuum features a wide variety of computing nodes and communication protocols. Deployments of the Spatial Web shall take advantage of the bandwidth and latency features in the different layers of the continuum.

The Spatial Web builds on internet mechanisms for time synchronization. Extensions are needed for managing internal local hstp.d clocks (repeating internal events) to manage query and updates against live resources.

9.1.4. Requirements and Recommendations

TBD

9.2. Content nodes

9.2.1. SW Content node

9.2.1.1. Spatial Web Node design

The primary, basic computing node in the Spatial Web is SW Content Node. The SW Content Node contains the primary software components for communications in the Spatial Web, e.g., hstp.d daemon for handling all hstp messages. Other SW Nodes extend the design SW Content Node.

A SW Content Node's central focus is to enable the discovery and sharing of information in a way that preserves privacy. Similar to a (SOLID Personal Online Data Store (PODS)), a HUMAN or AGENT user stores personal data hosted wherever the user desires. Applications that are authenticated are allowed to request data if the user has given the application permission. A HUMAN may distribute personal information among several pods; for example, different pods might contain personal profile data,

contact information, financial information, health, travel plans, or other information. The user could then join a DOMAIN community by giving it permission to access the appropriate information in a specific SW Node. The user retains complete ownership and control of data in the user's SW Conent Node.

Not all (perhaps not even most) DOMAINS will be in publicly available SW Nodes. Many DOMAINS will be private networks intended for access only by those with need to know (or to modify), especially those with IoT interconnections.

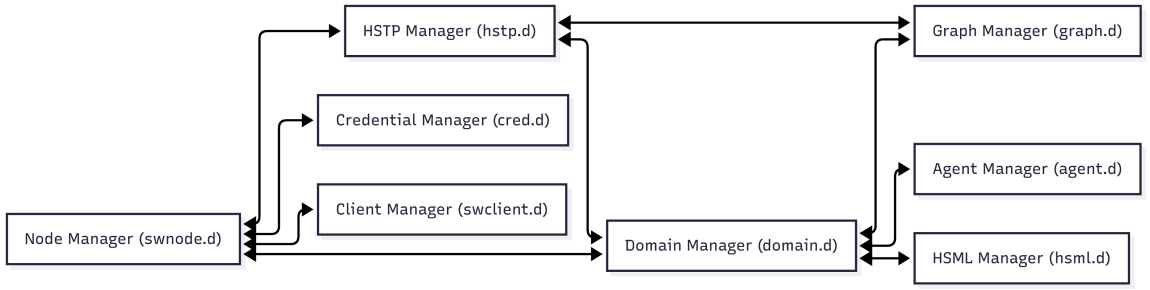
> *EDITORIAL NOTE*
>
> *the SW Content Node design shown here needs to be updated based on or replace by the design used for the Spatial Web Foundation SDK shown in https://gitbook.spatialwebfoundation.org/*

Structure of the SW Content Node, including the primary hstp.d and hsml.d daemons, the domain graph, a credential graph, and a schema graph.

A spatial web node is the abstraction of a long running application (the node daemon) that connects to other SW Nodes to provide information about specific HSML ENTITIES and communicats using HSTP. A SW Node can support both a server that maintains state of a local DOMAIN domain graph and provides stateful content (as HSML) and a client that can render that HSML in a form that a service or user can understand. The state of the local DOMAINS is maintained within a specially design *domain graph (currently built on top of a semantic RDF knowledge graph, though this likely will be hidden as an abstraction).*

The node is controlled by a specific daemon (or persistant process) called the Spatial Web Node Manager (indicated by the swnode.d process). The precise implementation of the node will vary, but at least to start with, it will usually run on a specific port (or more likely sets of ports) on a given machine.

FIGURE 34: SW Content Node Design



9.2.1.2. HSTP Manager

The HSTP Manager handles message routing from both external agents (users) and from other spatial web nodes, taking in HSTP based messages carrying HSML and payloads and transmitting HSTP responses back to users or (through the onboard

client) to other nodes. These messages are generally sent to the domain manager, rather than executed directly. (Note, this is different from the current implementation, though the primary change will likely just be which system handles these executions).

9.2.1.3. Domain Manager

The domain manager has a number of roles:

- generates a domain thread from a domain template and populates it with appropriate agents, things and places.
- works with the system clock (an internal tick) to invoke the activities on that thread of the relevant agents, which in turn is used for autonomous and semi-autonomous agents working in a state graph system,
- translates HSTP messages into UDG notifications for autonomous agents.
- manages inter- and intra-agent communication within domains
- determines whether the domain has reached critical states (such as an end state in a simulation)
- communications with the node domain graph to read and update state within the graph.
- passes relevant map responses (entity streams) back to the hstp manager.

9.2.1.4. Graph Manager

The graph manager is a low level service that interacts directly with the graphs within the node in order to provide an abstraction layer for graph management. It performs a number of functions.

- Translates HSQL query requests and updates into the implementation specific requests of the current graph technology. This exists primarily to ensure that there are no explicit dependencies upon the underlying graph store.
- Provides a mechanism to add multiple external graphs to the current graph so that they can be queried as if they were a single graph. This is what is known as a federated graph (and it is currently just specific to RDF, but that may change). Such a graph doesn't necessarily scale well towards a large number of nodes (>100), but it can be very useful when building a domain that scales across multiple machines. By separating the graph manager from the domain manager, it becomes possible for the domain manager on one machine to work with multiple nodes simultaneously without having to go across hstp.
- Graph replication. Replicating a graph (while something of an edge case) is easy enough to accomplish at the graph layer; RDF has global identifiers, and as such a graph can be replicated by simple serialisation into any RDF format. Record deduplication is similarly straightforward, as RDF is specifically built to work on an index format such that multiple resources with the same URI automatically to the relevant ntuple index. This is one of the many reasons that RDF is recommended from the graph layer.

9.2.1.5. HSML Manager

HSML is used to describe the state of domains within the graph, but it is also used to indicate activities, frames of activity over time, conditional expressions and contexts, both within the domains and within HSTP messages containing relevant changes and credentials. The HSML process is used in conjunction with the domain managers to provide indications of how entities change. It is not necessarily a daemon, but instead exists primarily as an interpreter that can then pass information off toe domain manager to implement, generally through the graph manager interface.

9.2.1.6. Agent Manager

Things within domains are agents. An agent can be thought of as something that is capable of change within a domain, with the most prominant such change being a change of motion within some phase space, or a change of state. The agent manager performs a number of roles.

- The domain manager typically manages the the "tick" of the system clock and its dissemination to the various agents. The agent manager is what interprets the messages of the domain to any given agent.
- Agent may be passive (they can only be activated by activities from other agents), active (they are capable of action independent of other agents), or inactive (they do not receive messages except for messages to activate in either passive or active mode). The inactive state exists primarily to reduce the number of cycles that a given agent requires for processing if not necessary.
- The agent manager handles moving an agent from place to place, either through linking or through replication across domains over spatial web node boundaries. If linking between such boundaries (typical, for instance, accross affiliated nodes that have similar domain constraints), then the agent manager will freeze an agent (make it inactive and hidden) on one machine, and will then replicate the agent on a different spatial web node, or updating the existing history of the agent to an existing proxy on a different node.
- Agents maintain an internal state history, the mechanisms for which are TBD, through the interface of the agent manager (there may be a history manager that specifically handles that operation, again TBD)

9.2.1.7. Credential Manager

The credential manager handles the creation of SWIDs on agents, places, and domains, as well as caching credentials from external hstp invocations, in effect acting as the wallet for the various domains within the node. This will typically be a proxy for various types of accreditation and verification mechanisms. Full implementation TBD.

9.2.1.8. Client Manager

This is a low level *command line interface* for text-based communication with a spatial web node. Every node supports some kind of CLI interface and may support others (multimodal chat, 2 or 2 1/2 D maps, 3D environments, animations, and so forth).

9.2.1.9. Dynamic handling of links

When a link is received by the domain manager, it uses the context determined by these parameters to determine other necessary metadata. These are then passed to the link's activation handler (or the defaults relevant by type) to perform the associated link action.

Links can be set up by the domain designer via the periodicity property as one of singleton (the link is only activated once) or periodic (the link is invoked across a given channel periodically until either the link is terminated or the channel's time-to-live (TTL) is exceeded). Once the link completes, it will either be reset (the default) or it will be expired (for links that expire upon use).

This operation is handled by the domain manager. Note that in fully autonomous operations, open links simply cause the agent to reset to the new place (and domain, if this changes, without UX involvement. However, key activation still requires the relevant credentials.

9.2.2. UDG Node and node network

9.2.2.1. Network of UDG nodes

9.2.2.1.1. Distributed graph database technology

A UDG Node is a a type of SW Node that performs UDG functions. UDG Nodes work as a network in as a Distributed replicate synchronization.

Design requirements are identified by thinking of the UDG Node Network as a Distributed Graph Database. A distributed graph database stores data as vertices and edges, distributing this information across multiple machines. This architecture offers significant performance advantages over traditional, centralized databases, especially when handling large, interconnected datasets. Instead of storing the entire UDG on a single server, a distributed graph database spreads it across numerous machines. This enables parallel processing, where multiple machines work concurrently on different parts of the data. AS the UDG grows, scalability is handled by add more machines to the network. Distribution also makes the system more fault tolerant. (The discussion here is from ([51]), but it is typical of other descriptions.)

Components that graph databases employ to handle the complexity and scale of graph data distributed across several machines include these:

- Sharding. Sharding breaks down a large graph into smaller, more manageable pieces. Each shard represents a subset of vertices and edges.
- Distributed query processing. When receiveing a query, the database first identifies which shards contain the relevant data. It then breaks the query into subqueries, each targeting a specific shard. Then the machines holding the shards process these subqueries in parallel.
- Distributed concurrency control. Distributed graph databases use concurrency control mechanisms to prevent conflicts and make sure that the database remains consistent even during concurrent operations.
- Replication. Replication guarantees the availability and fault tolerance of distributed graph databases. Each shard is replicated across several machines, meaning that

if one machine fails, the system can continue to function by redirecting queries to another replica.

Scalability of the UDG Node Network is needed in order to meet the estimated size of the Spatial Web (<udg-size-estimate>).

9.2.2.1.2. UDG node network tiers

To organize the UDG Node Network will require multiple tiers:

- Extended Domain Graphs — this extends the scope of a given domain by incorporating external graphs into the systems at the query level. This makes using common codebases and templates feasible
- Spatial Web Nodes — Spatial web nodes are the physical backbone of the spatial web, and are primarily the servers that host the various managers of resources.
- Affiliation Networks — Each node (and many domains within the nodes) belong to one or more affiliation networks. Some of these may be huge, with potentially millions of nodes, others may be the equivalent of local intranets. Moreover, affiliations can themselves be affiliated, creating a superstructure that can scale up to:
- The Spatial Web — This is the aggregate of all affiliation networks.

The affiliate design is a specific requirement for a decentralized architecture. A true peer-to-peer system likely will not scale to the same level (there are few Internet scale peer-to-peer systems after more than 35 years). This would especially be the case given the requirements to ensure private control over domains, along with the sensitivity of much of the internal data.

9.2.2.2. UDG Node modules

The UDG Node consists of a number of software modules. The UDG Node includes all components in the SW Content Node design. Additional components are needed in the UDG Node to accomplish the UDG functions as a distributed graph databased.

- The udg.d domain daemon: This is an internal high frequency loop that is used in order to animate domains to manage state. It is only very peripherally connected to HSTP (primarily when dealing with SW Node to Node communication), but in general it this process that handles the internal state changes to the domain graph.
- augmented graph module: This module works in conjunction with the hsml.d process to join multiple domain graphs together into a single comprehensive graph. Note that this is used primarily to extend the domain graph for handling complex environments, and generally sits outside of the normal hstp.d processes.
- Transformation Pipeline: Ordinarily, the output from a query will be some form of graph. However, for a number of reasons, there will be times when the output needs to be transformed into some other format (most especially HSML) or filtered in some other way (such as through an LLM). The transformation pipeline handles this process.
- Collector: Certain operations involve aggregating the results of queries across multiple spatial web nodes. The Collector module (part of hstp.d) is a queue that

collects incoming messages and aggregates them for transmission back to the client. The exact details of the collector module are still TBD.

- Extension Modules: In all of these components, the fundamental design will be modular, such that each component can be extended by code depending upon implementation. For instance, the HSML encoder, the UDG.d and the HSTP.d all have access points for agentic systems and e-commerce capabilities, components can be used for converting external datasets into data analytics forms, transformations can be written that generate images, maps, and related formats and so forth. These module extensions would be integrated in by individual SW Node Administrators.
- Hyperworld module: This module provides services on (Hyperworlds). The module provides a persistent store of (HRSs) and the ENTITIES embedded in the Worlds held in the node. The module peforms queries including spatial queries on Hyperworlds.

9.2.3. Registry Node and network

The (hierarchy of DOMAIN Registries) is implemented as a network of Registry Nodes.

A Registry Node performs the computing and communication functions necessary to deploy the content and functions defined in (Clause 7)

A Registry Node is a specific SW Node that provides a number of services related to discovery, definition, search, DiD and alias resolution of resources.

The Registry Node consists of a number of software modules. The Registry Node includes all components in the SW Content Node design. Additional components are needed in the Registry Node to accomplish the Registry functions.

- Membership Register: Module for persistent storage and management of the DOMAINS that are members of an upper DOMAIN.
- Credentials generator: Module to generate Verifiable Credentials that support the claim that a DOMAIN is a member of an upper DOMAIN.
- SWID Generator: Module to generate SWIDs using the did:swid method.
- Registry management: Module to satisfy auditability requirements of the registry.

EDITORIAL NOTE

this design linking SW Registry to IP concepts needs to be reviewed.

When a Spatial Web ENTITY is registered with a SW Registry several things happen:

- The ipv6 address of the node server is registered, along with a web domain name and (if different from the default) a port. The SW registry can also register the relevant IP addresses.
- A SW domain on a SW node can be assigned a public SW credential that indicates that the domain in question is a part of the SW network (similar networks can be established with different sets of credentials).

- Periodically, the spatial web node can send an update of all domains on that node that have the relevant credentials. This include any metadata (topics) that are associated with the domain. Note that these domains provide access points to other domains that may not necessarily be transmitted to the registry. As such they should be seen as starting points for various domain activities. Not all domains on a node need (or should) be so registered.
- Registries that issue their own credentials create *affiliation networks*. For instance, a given company that produces lines of IoT devices with associated HSML interfaces may end up providing an affiliation network of all nodes that make use of these devices, and as such share common domain and agent interfaces, taxonomies, structures and so forth. Similarly, a multi-system role playing game may set up an affiliation network where each node hosts one or more domains in that particular universe, with the ability for agents to move from one node to another through the use of supported credentials in that affiliation network.
- A SW Node (and associated domains) can be part of multiple affiliation networks. For instance, a federal government may provide a core affiliation network for its member states, each both sharing resources and providing information, as well as identifying what other nodes are part of that affiliation.
- Both a repository and a registry are spatial web nodes. What differentiates them is primarily whether they have the additional functions of registration and whether they permit sharing within one or more affiliate networks. This are additional modules that can be added on to the base functionality of the spatial web node.
- Moreover, a spatial web node can be both a repository and a registry.

9.2.3.1. Affiliation Networks

EDITORIAL NOTE

the need and design of affiliation networks needs to be reviewed

An *affiliation network* is a network of spatial web nodes which shares common resources, taxonomic classifications and typically a common registry. The registry serves as the hub of the network, identifying membership in the affiliation network as well as providing a mechanism for discovery within that network.

One of the roles of a registry is to issue and affiliation credential. This credential serves as a way of verifying that nodes within the network are in fact part of that network, and provide permissions that spatial web clients need to have in order to access certain features.

For instance, a group of universities in a given region may establish an affiliated network. This means that each university effectively agrees to abide by specific taxonomies as a way of organizing information, provides common set of activities for performing such tasks as transferring students between universities, enrolling in classes, and so forth, and will often allow students and faculty from one university to access resources or get consistent grading at other universities within the affiliation.

This is accomplished through a “university league” credential which is issued when the node is added to the network. When a student registers to a given node, their

user agent (the software client they interact with) within the system receives a corresponding private key credential that both makes the user a resource in the system and provides them access to that system.

This serves a number of functions. For instance, an administrator can perform an affiliation level search for a given student, faculty member, class, or program (among many other things), either by ID or by attributes. A student can register with another university within the affiliation to take a class remotely, or can even sign up to and use remotely controlled laboratories stations (such as observatory time at a telescope or participation within a collaborative concert). A teacher can make available resources such as books or training videos from protected repositories to all of her students.

In this particular case, the registry serves to identify those domains within the network of nodes of affiliated members that may contain the desired resources. When a query is made in the broader context of the affiliation, each of these affiliated nodes are then queried in turn and return the associated links to those resources as a structure (analogous to an RSS or Atom type structure) that are then collated by the calling domain.

Note that the nodes in these affiliated networks are not (typically) graph extensions. A graph extension expands the active domain graph of a given node and is normally secured, because it exposes all resources within that graph. An affiliation query, on the other hand, is a request for information (typically links but also maps) from other nodes in the affiliated network.

9.2.4. Requirements and Recommendations

TBD

9.3. Human and agent nodes

9.3.1. Agent nodes

AGENT ENTITIES are hosted by Agent Nodes. The Agent Node persists the model and goals used by the AGENT ENTITY. Agent Nodes will interact with the UDG in multiple ways.

Agent Nodes manage AGENT SITUATIONS. At any given time an AGENT may need to know what DOMAINS are in its environment. The SITUATON ENTITY is the set of DOMAINS currently in the context of the AGENT. An AGENT will establish part of its SITUATION by querying the UDG. If the AGENT is a cyber-physical ENTITY it will also use its local sensors to inform its SITUATION. A SITUATION is part of the Model held by the Agent Node.

Agent Nodes interact with UDG nodes for information about Hyperworlds. Hyperworlds are defined in vector space class of HYPERSPACE and may include any kind of DOMAIN, e.g., geographic, conceptual or a combination. Hyperworlds are persisted and queried in UDG Nodes. An AGENT will discover local DOMAINS by querying the (Hyperworlds) in which it is currently located.

Agent Nodes interact with the Domain Registry Network. An AGENT may be a member of one or more DOMAINS. The Agent Node will interact with a Registry Node in order to register the AGENT as a member of an upper DOMAIN. The Agent Node will contain a Wallet for holding the DOMAIN Membership Credentials. An Agent Node may query a Registry Node to determine the list of DOMAINS that are members of a DOMAIN.

Agent Nodes provide the means for social interactions of AGENTS with other ENTITIES. An Agent Node will iteract with other SW Nodes to provide the (idea flow) in the social network of the UDG. An Agent Node will provide the communication management associated with the AGENTS exploration and enagement with other social AGENTS. The Agent Node will maintain communications with the various levels of (social groups) in which the social AGENT engages. Note that the social networks are different from the SITUATION and DOMAINS based on Hyperworld location.

An Agent Node consists of a number of software modules. The Agent Node includes all components in the SW Content Node design. Additional components are needed in the Agent Node to accomplish the AGENT functions.

9.3.2. Library Agent Node

The (UDG librarian AGENT) provides the knowledge maintenance function essential to the Maintenance and continous improvement of the UDG. UDG Librarian AGENT functions are implemented in a Library Agent Node or Librarian Node.

A Librarian Node consists of a number of software modules. The Librarian Node includes all components in the SW Content Node design. Additional components are needed in the Librarian Node to accomplish the UDG Librarian AGENT functions.

9.3.3. Human Avatars

9.3.3.1. Humans as AGENTS

The Spatial Web ontology defines humans as a special type of AGENT ENTITY.

Humans participate in the Spatial Web through Agent Nodes with additional conditions.

Presentation of the UDG contents to a human is done by a user interface of an Agent Node.

> *EDITORIAL NOTE*
>
> *It is not certain that this material about Human user representations belongs in this UDG Specification. It may be moved to the Agent Framework Design Specification*

9.3.3.2. Human user representations

A *map* has a very general meaning in HSML: it is a *representation of a domain. A domain is a restful entity — it has an internal representation within the associated graph- but for a number of reasons the graph that gets produced when a request is made about a domain or other entity will likely not be identical to the internal graph, but will rather be a computed graph (as JSON-LD or similar structure).*

There are three alternative approaches that can be taken with regard to representation.

9.3.3.2.1. Thin Client (Declarative — Wave 1)

This was the original approach taken with the web from about 1993 to the early 2000s. In this case, HTML was the declarative language that defined the structure of a web document, and while there was a limited amount of interaction via scripting (Javascript arrived in December 1995), for the most part the client experience tended to vary from browser to browser. Significantly, it should also be noted that most documents were comparatively small and self contained, meaning that all of the state of that document could be transmitted as a single message.

9.3.3.2.2. Thick Client (Imperative — Wave 2)

This approach had its heyday from about 2005 to around 2022, and primarily involved the increasing use of Javascript to build applications of increasing complexity, while at the same time, building on increasing standardization on both core functionality (e.g., ECMAScript) and the increased modularization of defineable web components. In general, this approach works best for a clearly defined client — the browser page in effect acting as a platform for development of specialized clients.

9.3.3.2.3. Thin Client Streaming (Declarative — Wave 3?)

This is the rise of the chat interface — in essence a continuous stream of information that emerges as part of the prompt/response pattern inherent with LLMs and GenAI systems. This approach shifted the interface from being primarily static and fixed to one that reflected a continuous update, and has since shifted into a multimodal design pattern involving just-in-time editors being launched to create or edit objects that are generated as artifacts.

Ideally, the Spatial Web should support all three of these modalities. There are two aspects that are important in all of these, however. First is the fact that the Spatial Web is temporal in nature as well as spatial — a typical application will not make one query against a particular system, but more than likely will continuously ask for changes in the state of that system. In essence, a connection is stateful, reporting the state of a system repeatedly until it is told not to. At the same time, there is a compelling use case for providing a historical rendition of a given domain or agent as it changes over time.

This implies that the third case — streaming — will like be a major use case, which is actually one of the key advantages of using RDF. A typical interchange in this scenario may be as follows:

In this particular case, the user requests an hsml:Map object (not currently defined) which can be represented as a JSON instance or Turtle stream that gives the current state of the domain, then periodically a delta indicated a state change. For the apartment as an example, the map would indicate the full state of the map domain as

given above (assuming the user has the correct permissions). Thereafter, the domain will send messages along the lines of:

FIGURE 35

```
<<__:JaneDoe hsml:hasLocation (7,3) >> hsml:message [
hsml:status hsml:DeleteNode;
hsml:time "2025-08-04T09:10:11"^^xsd:dateTime ;
].
<<__:JaneDoe hsml:hasLocation (8,4) >> hsml:message [
hsml:status hsml:AddNode;
hsml:time "2025-08-04T09:10:11"^^xsd:dateTime ;
].
```

where `:JaneDoe` resolves to the internal identifier for the agent, and <<...>> indicates a reifier for the statement.

To summarize, the client requests that the Spatial Web Node opens a connection to a domain, handing back the channel key for that domain to the client for direct communication. Once the key is open, the client requests a map (i.e, descriptive representations) that gives the relevant queried context for the domain in question as a structure, then, as the environment changes, provides updates to the map indicating when resources have changed.

When the client sends a message to stop, the server will stop the update and generate a new whole map that reflects the state of the domain at the end of the run.

Note that this process can be interspersed with commands to the representative agent within the domain. The commands coming from the client do not directly change the state of the domain. Rather they indicate to the domain that the agents should be directed to change their configuration to either achieve the goal or determine that they can't achieve the goal. An internal loop then manages the updates to the graph to set the relevant changes in state within the various agents in the domain.

Note that this approach can also work well when you have multiple agents that are interacting in the same domain, driven by different external clients (or internal autonomous agents).

9.3.3.2.4. Icons

The Spatial Web by itself is not meant as a vehicle for transmitting imagery or 3D models, but because what it does generate are descriptions of physical systems, it is frequently desireable to have some way of indicating to a spatial web client how it should represent the entity in a map or projection. This is the role of icons, as represented by an `hsml:Icon` entity.

An *icon* is an entity with a reference to either an internal or external media source, likely in the form:

The `hsml:href` is a pointer to the media resource in question, while `hsml:mediaType` indicates which media type it is used. This may be inferred based upon the extension

in the href resource if this is known (as in the second example). The media type is used primarily to indicate to the user client how the resource should be displayed.

For instance, in the third example, you have an agent representing the Eiffel Tower in Paris, France. If the user client is a 2D browser, then this may be represented as a transparent PNG file on top of a map. On the other hand, if the client is a 3D browser, this may be represented using the EiffelTower.obj 3D model.

Icons can maintain positional and orientation information appropriate to the entity. The goal with such icons is not necessarily to provide a precise representation or rendering, but rather to provide to the user agent a way of constructing an approximate representation to indicate symbolic relationships.

Note that a given entity may include both an icon and a link. The link is an abstraction on the entity, not the icon.

9.3.4. Requirements and Recommendations

TBD

9.4. Thing and twin Nodes

9.4.1. Thing nodes

sensor and actuators without AGENT

FIGURE 36

```
*Thing Node*
```

THING ENTITIES are hosted by Thing Nodes. Thing Nodes include the physical devices layer of the (<<[[computing_continuum, computing continuum>>). A THING DOMAIN is a bounded entity without agency. A Thing Node provides a Spatial Web presence for physical devices. Thing Nodes will interact with the UDG in multiple ways.

Thing Nodes provide Observations to UDG. Observations are the result of a sensing event and the use of processing to convert the physical energy detected by the sensor into a digital message. The observation message is communicated using HSML. Multiple SW Nodes may access a Thing Node to observe the environment.

Thing Nodes affect the environment on behalf of other SW Nodes.

Thing Nodes interact with UDG nodes to provide information about (Hyperworlds). THING DOMAINS may be located in Hyperworlds. When an AGENT seeks information about DOMAINS in a Hyperworld, it may include requests to Thing Nodes. This is a bridge between cyber and physical worlds.

THING DOMAINS may be registered in a DOMAIN registry. In that case, the Thing Node will hold the Membership Credential that supports the claim of the THING being part of the upper DOMAIN.

An Agent Node consists of a number of software modules. The Agent Node includes all components in the SW Content Node design. Additional components are needed in the Agent Node to accomplish the AGENT functions.

9.4.2. Digital Twins for things

A Digital Twin is a virtual representation of entities and processes, synchronized at a specified frequency and fidelity.

Digital Twins, serve as a bridge between the physical and digital Domains, allowing agents to test hypotheses, optimize strategies, and learn from virtual scenarios without direct physical intervention. These interactions not only extend the operational capabilities of AGENTs but also foster a deeper integration of the physical and digital environments, shaping the outcomes and behaviors within both realms.

The Digital Twin of a THING may be part of the Thing Node or it may be a seperate Node.

9.4.3. Requirements and Recommendations

TBD

9.5. HSTP addressing, credentials

9.5.1. HSTP operations

HSTP is an application-level protocol for distributed and interoperable computer systems and end-point devices. HSTP enables HSML-compliant systems to communicate to one another to execute functionality and share data.

An HSTP operation as defined in ([std-3]), includes information from the UDG in order to determine the target for an hstp operation, including:

- Target. A target identifies the intended recipient or target system for the operation specified in the message. The format can be a spatial Domain with modifiers, a SWID with modifiers, a specific URI endpoint with modifiers, a protocol version, or the requester's own SWID. The primary formats expected are SWIDs or URIs.
- Neighbors An optional list of the sending system's (requester's in a request, target's in a response) publicly-listed neighboring SWIDs within the Universal Domain Graph (UDG).

EDITORIAL NOTE

STD-3 indicates that this information is under development and will be detailed in a future draft

9.5.2. Networks Neighborhoods of nodes

EDITORIAL NOTE

It is not clear how this clause relates to other items in the Spatial Web design. After editing, this clause may be useful in relation to HSTP operations or it may be deleted.

A SW Node communicates with other SW Nodes via HSTP. SW Nodes connect in one of two ways:

- Via *neighborhood connections* made when the node connects via a nodelink (a specialized form of link that identifies a node) to another node. The successful negotiation caches the nodelink in the node's *nodecache*. This is a peer-to-peer connection.
- Via registration with a *public node registry*. In this case, the node becomes visible to all other members of the registry. The registration process returns a set of credentials that translate selected links from a search into neighborhood links.

In both cases, when a node queries another node about its neighborhood, it can cache the connections of the queried node. Case 2 in fact is essentially Case 1 at a larger scale. Each node contains a certain amount of intrinsic metadata about the type of domains on the system, which can in turn be used to provide a facet query to find links from another node that have similar or overlapping facets.

For instance, if a given node mostly contained shipping information, the facets that it has can provide scores against another SW Node's registry metadata that was tied into shipping to retrieve those particular SW Nodes that are most like the requested nodes.

In addition to that, when a Spatial Web Node first registers with a registry, it can retrieve schema definitions, common resources, code libraries and a "starter" set of node links that can then be used to add to the neighborhood of the registering node.

Nodelinks typically have a time-to-live attribute (TTL) that indicates how long a link can remain active before it needs to be refreshed. If a nodelink cannot be resolved, then a secondary TTL is activated that indicates how long an interval should be taken before the link is considered dead, and consequently should be purged.

When a nodelink is activated, it follows a specific discovery process:

- Is there a corrresponding certificate in the local nodelink cache. If so, use this, if it's not expired.
- Otherwise, check the immediate peers (the immediate neighborhood) to see if a nodelink is resolvable. This is important because not all nodelinks are public.
- If the link cannot be resolved in the immediate neighborhood, go to the public node registries.
- If no node is found at that point of resolution, return an HSTP No Address error (the analog to an HTTP 404 code).

Once a connection is made, the corresponding certificate is cached so that this process of discovery doesn't need to take place again until the next TTL.

9.5.3. Augmented Node Graphs

> *EDITORIAL NOTE*
>
> *It is not clear how this clause relates to other items in the Spatial Web design. After editing, this clause may be useful in relation to HSTP operations or it may be deleted.*

In some cases, it is possible to attach the graph of one node to another in order to create a larger federated graph. This bypasses the normal HSTP protocol system. This is used primarily for those cases where a single domain may in fact be distributed across multiple servers, or where common resources such as schemas are shared across multiple chained nodes. The chained nodes are called *Augmenting Node Graphs* and the chaining graph is called the *Augmented Node Graph*. More information will be forthcoming regarding such graphs.

9.5.4. Directory Domain and Home Domains

> *EDITORIAL NOTE*
>
> *It is not clear how this clause relates to other items in the Spatial Web design. After editing, this clause may be useful in relation to HSTP operations or it may be deleted.*

If no DOMAIN is specified by an hstp request, this will be the default domain. This domain is designed to provide a directory or catalog of the domains that are accessible to a given external agent based upon their credential profile, and also provides mechanisms to "sign in" if this is required to change the domains that they see.

Similarly, within every domain, there is the option of specifying a home place. This is where agents are positioned when they first "enter" a given domain, if no domain is otherwise specified. In simple scenarios (such as the smart room scenario), there may be only one place in the domain, but in more complex scenarios (especially those representing tours or rpgs), this home place typically also serves the role of establishing context and backstory for the agent, providing instructions for interacting with the domain, and identifying pertinent "destinations".

There is assumed to be on a given spatial web node a designated Home Domain that is explicitly stated to be associated with the node itself. It's "agency" in this particular case is the action of the node daemons, with specific capabilities. When a spatial web node is first set up, this home domain/agent holds the configuration metadata for the node itself, as well as any credentials that are specific to the node.

Put another way, *from the standpoint of the UDG, the Spatial Web Node is a domain, with an implicit super agent. The mechanics of this are still to be determined.*

9.5.5. Spatial Web Addresses

EDITORIAL NOTE

It is not clear how this clause relates to other items in the Spatial Web design. After editing, this clause may be useful in relation to HSTP operations or it may be deleted.

In the Spatial Web, there is a distinction between a Spatial Web Identifier (SWID) and a *Spatial Web URL (here, proposed as SWURL). The SWID provides an address to a credential that verifies the existence of that resource, but does not in fact identify where a resource is within the spatial web. This makes it far more difficult to create a linking system as such credentials are not necessarily guaranteed to be within the same indexing system.*

_Addressing and *_credentialing* serve two different functions. A *spatial web resource locator* (or SWURL) identifies where a given resource is located on the spatial web. The address typically will identify a spatial node (the physical system where the resource is located) coupled with an identifier for that resource on that machine.

A SWAD does not make any guarantees by itself about the verifiability of the address (this is the role of the SWID), nor does it identify the resource semantically. Instead, the SWURL is a label that locates the resource on the web itself.

Just as a resource has a SWID, it also has a SWURL. The SWURL is a *local name that is assigned to the resource in question, utilizing HTTP naming conventions. A resource may have more than one SWAD, or none. If a resource has no SWURL, then the SWURL defaults to the portion of the SWID after the "did:swid:" method. If a resource has multiple SWRLs, then any of these can be used to reference that resource.*

The UGD.d resolves local SWRLs and returns the resource in question, but only after it verifies credential access for that resource via its SWID, returning an Unverified Access Error if the resource fails its credential check.

For instance, if the spatial web node has a SWURL of:

FIGURE 37

```
https://mySmartRoom.com:8200
```

with 8200 indicating the port number where the hstp.d daemon is located (there is no port specifically dedicated to the spatial web, but it would be a good idea to be thinking

about this), then resources that are defined on that node (such as domains, agents, scripts, etc.) can be further accessed by normal http qualification methods, such as:

FIGURE 38

```
https://mySmartRoom.com:8200#agent-light-123
```

If done with a content type of `application/hsml+json`, this would retrieve an HSML description giving the relevant details of the resource in JSON-LD (not necessarily the internal one-to-one encodings — the internal graph exists not for commonality but for state management). If the content type is `text/html` then what gets returned is a summary of that resource or system in an HTML format, and so forth.

This same entity is represented as a graph, quite possibly one given as a blank node:

FIGURE 39

```
# Turtle
prefix hsml: <http://spatialwebfoundation.org/hsml#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
prefix swid: <did:swid:>
@base: <https://mySmartRoom.com>

[] a hsml:Agent;
    hsml:swid swid:3195A951EF1109 ;
    hsml:swrl <#agent/light-123> ;
    rdfs:label "Light 123" ;
    .
```

The notation <agent/light-123> for the swrl is indicative that (at least in RDF) this is an IRI fragment relative to the containing spatial web node.

A *blank node* is a node that has an IRI that is defined within a graph, but is not defined globally. This structure makes it possible within Turtle to write something like:

FIGURE 40

```
# Turtle
prefix hsml: <http://spatialwebfoundation.org/hsml#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
prefix swid: <did:swid:>
@base: <https://mySmartRoom.com>

[] a hsml:Domain ;
    hsml:swid swid:EA519DEFFC1235 ;
    hsml:swrl <#domain/lightRoomScenario> ;
    hsml:hasAgent [
```

```
        a hsml:Agent;
        hsml:swid swid:3195A951EF1109 ;
        hsml:swrl <#agent/light-123> ;
        rdfs:label "Light 123" ;
        ]  .
```

The domain and agent SWRLs in this scenario then resolve to:

FIGURE 41

```
# Domain SWURL
<https://mySmartRoom.com#domain/lightRoomScenario>
# Agent SWURL
<https://mySmartRoom.com#agent/light-123>
```

Every spatial web node has a distinct base, and for the most part, resources are defined relative to those nodes. This is a bit of a departure from the normal best practices for the semantic web, but the distinction here is that most spatial web resources are effectively local to their nodes. Because a given resource can have multiple SWRLs, this also implies that most references will be indirect — "give me the (graph) node that has this SWURL", just as one would say "give me the (graph) node that has this SWID".

One other key point — the spatial web does not recognize URL parameters being passed as part of a GET request — if you need to pass parameters, these should be passed as the body of a POST request. This keeps the address space clean, makes it easier to validate incoming requests, and is more consistent with regards to semantic web principles.

9.5.6. Node access security and credentials

EDITORIAL NOTE

It is not clear how this clause relates to other items in the Spatial Web design. After editing, this clause may be useful in relation to HSTP operations or it may be deleted.

A central part of the Spatial Web is the use of secure credentials in order to maintain *surety within the web, where surety can be defined as the verification that an assertion being made about a particular entity was valid.*

Surety is made possible through the use of credentials that can be issued both by spatial web nodes that identify that specific resources have been created by that node, as well as assertions made by external authorities that a given agent has the relevant credentials to perform specific activities pursuant to a contract.

The mechanism that binds these credentials is the *Spatial Web Identifier* (or *SWID*), which is a specific key that references a credential *within the Spatial Web Node. This key*

is a decentralized identifier (or DID) according to the (W3C DID Core Specification). All DiDs issued by a spatial web node are further considered to have a SWID method that indicates that such credentials follow the Spatial Web standard (D3.3.1). The specific format for such credentials is still being worked out.

9.5.6.1. Credential Stores and Addresses

The *credential.d* daemon is responsible for both the issuance of SWIDs as well as the resolution of SWIDs. It is *recommended* that each Spatial Web node maintains a specific cache of credentials that are issued by it as part of the domain graph architecture, with the SWIDs then being treated as identifiers by the system to those credentials.

A credential in this particular case can serve primarily as a passthru reference to an external DiD that has a specific issuer that can be resolved within the internal SW Credential structure, and which utilizes a separate addressing mechanism (such as https) to identify the location of the issuing server *if that server is not the current spatial web node*.

A SWID is *not a Spatial Web URL (SWURL). The SWID serves to either identify the credential within the current Spatial Web Node or, through reference, to point to the location of an issuing server, while the SWURL provides an address (a Uniform Resource Locator or URL) to a resource within the broader spatial web network, which in turn may have a SWID to its relevant credential.*

The D3.3.1 specification indicates that all entities must have SWIDs. This perforce indicates that all entities must have credentials. It should be noted that not all credentials issued by the spatial web nodes *must* be cryptographically secure, though this may be a requirement imposed within a future specification.

9.5.6.2. Credential Issuance

A Spatial Web node is able to issue credentials to all entities that it creates. When that entity, such as a domain or agent, is created within the domain graph for the node, the SW Node will issue a cryptographically bound SWID that is associated with that entity and that consequently provides surety for the existence of that entity throughout the entity's life span.

Moreover, when an entity undergoes a material change, such as an agent moving from one domain to another which necessitates the creation of an additional proxy between those domains, then a new credential is issued indicating the change of "ownership" of that entity, along with a pointer indicating the previous owner (in effect forming a transitive chain). Such SWID transfers act, in effect, as a chain of custody for the resource.

One key point — an entity is always bound to its spatial web node. The flipside to this is that *each spatial web node issues its own SWIDs. Put another way, there is no centralized authority for the issuance of SWIDs on resources. Instead, to find a given entity, you use the SWURL for that entity to locate it in the Spatial Web, then you validate that the entity is as stated based upon its credential on the indicated node.*

Additionally, additional credentials can be bound to the same SWID, a key point in making contracts work. These are typically tied into activities and norms and often require multiple different SWID holders to create a contract with its own SWID that

binds the activities of agents together as specified by the boundaries of the contract itself. This work is still under development.

9.5.6.3. Credential Revocation and Registries

Just as the Spatial Web Node is the issuer of a credential, so too can it revoke a particular credential to indicate that the credential is no longer valid. Note that Spatial Web Nodes can also issue credentials indicating membership by other spatial web nodes within an affiliated network for which it acts as a registry.

This in turn means that revocation of a given spatial web node from a given affiliation network is never accomplished by that node, but rather by the affiliation holder, unless the registry node is also part of the affiliation network (ie, is self registering).

> *EDITORIAL NOTE*
>
> *It may be that a given registry is explicitly not a part of its own affiliation network. This is still to be determined, as it has implications on what a registry node can support.*

Because a spatial web node has its own implicit home domain, a node can be removed from a network by revoking the credentials of the home domain for that machine. The machine is still findable via a URL, but the lack of credentials mean that the request for data can't validate (it will send back an error across hstp indicating the data won't validate).

9.5.7. Requirements and Recommendations

TBD

9.6. Queries

9.6.1. Domain Graph Queries

There are two distinct methods that can be used for querying the state of a domain within a Spatial Web node: HSTP Node Queries and UDG Graph Queryies. Both of these work on the Domain Graphs for a given node, but do so in very different ways.

9.6.1.1. Domain Graphs

The *Domain Graph* for a given node consists of a semantic graph that represents all of the domains along with the relevant definition files (schema files) and relevant scripts (activities and policies). This domain graph is represented using RDF.

A DOMAIN is controlled by one and only one domain manager on a Spatial Web node, and a domain does not extend beyond that node. This makes it possible to effectively manage containment and synchronicity, as well as to ensure that the same active agent does not appear on multiple services node servers at the same time.

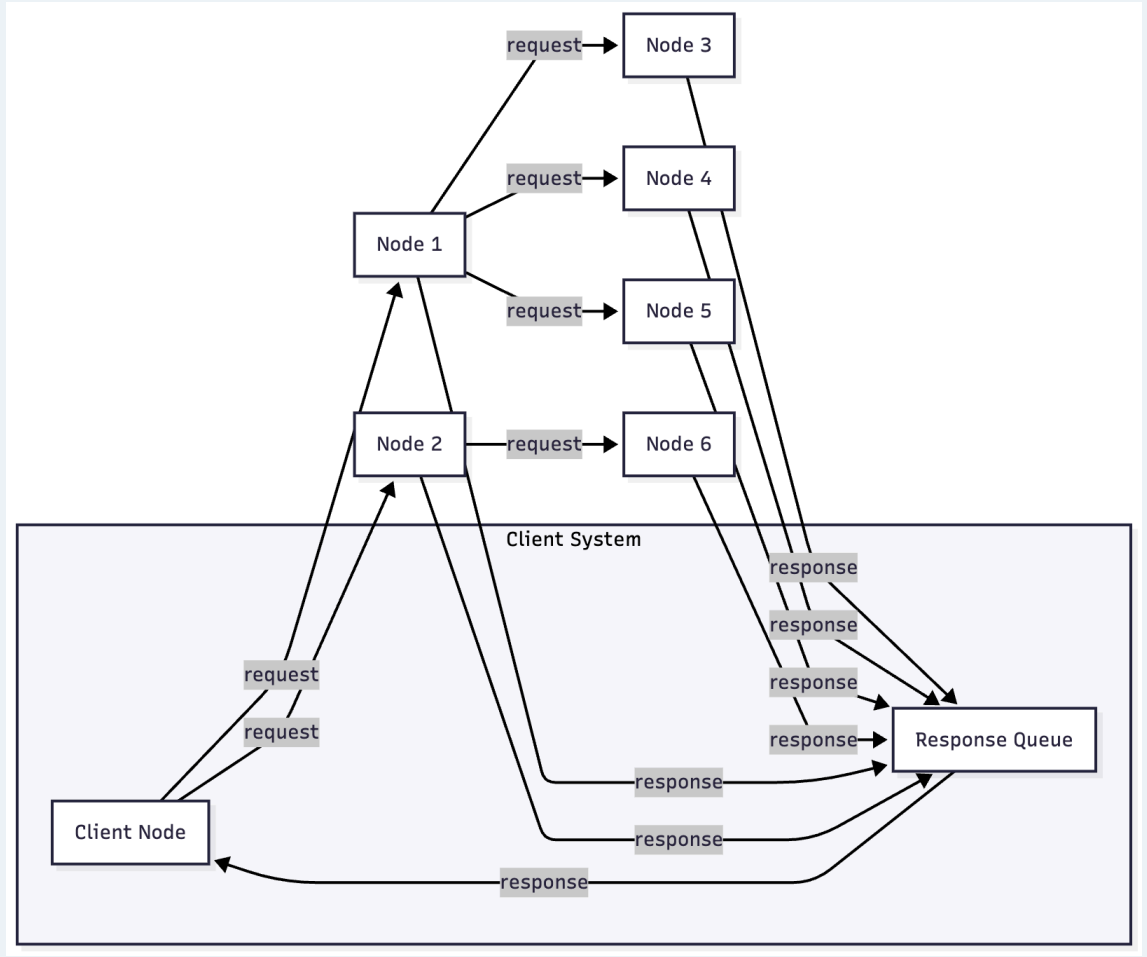
9.6.1.2. HSTP Node Queries

An HSTP Node query is typically sent from a client or another spatial node, and it usually queries the state of a given domain or set of domains (or updates that state through an interface call). Such queries represent the majority of calls in the Spatial Web, and are usually bounded by credentials that determine whether a given querant (an external agent) can in fact get specific information about the domain from the perspective of that domain.

Node queries are often sent to a cluster of different nodes simultaneously, where there is no guarantee that the nodes in question are even in the same network. Such queries get back maps — descriptions of a given node limited by the permission layers and scope of the querant that provide a view of relevant and available items in each domain. It should be noted that such node queries are usually expressed as HSQL, and consequently are filtered prior to being executed in the native query format of the graph.

Such queries can also be submitted to other nodes from a given node as a form of forwarding, though there is an upper bound as to how deep such queries can be, using the HSTP messaging envelope to indicated where the resulting response(s) should be sent.

FIGURE 42: HSTP node queries



In the diagram, a client node sends out a query to nodes 1 and 2, which in turn both sends a query to node 3,4,5 and 6. Each node then sends its response a response queue to the client. It should be noted that in such a query, there is no guarantee of order; the nodes return responses when they have completed the query. The response queue exists to determine whether all items have either returned a response or indicated that they have timed out, and if necessary to transform the response into a form that the client can use.

Just as every node has a client manager, every node also has a response queue, which contains response messages sent over hstp through hsml channels.

Again, it's worth stating that the HSTP Node query ONLY talks to domains within a the node, though it can parameterize requests to just get one specific agent within a domain, and it never communicates directly with the domain graph.

9.6.1.3. UDG Graph Queries

Each domain graph contains a wealth of information, but much of it should not, for one reason or another, be directly exposed to a query. This is where graph queries come in.

A *UDG Graph Query* is a query that is made through the graph manager, typically in the native language for that graph, and then accessible via a named query or update. This query communicates with the full graph that is accessible to the node.

The HSML for the domain includes soft links to dependent graphs, called SERVICES. A *Service* is a graph endpoint that can be queried directly from within the graph language as if it was a specialized named graph (this is supported by most modern RDF forms). Once defined, such graphs are otherwise undistinguishable from normal queries, save that they may have intrinsic latencies. Such endpoints do not necessarily need to be spatial web nodes, they just need to be able to serialize content.

This implies that a spatial web dependent node may need to expose a graph endpoint independent of the node itself. In the current implementation, this would be a SPARQL or SPARQL Update endpoint, but this isn't necessarily a requirement.

The dependent UDF graph neighborhood, unlike the SW Node query neighborhood, is linear — you attach a dependent graph to the independent node, but it's not a multi-tier peer-to-peer network.

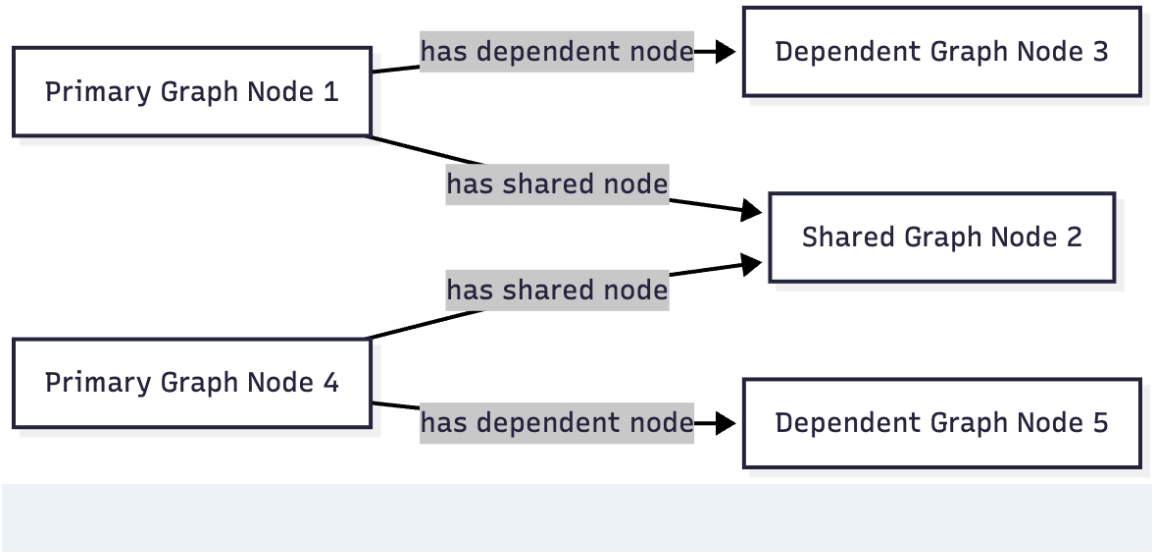
Additionally, there are security risks that you have working with UDG Graph Queries that you don't have working with node queries, as these DO have access to information that is not protected by credentials. As a consequence, USG Graph Queries are considered to be accessible only to the domain or node authority, but not to most users or external agents.

9.6.1.4. Combining the Two

There is no reason why you cannot have both node queries and UDG queries in the system. For instance, you can add multiple dependent graphs to the primary graph in a given domain so that they can distributed across multiple machines. You can then perform a node domain query on the federated distributed graph, treating it as if was a single large graph that just happens to span more than one node. Moreover, there is no formal requirement that a graph extension is a spatial web node, only that it have the appropriate endpoints for graph access.

This approach will likely actually be the norm, especially for large scale domains such as multi-user role playing games, simulations, tours, and large scale IoT systems. If there are common resources (such as schemas, taxonomies, scripts and so forth), then these might be contained in a separate graph server node that is then attached to the primary graph but used by multiple spatial web nodes at once (what's called a shared node).

FIGURE 43: HSTP shared node queries



In the above illustration Node 2 is a shared node between node 1 and node 4.

Dependent nodes are also called content nodes, in that they are able to store content that may not necessarily be HSML specific. *Note that while a Spatial Web node can also expose a graph endpoint interface, content nodes that do not also have a node manager are not considered spatial web nodes per se.*

9.6.2. Named Queries and Security Considerations

Graph extensions are, by their very nature, insecure, primarily because they exist outside of the scope of the credentialling system utilized by hstp. As a consequence, most hstp queries will likely be invoked via a particular name, with parameters passed as a dictionary to the graph manager. This will likely be passed in a manner similar to MCP or the OpenAPI format.

At no time should HSTP directly call the system graph query language; it should always go through some kind of hosted proxy (the graph manager). There are several reasons for this:

- This provides an operational security layer, making it possible to validate an incoming request before performing the query both from a functional and permissions standpoint.
- The invocations better match the declarative visibility principle — an agent can only "see" a given activity if it has the relevant credentials to do so.
- This keeps operational and sensitive data hidden from hacking through HSTP, and it ensures that output can be transformed into "clean" versions that removes such sensitive information before it gets sent back as part of a response.

Named queries and mutations are defined within activities at various scopes. Any query on a domain, for instance, would in turn invoke a graph query that is specific to that

domain, and may be customed to refer to a particular agent (or agent(s)) or place(s) in the system. It's worth noting that the domain itself has access to all aspects of the graph, including the agents and places within the domain.

9.6.3. Understanding Graph Queries

Graph queries are somewhat different from traditional data structures. In a normal query, you typically pass an identifier (or some descriptive metadata), and return a document or a list of identifiers (with metadata) to documents.

In a graph query, however, there are typically two different kinds of query. The first is similar to a search result — a table consisting of fields of value. This is very much akin to a SELECT query in SQL, and this form is useful for generating reports and maps. For instance, given the current architecture, the following query retrieves a list of all of the agents in a given domain (here, a list of people in a given building)

FIGURE 44: Understanding Graph Queries

```
# Sparql

SELECT (?personLabel as ?Person) (?roomLabel as ?Room)
WHERE {
    ?person a Class:Person .
    ?room a Class:Room.
    ?person rdfs:label ?personLabel .
    ?room rdfs:label ?roomLabel .
    ?room Place:hasAgent ?person .
    ?domain Domain:hasAgent ?person .
    ?domain Domain:hasPlace ?room .
} order by ?Room ?Person
```

This generates a table:

TABLE 6: Graph Queries table

Person	Room
Jane Doe	Room 101
Karen Free	Room 101
Bill Barnes	Room 103
Alice Nims	Room 205
Michel Thrush	Room 207
Stephen Blain	Room 302

Person	Room
Leeane Hardin	Room 302

In this case, the select statement reads labeled properties from the WHERE statement, which in turn matches assertions in the graph, resulting in a subgraph.

FIGURE 45: Turtle for Graph Queries

```
# Turtle
Agent:JaneDoe a Class:Person ;
    rdfs:label "Jane Doe" ;
    .
Place:Room101 a Class:Room ;
    rdfs:label "Room 101" ;
    Place:hasAgent Agent:JaneDoe, Agent:KarenFree ;
    .
Place:Floor1 a Class:Floor ;
Place:contains Place:Room101, Place:Room102,
    Place:Room103, Place:Room104 .

Place:ApartmentBuilding1000 a Class:Building ;
    Place:contains Place:Floor1, Place:Floor2,
        Place:Floor3 .

Domain:ApartmentScenario_123 a Class:Domain ;
    Domain:hasAgent Agent:JaneDoe, Agent:KarenFree, ... ;
    Domain:hasPlace Place:Room101, Place:Room102,
     Place:Room103, ... ;
    .
...
```

Construct statements can then be used with the same WHERE statement to generate the subgraphs as Turtle, RDF/XML or JSON-LD, along with additional metadata.

FIGURE 46: Sparql for Graph Queries

```
# Sparql

CONSTRUCT {
    ?person ?personP ?person0.
    ?room ?roomP ?room0.
    ?domain ?domainP ?domain0.
}
WHERE {
    ?person a Class:Person .
    ?person ?personP ?person0.
    ?room a Class:Room.
    ?room ?roomP ?room0.
    ?person rdfs:label ?personLabel .
    ?room rdfs:label ?roomLabel .
```

```
        ?room Place:hasAgent ?person .
        ?building a Class:Building .
        ?building Place:contains+ ?room .
        ?domain Domain:hasAgent ?person .
        ?domain Domain:hasPlace ?room .
        ?domain ?domainP ?domainO.
    } order by ?Room ?Person
```

This will give you the graphs of ALL of the domains with all of the agents in all of the places in each domain, where the agents are people, and the places are rooms.

Most SPARQL queries are constraint queries — they limit the facets so that rather than dealing with a potentially huge graph, you are dealing only with constrained subgraphs. For instance, if you only wanted rooms that are in a specific building, in a certain domain, you could parameterise the query to constrain the query.

For instance, you can use the above query and set the variable `?building` to the IRI `<Place:ApartmentBuilding1000>`. This would give you all occupied rooms in *Apartment Building 1000* across all domains that contain that apartment building.

The same query, however, could also take as an argument the `?person` variable with value `<Person:JaneDoe>`. Since there should only be one active agent in the spatial web with this identifier, this will also tell you what apartment building, floor, and room that particular agent is located in.

This is an important point, because it means that the results of a query will be dependent upon a linear dictionary of named variables and values passed to the query. This flexibility makes SPARQL queries much more powerful than their SQL counterparts, especially when you can also use inferencing to determine the relationships between structures. This dictionary is called a *Query Context.*

9.6.4. Named Queries, Mutations, and Metadata

A SPARQL query is a script that can be stored, assigned a given name (IRI), retrieved, and evaluated wih a given context. Because the query has an IRI, it can also store metadata, including descriptions about what kind of query context it takes, taxonomy classification for identifying the utility of that query, and determination about the fitness of this query compared to others. Additionally, the script in question can be *mutational* — it can change the state of the graph, not just for one particular entity, but all entities that satisfy the query context.

In the RDF graph description, the mutational capabilities are a part of SPARQL Update, which can update the graph dynamically. At the simplest level, this can be used to change multiple states for a given entity simultaneously, in effect locking the graph to mutational changes outside of the scope of its own graph update. This makes such updates *transactional in nature, a key requirement for data consistency. If an update fails, the graph is returned to its previous state.*

This extends to external services as well. If an external update *service* (such as to an IoT device) fails to complete, then this failure will propagate through the query, and any changes made by the update service will be rolled back.

The association of metadata with a given named query or update is significant, because it plays a big role in *discovery*. The domain manager can interrogate all of the agents within its scope, checking the metadata associated with the agent, its place within the domain, and its current state. Similarly, the domain can maintain its own metadata based upon the general domain taxonomy (covered in its own section).

The specific mechanism for adding metadata to an entity is still under discussion, but likely will be of the form Entity:hasTopic.

9.7. Use Cases

The following use cases identify operations that are considered part of the UDG. Because these tend to be interrelated, they are provided primarily in terms of creation and navigational progress.

9.7.1. Setting Up a Spatial Web Node

1. Go to Spatial Web Foundation site and download the spatial web foundation application.
2. At this time, the authority for the site submits a form that will allow you to specify the domain information, authorization, and categorization for the spatial web node itself.
3. At this time, the authority may choose zero or more affiliated networks that the spatial web node may participate in.
4. The default network is the Spatial Web Registry Authority, but this doesn't have to be the one selected. Networks are defined thematically (by topic and language) and as such can be searched and ranked (rated).
5. Each network is managed by an affiliate manager (a node registry) that maintains a cache of links in its primary domain. These node registries
6. Once an ensemble of networks are selected, a request link is sent to each affiliate manager in turn that adds the link to the new node (and it's primary domain) to that of the network in question. If the link is approved then the newly created node is added to the network (see Registering a Node on Affiliation Registry).
7. Once registered, other domains and other entities can be added to the spatial web node (see Creating an Entity).

9.7.2. Generating and Resolving SWIDs

1. Because of the centrality of SWIDS in the Spatial Web architecture, a Spatial Web Node incorporates a SWID generator/resolver as part of the architecture.
2. When an entity is created by the system, a SWID and corresponding documents are created, with the documents (likely public and private keys, or PPKs) stored as graph entities in a cryptographically secured named graph as part of the domain graph.
3. This method exists primarily for convenience and performance as in general the ability to retrieve, parse, and reference external blockchains or similar mechanisms

will be a major performance hit unless that ability is accessible from (and contained in) the graph.

4. This becomes especially important if there is specific surety information within the SWID that is particular to the did.swid method, as will likely be the case.
5. The Credential API handles SWID generation and resolution and acts as a wrapper for abstracting multiple different approaches for SWID management.

9.7.3. Registering a Node on an Affiliation Registry

1. Registering an entity on a spatial web node makes that node available for search according to a specific affiliation credential on that entity. What you are actually registering is a link to the entity in question.
2. That link may contain additional metadata (such as topical or annotative information) that makes it easier to search for similar entities.
3. An affiliate spatial web registry is a spatial web node with a single primary domain that mainly contains links to other entities on other affiliated nodes.
4. To register on such an affiliated registry, a domain authority submits a request containing the link and associated metadata to the registry node, and if the request is accepted, a contract is formed by the registry (containing a new SWID) that contains the relevant links and metadata, and a credential containing this contract is then sent back to the requesting node, where it is stored in the relevant domain (typically the home domain of the node).
5. Note, the registry does not contain the indicated entity, only a reference to that entity in the form of a link on a contract.

9.7.4. Searching an Affiliation Registry (User Client)

1. When a user client is downloaded, the user is directed to a web page that lets the user select one or more affiliation networks (including the Spatial Web Registry Authority).
2. Alternatively, the user can download affiliation packages that contain links to common affiliation servers in a bundle.
3. Affiliation servers periodically read each of spatial web node within their affiliation and extract links and metadata of those entities that have been given an affiliation credential (typically domains, but they could be other resources).
4. The client sends a request that can be decomposed as taxonomic metadata (or by indexing descripions and labels via vector search or AI), which can then be used to retrieve the links that best match the search.
5. It should be noted that the affiliation registries will generally not be completely current (they will typically iterate through their affiliated nodes on a daily basis, but probably not much faster than that), and this will only pick up the domains and other entities that have received an affiliation credential.

6. It is also possible (albeit probably not advisable) to retrieve a list of affiliates directly and query each node independently. It would require the user client, however, having the relevant credentials for each domain on each node

9.7.5. Refreshing a Registry

1. Periodically, a registry will iterate through its contracts and request updated metadata from an affiliated node for entities that have the relevant affiliated credentials.
2. Not all (not even most) entities on a given node will have these credentials, only those that need to be identified by the affiliation registry.
3. Similarly, a node may be self-affiliated, with the home domain containing contracted links for entities that are considered important enough to be visible for search on that domain. This can be considered the directory for that node. A registry is a directory for domains (formally entities) that are external to that node.

9.7.6. Logging Into a Domain

Requirements: DSA-5

1. An external agent (such as a user agent) will have a link to a domain, presented as a SWID or SWURL, and will send this (with any appropriate metadata) to the resolved spatial web node.
2. The SWNode (the node manager) receives a request to initiate a connection, determines whether the relevant domain exists, and determines whether the external agent already has a proxy agent on the system representing that external agent.
3. If the agent exists and the credential to access that agent is cached for (perhaps within, TBD) the relevant domain, then a channel is established between the external agent and the proxy agent, with a message then sent back to the external agent providing confirmation.
4. If no credential exists (either this is a new user agent or the credential has expired), the SWNode sends a message back to the external agent requesting credentials. In the case of expiry, this is just a revalidation, new credentials are set up and the connection is made.
5. If no agent exists for that domain, then a new agent is registered, typically at the home place for that domain, once credentials have been created and confirmed. This is the proxy agent for that user agent on that domain.

9.7.7. Creating an Entity []

1. Request a credential to create a particular entity (domain, agent, place, etc.).
2. If the credential is valid, this returns a SWID for the parent entity.

3. Submit an HSML document that describes the entity, using an HSML posted message that includes the containing SWID (this may be accomplished via some form of an editor)
4. The HSML document is checked for validity, and is rejected if it fails a validity check.
5. If the document is accepted, the document is created within a named graph.
6. For all entities within the named graph, SWIDs are created and attached to each entity.
7. The named graph identifier is then attached to the parent entity.
8. At the time of creation, an entity MUST be assigned an *Internal State domain or IS Domain (see Changing Internal State Domain).*
9. At the time of creation, an entity MAY be assigned a *level of detail domain or LoD Domain (see Changing Level of Detail).*

   NOTE *It is possible that this will need to be changed to MUST and needs further discussion.*

9.7.8. Attach a Credential to an Entity

1. If an agent has a relevant mutation credential on a given entity (meaning that they can edit that node), the agent can attach a credential referencing the SWID of that entity through HSTP.
2. If the credential is an affiliation credential, then the entity becomes visible through queries against that node if the querant has the corresponding affiliation key.
3. A public entity is one that has a Public Affiliation Key, meaning that it is visible to anyone on the spatial web if they reference the spatial web node. This will generally apply to domains.
4. All immediate entities within a domain will share the credentials within that domain. If a subdomain exists on an entity, the entity needs to extend the credential to that domain explicitly.

9.7.9. Invalidate an Entity

1. An entity is made invalid by setting the :isInactive flag (typically through a sparql update).
2. An inactive entity remains in the system but is no longer visible to queries (all queries check the inactive flag for that entity).
3. When an entity is made inactive, the datetime is noted, and after a system settable time, the entity will be purged. Note that if an entity has a subordinate or linked domain, that domain will NOT be made inactive (there may be other references to the subdomain).
4. All queries against an entity must specifically check to see if the entity is valid before returning it as part of a search result.

9.7.10. Querying an Entity

1. All entities have a default Query Activity that will retrieve a JSON-LD representation of that entity (this may not be a faithful copy of the internal state of the entity).
2. The editor of that entity may incorporate one or more override activities that provides different representations based upon parameters sent within the HSTP request message.
3. The querant may request that the query be made subscribable, which means that a new message is passed every time a change is made to the state of the entity in question.

9.7.11. Querying a Specific State of an Entity

1. The querant can request a specific state variable for a given entity. This will retrieve a JSON structure containing the variable and it's associated value.
2. As with querying an entity, querying the state of an entity can be done asynchronously using a pub/sub protocol. This will return information about the state periodically as it changes.
3. A query can also be made to retrieve the entity state array, either once or upon state changes.
4. Any asynchronous query will return an identifier for that query, and the calling agent may cancel the query by passing back that identifier.

9.7.12. Modifying the Specific State of an Entity

1. If a particular state of an entity is modifiable, then this will cause a mutation event to occur that will instruct the entity to initiate a mutation activity to occur.
2. In the simplest (default) case, this just updates the value of the state in the graph.
3. If the agent is autonomous, this will cause the agent manager to attempt to align the agent to the requested condition.
4. If the agent is also bound to a physical twin, the agent manager will make the attempt to change the state of the physical twin before updating. If this fails, an error will be raised, and any changes will be rolled back.

9.7.13. Subscribing to a State of an Entity

1. Subscribing to the state of an entity is the same as querying the state of an entity asynchronously.
2. When a state changes in the subscribed entity, the subscribing entity will receive a notification (via domain.d) that can be caught with a subscribed state update event activity (the default is to do nothing).

3. If the publishing entity is located on a different node, the message will be routed through hstp.d first, and then to the relevant entity.
4. The first message returned from the publisher will be the current state, even if that state has not changed.
5. The exact contents of various entity state descriptors are TBD, but will likely be a stream of contained entity messages (filtered by specific state if this is requested).
6. Typically, such messages will be managed over channels, possibly as a part of a message queue.

9.7.14. Extending an Entity Graph

1. The graph for a given entity (primarily domains) may be extended by use of the hsml:include property.
2. This provides a (generally) read-only ability to query an exterior graph, either from a different domain on the current machine, a different domain from an external domain, or a non-spatial web graph resource.
3. This is frequently used to access domains containing collections of commonly defined entities (such as places, activities, agents, contracts and so forth).
4. Such extensions typically require having the relevant credentials to access the external servers, and more than likely will be associated with affiliated nodes.

9.7.15. Importing an Entity Graph

1. An imported graph is one that is copied from an entity outside the existing domaing graph. Unlike extended graphs, imports effectively copy the contents of the given external entity but assign new SWIDs. SWURLs are typically fragments, so take on a new identifier (via it's HTTP domain).
2. Importing a domain is the same as creating a domain, including assigning new SWIDs as needed.
3. Importing a domain creates a copy of that domain. This will typically be use when a domain acts as the "template" that is then filled out parametrically, such as that used by games or simulations.
4. Importing a domain is considered an HTTP operation, while extending (including) a domain is part of UDG.

9.7.16. Interacting with the Domain: User Agents

1. A *user agent* is an agent that represents the interest and focus of an external agent within the domain. It is typically the *thing* or *person* that navigates the domain on behalf of that external agent.

2. When an external agent "logs in" to a domain, the domain manager establishes a user agent representing that external agent, adding the user agent's credential to the domain credential store.
3. If no user agent exists within that domain for that external agent, the domain manager creates that user agent and adds them to the *home place for that domain. This can be thought of as the landing place for new agents.*
4. If a user agent already exists for the external agent, they will already be sited within the domain at a specific place. This establishes the context for that user agent within the domain.
5. A user agent can interact with other agents within a given place, or with agents within an linked neighborhood (agents on a place that is directly connected to the current place). This is called the *interactive neighborhood*.
6. Within the interactive neighborhood, the *state matrix and activity matrix of all other agents in that neighborhood become visible. The activity matrix indicates all activities that a given agent can perform, relative to the user agent, while the state matrix identifies the state that is exposed to the user agent based upon the same mechanisms (typically a credential).*
7. It should be noted that such interactions are reciprocal — the user agent also exposes their state and activity matrices to other agents in the same way.
8. The interactive neighborhood exists for two reasons — it more closely reflects the reality in which people have personal spaces that determine how they specifically interact, and it serves to reduce the overall complexity of any given domain. Note that if a communication link exists between two agents, this is considered part of the interactive neighborhood for each of those agents.

9.7.17. Activating an Agent's Activity

1. In the case of a user-agent, the external agent is presented an activity matrix that indicates what specific activities the user-agent can perform. One of these activities as `selecting_an_activity`. This allows the agent to choose one from a set of activities that may be available of another agent, and make it the focus for subsequent actions (this may be set up on the agent as the hsml:targetEntity)
2. Once an activity is selected, the user agent may then `activate_an_activity`. This is a signal to the targeted agent that the targeting agent is requesting that an activity be accomplished.
3. The targeted agent that evaluates the request and, if it is within its capability and goals, will return a contract to the targeting agent with its conditions. If the conditions are acceptable to the targeting agent (for instance, if a fee is involved and paid, establishing a credential) then the activity will be initiated.
4. Note that a contract can be extended to cover all activities that are visible to the targeting agent, and can remain in force until explicily terminated. This can reduce the negotiation process for subsequent calls to invoke other activities of a given agent.

5. The targeted agent will then asynchronously perform the activity until the activity is completed, whereupon it notifies the targeting agent that the activity has been completed.
6. If the targeted agent is unable to complete the activity, then it performs a forfeit activity (such as reimbursing the targeting agent) according to the terms of the contract.
7. The activating agent can also perform an action to terminate the contract, but only once the contract has been either satisfied or forfeited. Most simple contracts are self-terminating.

9.7.18. Maintaining History

1. Maintaining history is handled in one of two ways — reifications and sampling, with four total options:
   - *Reification* involves the creation of assertions concerning the changes in the state of the various entities within a given domain. Reification can provide an exact replay of changes over time, but at the cost of performance and additional space.
   - *Sampling* involves the periodic sampling of the state matrix of one or more of the entities in a given domain, persisting them to an external channel. Sampling is more efficient, but it loses resolution.
   - *Neither.* An entity simply does not maintain a history because it doesn't need to.
   - *Both.* Persistence tracking via reification allows for the replay of a given domain while external reporting can be done on the entity. This is the most comprehensive, but it is also the most processor intensive.
2. Reification is part of the graph services and is managed via graph.d. Sampling is part of the domain services and ties in more closely with HSTP and its associated daemon.

9.7.19. Changing Internal State Domain of an Entity

1. A given non-domain entity (such as a place or an agent) may have as *Internal State Link* to a different domain that represents the internal state of that entity.
2. Such a domain may be empty of child agents or places.
3. The current implementation of such an ISL domain is a named graph, but this may change based upon system representation of data structures.
4. The ISL graph is used primarily to represent the internal state of that entity, though it can also (especially in the case of Places) represent a zoomed in view of the entity (such as a country place showing a detail of the various roads, cities, etc. within that country).
5. ISL domains may have a state matrix that is similar to that of an non-domain entity, which is used primarily to store measurements and intermediate values from the interaction of the components within the subdomain.

6. In general, access to the ISL is limited to administrators, and on user agents (spatial web browsers) will have a specialized entry point because of this.

9.7.20. Changing Level of Detail Graphs of an Entity

1. A given non-domain entity (such as a place or an agent) may have one or more *Level of Detail Links* (*LoDs*) to a different domain that represents a drill-down of subcomponents of that entity.
2. Unlike an ISL, a Level of Detail link is typically used to provide different representations or subsystems for a given entity. A country (a place) for instance, may have one LoD showing critical population centers, another showing primary traffic routes, another showing watersheds and other features.
3. An LoD domain differs from an internal state domain primarily in that it does not communicate state changes back to the parent entity. This is important because it reduces synchronization issues.
4. As a rule of thumb, if there is a child domain of a given non-domain entity that has multiple overlapping and interconnected systems, these would best be contained within a single ISL, while if there are mostly disconnected systems (such as the plumbing vs. electrical system in a house), this would work better as multiple LoD systems.

9.7.21. Subscribing to a Channel

1. A channel is an entity, and utilizes the same mechanism that any entity does when receiving changes in state.
2. In this particular case, an inbound channel has a queue that receives messages. When a message comes in, any entity that has subscribed to this channel will received a notification that new messages are in the queue that are specifically addressed to that entity.
3. A domain or entity within that domain may also publish to a channel through an activity. This is what is used for multiagent communication.

9.7.22. Moving an agent from one domain to another

1. Agents, especially proxy agents, are typically mobile. When a proxy agent initiates a link connecting two places, a link between the old place and the agent will be augmented to indicate that the link is no longer active (likely through reification, but this is an implementation detail).
2. If a link has an active credential requirement, then the credential must be presented or satisfied before the transfer can be initiated.
3. Once the credentials have been satisfied, the connection between the place and the agent will be set as deprecated (likely through a reification), and a new connection is established between the target place and the agent.

4. If the new place is not located on the same node, then a check is made whether there exists an agent representing the same user agent on the target node. If there is, then the agent is "revived" and any relevant history data is transferred to the new node, then a new connection is established between the target place on the new node and the proxy agent on *that* machine. (This is primarily for performance purposes).
5. The deprecated connection will also include a forwarding address to the new agent. This way, if an agent is known but it has moved "off-node", then the movement through different nodes can be traced.

9.7.23. Transporting an Agent Via Another Agent

1. An agent with an associated subdomain can "transport" another agent within that subdomain. This may be the case when an agent is acting as a container or carrier.
2. Moving a given agent into another agent's subdomain is the same as moving an agent from one domain to another. From the standpoint of the initial domain, the "carried" agent is effectively no longer in scope of the carrier's superdomain.
3. When an agent moves, the link to the subdomain for that agent remains the same — even if the agent moves from one node to another.
4. The carrier agent can release the carried agent in a new place, at which point this is treated as a transfer of the carried agent from one domain to another.

9.7.24. Creating a New Place

1. Create an HSML Place definition and instantiate it (see Creating an Entity), appending it to the relevant domain through the `hsml:hasPlace` predicate.
2. If the place is intended to be a proxy for an established place, create the relevant proxied link (e.g., Place:Earth).
3. If the new place needs links to existing places, create link children (either directly on the link or indirectly through an object) on both the current place and on relevant backlinks (if the link is not bidirectional).
4. Once links are created, a domain function can be identified called resolve_links, which creates backlinks if a link is bi-directional.
5. Note that links are sensitive to the types of agents involved. For instance, in a chess game simulator there may be links of type rank, file, diagonal, and knight (the L shaped link) between different squares, and the movements that are possible will consequently be composed of the set of all paths that can be made to a given square from the starting square based upon the piece. The set of all possible paths that a given piece (agent) can take is known as an ensemble, and this represents the local hyperspace of that piece relative to the agent type.
6. As with other entities, places can be deprecated, typically by reification.

9.7.25. Creating an Entity Instance

1. An *Entity Instance* is a copy of an existing entity that is used as a template. It is frequently used in those situations where you have multiple different instances of a given environment, such as a game or simulation.
2. An entity instance can only be created if the entity or some subcomponent of that entity is not a shadow for an IoT device or similar physical system that can be mutated (such as turning on a light in a smart room).
3. The domain.d API includes a call to create an entity instance, which will take the current entity definition passed as a SWID and then instantiate a new instance that generates independent SWIDs and relevant identifiers.
4. Entity instances do not necessarily copy affiliation credentials (this is a flag), meaning that while the original entity may be visible to an affiliation search, the instances do not necessarily need to be, though domain nodes will still appear in the landing page of the node directory, if this has been set up.

9.7.26. Using the Node Domain Directory

1. A user can query the spatial web node for all of the domains for which the user agent has credentials. Typically this will be supported in the node domain, which has a specific agent that allows for generating and searching these domains (a *domain directory*).
2. The domain directory is a kiosk control that can also be used to view and filter the domains by their relevant topics, and provide relevant summaries and metadata for each domain.
3. When invoked as JSON-LD (say via discovery applications), the domain directory generates either an HSML, Atom or RSS feed that contains this same metadata.

9.7.27. Rendering an Entity

1. When a query is made on domains or other entities, the request may incorporate a content-type parameter.
2. If a content-type is provided, The results of the query along with the content-type are then passed to the render.d manager.
3. The manager checks to see if there is a rendered plugin that matches the content type. If there is, the HSML is passed to the plugin to generate an appropriate output; if not, then the content continues as HSML.
4. The output is then attached to the HSTP response message as an attachment, then sent to the requisite user-client.

9.7.28. Handle Fast/Slow State Changes

1. Each domain has a heartbeat that determines how frequently up updates are made (and how frequently external systems are polled). When creating the domain, the heartbeat can be established as a property on the domain, and can be increased or decreased as need be.
2. For those situations where the domain does not incorporate IoT devices, this heartbeat can usually be fairly fast, as the mechanisms for transmitting information exist primarily in the same process.
3. For those domains where external services or IoT device connections exist, the heartbeat can generally be slowed dow (or sped up) to handle polling or publication/subscription (pub/sub) type architectures.
4. Please note that the spatial web is primarily intended to be a predictive systems, involving a large amount of contextual data, rather than a close monitoring system.
5. It is possible (though the exact mechanism is still TBD) for a service to spawn a direct connection to an Iot device or similar fast moving system, one that bypasses the normal domain calles. In such cases, simple filters may be placed on incoming messages that allow for specific signals to be detected which then prompts an update back into the domain manager.

9.7.29. Replication and Failover

1. The specific implementation of replication is dependent upon the particular knowledge graph store in question. The assumption here is that whatever KG store will likely have some native replication for multiple servers supporting failover by periodically streaming triples that are active as part of revisions to the graph. This will likely be expressed in more detail as prototypes reach a sufficient level of maturity.

Annex A
(normative)

Annex Requirements Allocation

A.1. IEEE requirements mapped to UDG design spec

The table lists requirements in SWF STD-1:2025 (v3.3.2) September 26, 2025 that are allocated to the UDG Design Specification. The Table also includes the clause of the UDG Specificaiton where the System requirements are addressed.

TABLE A.1: IEEE requirements mapped to UDG design spec

IEEE Clause	Requirement	UDG Clause
5.2.3.2.2.	Enable discovery of virtual representations of physical entities	
5.2.3.3.2.	Enable discovery of physical and virtual entities via discovery services	
6.3.3.4.	Validate SWIDs generated using SWID Method	
6.3.3.4.	Include Spatial Web registration service	
1.2.5	Register all SWIDs in Spatial Web Registry	
1.2.6	Enable domain verification and validation	
1.2.7	Support flexible SWID generation	Entity Registries
6.3.3.4		1.2.8
Ensure SWID uniqueness	UDG SWIDs	6.3.3.4
	1.2.9	Maintain SWIDs in Registry
Entity Registries	6.3.3.4	
1.2.10	Maintain resilient operations	UDG and HSTP

IEEE Clause	Requirement	UDG Clause
6.3.4.9		1.2.11
Provide distributed operations	UDG and HSTP	6.3.5.3
	1.2.12	Provide seamless domain interactions
UDG and HSTP	6.3.5.3	
1.2.13	Implement registration processes	Node Registries
6.3.5.3		1.2.14
Enable varied access methods	UDG and HSTP	6.3.5.3
	1.2.15	Register and manage ACTIVITIES
UDG Activities	6.4.4.8	
1.2.16	Record HSML ACTIVITIES	UDG Activities
6.4.4.8		1.2.17
Support high-performance networking	UDG and HSTP	7.1.3
	1.2.18	Enable automatic node discovery
Node Registries	7.1.3	
1.2.19	Scale to internet level	Node Registries
7.1.3		1.2.20
Include Spatial Index Servers	Hyperspace, Entity Registries	7.2.2

IEEE Clause	Requirement	UDG Clause
	1.2.21	Manage entity updates
UDG and HSTP	7.2.2	
1.2.22	Manage rapid entity changes	UDG and HSTP
7.2.2		1.2.23
Manage slow-changing entities	UDG and HSTP	7.2.2
	1.2.24	Handle consensus latency
UDG and HSTP	7.2.2	
1.2.25	Implement specified use cases	UDG and HSTP, Representations

STD-1 Clause STD-1 Requirement UDG Clause 5.2.3.2.2. UDG shall enable discovery of the virtual representation of physical entities. 5.2.3.3.2. UDG shall enable discovery of physical and virtual entities via discovery services. 6.3.3.4. UDG shall validate SWIDs generated using SWID Method prior to issuance, e.g., assess uniqueness. 6.3.3.4. UDG shall include a Spatial Web registration service for Public and Top domains. 6.3.3.4. UDG shall, for audit purposes, register all SWIDs related to all public and top domains in a Spatial Web Registry. 6.3.3.4. UDG shall enable verification and validation services for domains prior to their registration. 6.3.3.4. UDG shall support the generation of SWIDs one at a time, such as for Top Domains, or generate many at a time, such as for Public Domains. 6.3.3.4. UDG shall ensure SWID uniqueness. 6.3.3.4. UDG shall ensure that SWIDs are maintained in the Spatial Web Registry. 6.3.4.9. UDG operations shall be resilient to inconsistencies in relationships between nodes and in the content of nodes. 6.3.5.3. UDG shall provide for distributed operations of the UDG including propagation of changes and consistency. 6.3.5.3. UDG shall provide Spatial Web Domain interactions that are seamlessly managed and integrated. 6.3.5.3. UDG shall implement Spatial Web Domain registration processes as defined in clause 6.3.6.

6.3.5.3. UDG design and procedures shall enable a range of methods for accessing the UDG from basic, open access to UDG access services with enhanced value in accord with economic exchange, e.g, fee, advertising, etc. 6.4.4.8. UDG shall provide the capability to register and manage ACTIVITIES that are associated with AGENTs, reflecting their capabilities and permissions. 6.4.4.8. UDG shall keep a record of HSML ACTIVITIES that were executed as part of a Contract, providing a history of the Activity, verification of the execution of the Activity, and enabling the tracking of the Activity's progress. 7.1.3. UDG shall be designed to operate with communication

network performance where bandwidth ranging from hundreds of gigabits per second to several terabits per second (i.e., having latency in the sub-millisecond range). 7.1.3. UDG shall provide mechanisms for automatic discovery of nodes, and their properties and capabilities as well as the means to access them. 7.1.3. UDG shall support the ability to accommodate an increasing number of connectivity endpoints, reaching internet scale. 7.2.2. UDG shall include Spatial Index Servers that make maps ranging from simple SQL indexes to graph-based databases to widely adopted and standard spatial indexing services which deliver spatial indexing. 7.2.2. UDG shall manage entity replication and update with consideration of how quickly the entities are changing. 7.2.2. UDG shall manage rapidly changing entities using a peer-to-peer methodology between Spatial Servers, managed by cloud instance(s), but bound by spatial CONTRACTs stored in a DLT Spatial Domain. 7.2.2. UDG shall manage slow-changing cross-ledger entities and CONTRACTs on a distributed ledger. 7.2.2. UDG System may incur latency when achieving consensus. 7.4.2. UDG shall implement the use cases: 7.4.4, and 7.4.11.

A.2. UDG requirements allocated to components

A.2.1. Spatial Web components

Requirements defined in this UDG Design Specification are allocated to these components of the Spatial Web design:

- Hyperspace Modeling Language (HSML) (SWF STD-02)
- Hyperspatial Transaction Protocol (HSTP) (SWF STD-03)
- SWID Documents (SWF STD-04)
- Spatial Web DID method (SWF STD-05)
- Spatial Web Registry
- UDG Node Design
- Agent Verification plan
- DSA (Domain-Specific Architectures)
- SWG (Spatial Web Governance)

A.2.2. UDG Design Specification requirements allocated to HSML

TBD

A.2.3. UDG Design Specification requirements allocated to HSTP

TBD

A.2.4. UDG Design Specification requirements allocated to SWID Docs

TBD

A.2.5. UDG Design Specification requirements allocated to did:swid method

TBD

A.2.6. UDG Design Specification requirements allocated to Registry

TBD

A.2.7. UDG Design Specification requirements allocated to UDG Node

TBD

A.2.8. UDG Design Specification requirements allocated to Agent

TBD

A.2.9. UDG Design Specification requirements allocated to DSA

TBD

A.2.10. UDG Design Specification requirements allocated to SWG

TBD

Annex B
(informative)

Bibliography

[1] Nudging for Autonomous Systems, [NO INFORMATION AVAILABLE]

[2] ISO 19111:2019, International Organization for Standardization. *Geographic information — Referencing by coordinates*. Third edition. 2019. Geneva. https://www.iso.org/standard/74039.html.

[3] ISO 19135-1:2015, International Organization for Standardization. *Geographic information — Procedures for item registration — Part 1: Fundamentals*. First edition. 2015. Geneva. https://www.iso.org/standard/54721.html.

[4] ISO/IEC TR 23188:2020, International Organization for Standardization and International Electrotechnical Commission. *Information technology — Cloud computing — Edge computing landscape*. First edition. 2020. Geneva. https://www.iso.org/standard/74846.html.

[5] ISO/IEC/IEEE 24765:2017, International Organization for Standardization, International Electrotechnical Commission and Institute of Electrical and Electronics Engineers. *Systems and software engineering — Vocabulary*. Second edition. 2017. Geneva. https://www.iso.org/standard/71952.html.

[6] ISO/IEC/IEEE 42010:2022, International Organization for Standardization, International Electrotechnical Commission and Institute of Electrical and Electronics Engineers. *Software, systems and enterprise — Architecture description*. Second edition. 2022. Geneva. https://www.iso.org/standard/74393.html.

[7] FRISTON, Karl J, Maxwell JD RAMSTEAD, Alex B KIEFER, Alexander TSCHANTZ, Christopher L BUCKLEY, Mahault ALBARRACIN, Riddhi J PITLIYA, Conor HEINS, Brennan KLEIN, Beren MILLIDGE, Dalton AR SAKTHIVADIVEL, Toby ST CLERE SMITHE, Magnus KOUDAHL, Safae Essafi TREMBLAY, Capm PETERSEN, Kaiser FUNG, Jason G FOX, Steven SWANSON, Dan MAPES and Gabriel RENÉ. Designing ecosystems of intelligence from first principles. *Collective Intelligence*. vol. 3 no. 1. SAGE Publications. 2024. DOI: DOI 10.1177/26339137231222481. ISSN: issn.print 2633-9137. ISSN: issn.print 2633-9137. http://journals.sagepub.com/doi/10.1177/26339137231222481.

[8] JI, Xiaoyu, Yibing CAO, Jiangshui ZHANG and Xinke ZHAO. STSE: Spatio-temporal state embedding for knowledge graph completion. *Knowledge-Based Systems*. vol. 317, p. 113469. Elsevier BV. 2025. DOI: DOI 10.1016/j.knosys.2025.113469. ISSN: issn.print 0950-7051. https://linkinghub.elsevier.com/retrieve/pii/S0950705125005167.

[9] GILMAN, J., A. HASSAN and N. ZIMMERMAN. The Case for a Centralized Earth Observation Vector Embeddings Catalog. Element 84. 2025. https://github.com/Element84/vector-embeddings-catalog-whitepaper/blob/main/VectorEmbeddings_WhitePaper_June2025.pdf.

[10] BLATTNER, M., V.J. PASQUARELLA, M.R. KAZMIERSKI, M. SAMSIKOVA, W.J. RUCKLIDGE and C.F. BROWN. Embeddings of Earth: A Foundation Model for Planetary Representation Learning. 2024. https://arxiv.org/abs/2412.05600.

[11] BROWN, C.F., M.R. KAZMIERSKI, V.J. PASQUARELLA, W.J. RUCKLIDGE and M. SAMSIKOVA. AlphaEarth Foundations: An embedding field model for accurate and efficient global mapping from sparse label data. 2025. https://arxiv.org/abs/2507.22291.

[12] BLATTNER, M. Tangential Action Spaces: Geometry, Memory and Cost in Holonomic and Nonholonomic Agents. 2025. https://arxiv.org/abs/2509.03399.

[13] SKUPIN, A. BigKnowledge: Not Just a Metaphor. Sage Research Methods Community. 2024. https://researchmethodscommunity.sagepub.com/blog/concept-grant-update-bigknowledge.

[14] KANERVA, Pentti. Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors. *Cognitive Computation*. vol. 1 no. 2, pp. 139–159. Springer Science and Business Media LLC. 2009. DOI: DOI 10.1007/s12559-009-9009-8. ISSN: issn.print 1866-9956. ISSN: issn.electronic 1866-9964. http://link.springer.com/10.1007/s12559-009-9009-8.

[15] MIKOLOV, T., I. SUTSKEVER, K. CHEN, G. CORRADO and J. DEAN. Distributed Representations of Words and Phrases and their Compositionality. 2013. https://arxiv.org/abs/1310.4546.

[16] GitHub. *word2vec*. https://github.com/dav/word2vec.

[17] World2Vec, LERER, A., L. WU, J. SHEN, T. LACROIX, L. WEHRSTEDT, A. BOSE and A. PEYSAKHOVICH. PyTorch-BigGraph: A Large-scale Graph Embedding System. In. n.p.: 2019. World2Vec. https://arxiv.org/abs/1903.12287.

[18] TEMPLETON, A., T. CONERLY, J. MARCUS, J. LINDSEY, T. BRICKEN, B. CHEN, A. PEARCE and Anthropic. Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet. In. n.p.: 2024. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[19] JHA, R., C. ZHANG, V. SHMATIKOV and J. X. MORRIS. Harnessing the Universal Geometry of Embeddings. In. n.p.: 2025. https://arxiv.org/abs/2505.12540.

[20] HUH, M., B. CHEUNG, T. WANG and P. ISOLA. The Platonic Representation Hypothesis. In. n.p.: 2024. https://arxiv.org/abs/2405.07987.

[21] GAVRANOVIĆ, B., P. LESSARD, A. DUDZIK, T. VON GLEHN, J. G. M. ARAÚJO and P. VELIČKOVIĆ. Position: Categorical Deep Learning is an Algebraic Theory of All Architectures. In. n.p.: 2024. https://arxiv.org/abs/2402.15332.

[22] HULL, Vanessa and Jianguo LIU. Telecoupling: A new frontier for global sustainability. *Ecology and Society*. vol. 23 no. 4. Resilience Alliance, Inc. 2018. DOI: DOI 10.5751/es-10494-230441. ISSN: issn.print 1708-3087. https://www.ecologyandsociety.org/vol23/iss4/art41/.

[23] MANNING, Nicholas, Yingjie LI and Jianguo LIU. Broader applicability of the metacoupling framework than Tobler's first law of geography for global sustainability: A systematic review. *Geography and Sustainability*. vol. 4 no. 1, pp. 6–18. Elsevier BV. 2023. DOI: DOI 10.1016/j.geosus.2022.11.003. ISSN: issn.print 2666-6839. https://linkinghub.elsevier.com/retrieve/pii/S2666683922000748.

[24] NOY, Natasha, Yuqing GAO, Anshu JAIN, Anant NARAYANAN, Alan PATTERSON and Jamie TAYLOR. Industry-scale knowledge graphs. *Communications of the ACM*. vol. 62 no. 8, pp. 36–43. Association for Computing Machinery (ACM). 2019. DOI: DOI 10.1145/3331166. ISSN: issn.print 0001-0782. ISSN: issn.electronic 1557-7317. https://dl.acm.org/doi/10.1145/3331166.

[25] WELLER, O., M. BORATKO, I. NAIM and J. LEE. On the Theoretical Limitations of Embedding-Based Retrieval. 2025. https://arxiv.org/abs/2508.21038.

[26] PENTLAND, A. *Social Physics: How Good Ideas Spread—The Lessons from a New Science*. n.p.: Penguin Books. 2015. https://www.penguinrandomhouse.com/books/314230/social-physics-by-alex-pentland/.

[27] HAYEK, F.A. The Use of Knowledge in Society. vol. 35, pp. 519–530. 1945. https://www.jstor.org/stable/1809376.

[28] ZOLLMAN, K.J.S. *Network Epistemology*. 2025.

[29] WEI PAN, WEN DONG, M. CEBRIAN, TAEMIE KIM, J. H. FOWLER and A. S. PENTLAND. Modeling Dynamical Influence in Human Interaction: Using data to make better inferences about influence within social systems. *IEEE Signal Processing Magazine*. vol. 29 no. 2, pp. 77–86. Institute of Electrical and Electronics Engineers (IEEE). 2012. DOI: DOI 10.1109/msp.2011.942737. ISSN: issn.print 1053-5888. http://ieeexplore.ieee.org/document/6153598/.

[30] ALBARRACIN, M., S. DE JAGER, D. HYLAND and S.G. MANSKI. The Physics and Metaphysics of Social Powers: Bridging Cognitive Processing and Social Dynamics, a New Perspective on Power through Active Inference. 2025. https://arxiv.org/abs/2501.19368.

[31] VEISSIÈRE, Samuel P. L., Axel CONSTANT, Maxwell J. D. RAMSTEAD, Karl J. FRISTON and Laurence J. KIRMAYER. Thinking through other minds: A variational approach to cognition and culture. *Behavioral and Brain Sciences*. vol. 43. Cambridge University Press (CUP). 2019. DOI: DOI 10.1017/s0140525x19001213. ISSN: issn.print 0140-525X. ISSN: issn.electronic 1469-1825. https://www.cambridge.org/core/product/identifier/S0140525X19001213/type/journal_article.

[32] TOMASELLO, M. *The Evolution of Agency: Behavioral Organization from Lizards to Humans*. n.p.: MIT Press. 2022.

[33] NAGEL, Jennifer. Natural Curiosity. In: LOGINS, Artūrs and Jacques-Henri VOLLET (eds.): *Putting Knowledge to Work*. n.p.: Oxford University PressOxford. 2024. pp. 170–200. DOI: DOI 10.1093/9780191976766.003.0007. ISBN: ISBN 9780192882370. ISBN: ISBN 9780191976766. https://academic.oup.com/book/57839/chapter/471730323.

[34] FRISTON, Karl J., Marco LIN, Christopher D. FRITH, Giovanni PEZZULO, J. Allan HOBSON and Sasha ONDOBAKA. Active Inference, Curiosity and Insight. *Neural Computation*. vol. 29 no. 10, pp. 2633–2683. MIT Press. 2017. DOI: DOI 10.1162/neco_a_00999. ISSN: issn.print 0899-7667. ISSN: issn.electronic 1530-888X. https://direct.mit.edu/neco/article/29/10/2633-2683/8300.

[35] MILGRAM, S. The Small World Problem. pp. 60–67. Ziff-Davis Publishing Company. 1967.

[36] SEARLE, J.R. *The Construction of Social Reality*. n.p.: Free Press. 1995.

[37] VERHAGEN, H., M. NEUMANN and M.P. SINGH. Normative Multiagent Systems: Foundations and History. In. n.p.: College Publications. 2018. pp. 3–25.

[38] VINITSKY, Eugene, Raphael KÖSTER, John P AGAPIOU, Edgar A DUÉÑEZ-GUZMÁN, Alexander S VEZHNEVETS and Joel Z LEIBO. A learning agent that acquires social norms from public sanctions in decentralized multi-agent settings.

Collective Intelligence. vol. 2 no. 2, p. 263391372311620. SAGE Publications. 2023. DOI: DOI 10.1177/26339137231162025. ISSN: issn.print 2633-9137. ISSN: issn.print 2633-9137. http://journals.sagepub.com/doi/10.1177/26339137231162025.

[39] FISHER, Michael, Viviana MASCARDI, Kristin Yvonne ROZIER, Bernd-Holger SCHLINGLOFF, Michael WINIKOFF and Neil YORKE-SMITH. Towards a framework for certification of reliable autonomous systems. *Autonomous Agents and Multi-Agent Systems*. vol. 35 no. 1. Springer Science and Business Media LLC. 2020. DOI: DOI 10.1007/s10458-020-09487-2. ISSN: issn.print 1387-2532. ISSN: issn.electronic 1573-7454. https://link.springer.com/10.1007/s10458-020-09487-2.

[40] LI, X., L.-K. SOH and University of Nebraska–Lincoln, Computer Science and Engineering. *Applications of Decision and Utility Theory in Multi-Agent Systems*. 2004. https://digitalcommons.unl.edu/csetechreports/85/.

[41] ALBARRACIN, M., I. HIPÓLITO, S. ESSAFI TREMBLAY, J. G. FOX, G. RENÉ, K. FRISTON and M. J. D. RAMSTEAD. Designing explainable artificial intelligence with active inference: A framework for transparent introspection and decision-making. In. n.p.: 2023. https://arxiv.org/abs/2306.04025.

[42] LEONARD, Naomi Ehrich and Simon A LEVIN. Collective intelligence as a public good. *Collective Intelligence*. vol. 1 no. 1. SAGE Publications. 2022. DOI: DOI 10.1177/26339137221083293. ISSN: issn.print 2633-9137. ISSN: issn.print 2633-9137. https://journals.sagepub.com/doi/pdf/10.1177/26339137221083293.

[43] HERRMANN, Esther, Josep CALL, María Victoria HERNÀNDEZ-LLOREDA, Brian HARE and Michael TOMASELLO. Humans Have Evolved Specialized Skills of Social Cognition: The Cultural Intelligence Hypothesis. *Science*. vol. 317 no. 5843, pp. 1360–1366. American Association for the Advancement of Science (AAAS). 2007. DOI: DOI 10.1126/science.1146282. ISSN: issn.print 0036-8075. ISSN: issn.electronic 1095-9203. https://www.science.org/doi/10.1126/science.1146282.

[44] LEVY, P., R. BONONNO and Perseus Books. *Collective Intelligence: Mankind's Emerging World in Cyberspace*. n.p.: 1997.

[45] BERNERS-LEE, T. and Farrar, Straus and Giroux. *This is for everyone : the unfinished story of the World Wide Web*. n.p.: 2025.

[46] ZHANG, K., Z. YANG and T. BAŞAR. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. In. n.p.: 2021. https://arxiv.org/abs/1911.10635.

[47] MANI, Ankur, Iyad RAHWAN and Alex PENTLAND. Inducing Peer Pressure to Promote Cooperation. *Scientific Reports*. vol. 3 no. 1. Springer Science and Business Media LLC. 2013. DOI: DOI 10.1038/srep01735. ISSN: issn.electronic 2045-2322. https://www.nature.com/articles/srep01735.

[48] WOOLLEY, Anita Williams, Christopher F. CHABRIS, Alex PENTLAND, Nada HASHMI and Thomas W. MALONE. Evidence for a Collective Intelligence Factor in the Performance of Human Groups. *Science*. vol. 330 no. 6004, pp. 686–688. American Association for the Advancement of Science (AAAS).

2010. DOI: DOI 10.1126/science.1193147. ISSN: issn.print 0036-8075. ISSN: issn.electronic 1095-9203. https://www.science.org/doi/10.1126/science.1193147.

[49] Knowledge Graph. n.d. https://en.wikipedia.org/wiki/Knowledge_graph.

[50] Wikipedia. *Solid (web decentralization project)*. https://en.wikipedia.org/wiki/Solid_(web_decentralization_project).

[51] PuppyGraph. *Distributed Graph Database*. https://www.puppygraph.com/blog/distributed-graph-database.

Annex C
(informative)

Application Scenarios

C.1. Cultural Location Tourism

C.1.1. Overview

This annex is based on the Cultural Location Tourism application scenario from Spatial Web Protocol Architecture and Governance clause 5.3.5.

C.1.1.1. 5.3.5.1. A Virtual Tour

C.1.1.1.1. Scenario

Jane is interested in getting a virtual tour of the Smithsonian Institute's Natural History Museum (SI-NHM) through her laptop, in preparation for physically touring the museum. She goes to the SI-NHM's Tour Domain, looks at the map showing an exploded view of the museum, then clicks on the first exhibit she wants to see (the Gem Room), which then takes her to this room, where she can go from station to station (place to place).

At each exhibit, she can see photographs, 3D renderings or movies showing the content, along with a detailed synopsis (card) of the exhibit's history. She can continue "walking" through the exhibits, walk to the next room, or can search to find exhibits on topics that she's interested in (which she can also bookmark). She can continue in this manner until she leaves. Once she's done, she can ask the spatial web server to calculate the best tour that she can take to hit the exhibits of most interest to her. As Jane moves through the museum, she can add comments (annotations) on each place that she's in.

C.1.1.1.2. What Does This Test?

- Placement of agent within a domain
- Generation of a map
- Navigation within a domain
- Retrieval of Content Metadata
- Retrieval of Media
- Optimization of Paths

- Persistence of Domain Data
- Presentation through Devices (VR glasses or smart phones)

C.1.1.1.3. Requirements

- Each gallery within the museum can be modeled as a place.
- Each station within a gallery can also be modeled as a place within a holonic link to the associated gallery.
- Each station in a gallery has a thing (a kiosk) which displays information about the station's topic. The information can either be stored within a `<hsml:Content>` node or can be referenced via an external API (same element, but with an external href). Syntax TBD.
- Each place has a link to each of its neighbors. These links are contextual (the agent's permissions may determine whether a link is visible or active, for instance).
- Motion is tracked via historical Events that get persisted when a particular place is visited or activity is undertaken.
- The Agent in this case represents the actor (Jane) within the context of the domain.
- Annotations can be bound to either the historical log or the place.
- The museum domain has a home place that would correspond to the entryway into the museum, and each gallery would have a home place that would indicate the first kiosk "seen" by the agent when entering the agent.
- The default map function on the museum domain would show each gallery, while the default map function on each gallery domain would show the kiosks within that gallery.
- A map function can also pass a level of detail (LOD) parameter that would indication how many levels deep the map response whould discover. Thus, an LOD of 2 would show each gallery and each station in the gallery, along with all agents within the relevant domain at that level, depending upon agent access privileges.
- The data feeds for the kiosks are language sensitive. This means that the agent can set the language for output (or at least select from a list of available languages).

C.1.1.1.4. Observations

- The tour is one of the most prevalent paradigms of the spatial web. Almost every scenario involves either requesting a map, moving through that map via links, or interacting with a thing within that domain as part of that tour.
- In this respect, you can think of one of the roles of an agent is to act as the focus of intent within a given domain.
- This museum is a single agent instance of a SI-NHM museum domain template. It will stay active until either an end condition is met (even if that is simply terminating the instance). As such domains are kind of like virtual machines — they can be paused to retain their state, or they can be deleted when no longer needed.

C.1.1.2. 5.3.5.2. An XR Experience

This is a similar scenario to 5.3.5.1, with the following differences:

C.1.1.2.1. Scenario

Jane goes to the museum, with her trusty VR glasses perched on her nose, synched to the museum's instance. As she enters the front entrance, the spatial web sensors correlate the position (via GPS or other positional sensor tech) to a given place, as defined by its H3 tile(s). If she is within the relevant tile, the spatial web glasses indicate that there is additional relevant metadata in the place, which can be activated to show the relevant media (videos, perhaps).

Jane summons up a map to see what is available, and starts to move, and as she leaves a given region and moves into another, the old billboard goes away to be replaced by a new map or billboard icon (possibly both) that can then be expanded.

At some point, she gets hungry, and wants to know how to get to the cafeteria. She asks for the cafeteria and either an arrow will pop up in front of her indicating the direction to follow or a map will appear with a path to that area.

As she eats, Jane decides that she wants to go from a visual to a light visual audio display. From then on, instead of billboards, a running commentary comes through her earbuds, with directions, recommendations, and warnings being spoken rather than displayed as imagery or text video.

At the end of her tour, she can ask for an itinerary and transcription, which reproduces critical information that can then be transmitted as compressed HSML and saved for later review.

C.1.1.2.2. What Does This Test?

- Everything in 5.3.5.1
- Geopositioning with external environment synching.
- Search
- Modalities of perception
- Path negotiation and optimization
- History and Transcription

C.1.1.2.3. Requirements

- Positioning Sensors
- Search Capabilities
- Modality Control
- History of Events

C.1.1.2.4. Observations

- Sensors within the client device can provide mapping to a spatial position, which can then be transformed into a tile position, correlating with a given place within

the model. Note that if somehow the actor ends up outside of defined tiles, then an algorithm can be used to determine the closest place within the domain, which can be correlated to suggest directions.

- Search is a query against places, things and agents respectively that will suggest candidates that most closely match the query parameters. This will generally be displayed as a list, and can be filtered by type. Search is sensitive to agent permissions.
- Modality may be a function of the client or the node, but will typically work by transforming a map in RDF into some other form (an image, a diagram, audio, a movie, 3d environment, etc.) that can be consequently rendered by the client. The exact mechanism for performing this is TBD.
- As an agent moves through a domain, that agent creates a history correlated to the agent and the domain that can be persisted, then transformed into various forms, such as a transcript or summary. The exact mechanism for creating a history of events is still TBD.
- Note here the symmetry between actor (Jane) and her agent (Jane_Agent). Jane moves through the physical world, with sensors indicating a geospatial position. Jane_Agent moves through an abstract conceptual world from Place to Place, correlating with the physical to a certain degree. In effect, Jane_Agent is the digital twin of Jane within the domain of the museum.
- The terms *billboard, screen, and kiosk are used to describe Things in the virtual world. A billboard can be thought of as a read-only interface or display, and is usually fully visible when an agent moves within the Place where a billboard is resident. It can have any representation (it is not limited to being a billboard in the physical sense) but generally provides external information in various formats that the user has no immediate control over.*
- A *screen* is a *billboard that specifically displays a dynamic map of another domain, where a map is a representation (an image, 3d rendering, video, text description, RDF, etc.) of a domain. It could be visualized as a screen showing the projection taken by a camera of another area, a glass plane showing what's in the next gallery,an aural representation where the voices from somewhere else can be heard as if through a mic, a structured HSML representation of the domain, and so forth. The map is the representation of a domain, the screen is the presentation (or medium) of that representation in the current domain of the agent.*
- Note also that a *map is in effect the view as seen by a separate agent within a remote domain. The remote agent here is acting in the role of a camera. A kiosk is a Thing that combines a screen with a control mechanism for that remote agent. For instance, an agent (a drone operator, for instance), uses another agent (the drone) in a different domain to "see" that remote domain from the perspective of the drone. The drone operator agent interacts with the remote drone agent via a virtual kiosk.*
- These agent chains are very common in most video games, particularly when dealing with IoT devices (especially cameras). A camera is a specialized form of *sensor, a device that creates a representation (map) from the perspective of a given agent or thing. Screens are linked to agents, and a screen can consequently target different agents to see different perspectives of the domain.*

C.1.1.2.5. Maps and Screens

FIGURE C.1

```
---
config:
   layout: elk
---
flowchart TD
    remoteAgent -->|uses| sensor --> |to create| map --> |
of| remoteDomain
    localAgent --> |uses| screen -->|to display| map
    screen -->|linked to| remoteAgent
    remoteDomain --> |as seen by| remoteAgent
```

C.1.1.3. 5.3.5.3. Multi-Agent XR

This builds on 5.3.5.2, with the following differences:

C.1.1.3.1. Scenario

Jane joins a tour group of other actors (with no distinction about whether those actors are human or AI based). They are led by a spatial web mediated tour guide, and each agent can ask questions of other agents or the tour guide.

The tour guide will periodically ask questions of the various tour members. If they answer the question correctly, they get a special token which they can accumulate. At the end of the tour, each agent can exchange tokens as discounts on the price of items in the gift shop.

This scenario is like 5.3.5.1 in that it is mediated over the spatial web client, rather than in person, but could be supported IRL as well.

C.1.1.3.2. What Does This Test?

- Registering Agents and forming Teams
- Direct Communication between agents
- Granting or Exchanging Certs
- Agent/Actor Interactions
- E-Commerce Fundamentals

C.1.1.3.3. Observations

- When a domain is created from a domain template, the domain goes through a provisioning phase.In this phase, one or more autonomous agents wait until a minimum condition is met (here, both a minimum number of people and a set period of time). This set of agents becomes known as a *Team*. Teams can be thought of as neighborhoods of agents. A given agent may also be part of more than one team.

- From a design standpoint, it is often preferable to talk about a team with only one member, also known as a singleton team. For instance, in Chess, you effectively have two singleton teams — a white piece team and a black piece team.
- This notion of teams is an important one, because team members often have a much higher need to communicate with one another, and benefit far more from that interaction. Moreover, teams have identities (and histories) that individual agents don't, and frequently have needs for permissions (credentials) that two random people don't.
- In this use case, the tour group is a team. Each team member registers with the team (here, they would pay the price for the guided tour), and they share in a communication channel that is consequently privileged. The exact nature of that chennel (point-to-point, broadcast, narrowcast, etc.) would be spelled out in the *contract that the actors agree to through their agent proxies when they register with the team. This also suggests that registration is the process of an actor (through the agent) accepting a contract issued by the domain that encapsulates these policies.*
- Please note that registration is a domain scope activity. When the registration is complete, the *agent then belongs to a team within the domain.*

FIGURE C.2

```
---
config:
    layout: elk
---
graph TD
    actor -->|uses| agent -->|to agree to| contract -->|
with| team -->|within| domain
    agent -->|becomes member of| team
    agent -->|is within| domain
```

- Channel communication: A *communication channnel* is a *channel specifically for text communication between agents and/or teams. It passes an HSML message (structure TBD) from the sender to the recipient using the following workflow:*

FIGURE C.3

```
---
config:
    layout: elk
---
graph
    actor1 -->|writes message to| agent1
    agent1 -->|sends message to| domain
    domain -->|"caches message in"| messageCache
    messageCache -->|"sends message to"| agent2
```

```
        agent2 -->|"writes message to"| actor2
```

- The *MessageCache* is a stack within the UDG.d for passing messages between entities. It works at the domain level rather than direct point-to-point primarily because messages will still need to be logged as part of the history stream and because any form of broadcast ultimately will need to be transmitted to some or all of the participants in a domain, which can best be done through a centralization mechanism.
- There is a question concerning whether internal communication and messaging channels are part of the external hstp *channels* architecture or are different. Certainly, *actor1* and *actor2* communication with their respective agents are handled via the former. This is one of those areas that still requires a certain amount of discussion.
- *E-Commerce, Contracts, and Tokens*. The assumption being made in the architecture is that an *e-commerce layer will likely be a later module that lays on top of the UDG, and more than likely will be mediated via a services layer with external financial networks, in much the same way that e-commerce systems in the modern web are generally not considered a core part of HTML/HTTP. The one caveat on this is that, because of the use of SWIDs to manage credentialling, verification, validation, and authentication, the ecommerce architecture will likely utilize DID-based encryption vs. the HTTPS secure architecture used for the web.*
- One additional caveat is that in general, assets (specializations of THINGs) will be represented as encrypted key entities that can consequently be transferred to the wallet of the client controlling the respective agent of a given actor. For instance, in the gift shop example given in this use case, Jane can be awarded a magic shell (a *token) from the tour guide for answering a question correctly. The exact representation of the token will obvious vary from domain to domain (as will it's value), but it clearly represents an asset that can be assessed within some e-commerce system. A token in this case can be thought of as a specific store of value within the spatial web. The actor authorizes the agent to spend tokens within the domain in order to fulfill the terms of a contract, and correspondingly retrieves tokens when the conditions for fulfilling a clause of the contract have happened. It is still TBD whether or not tokens issues within one domain are fungible within other domains.*

C.1.1.4. 5.3.5.4. Tracking Movement

This builds on 5.3.5.2, with the following differences:

C.1.1.4.1. Scenario

Jane wants to be able to see where the other members of the tour (including her family) are, and send them messages to meet at a particular place at a certain time.

C.1.1.4.2. What Does This Test?

- Hyperspace
- Positioning within Places

- Avatars and Map Representations

C.1.1.4.3. Observations

- The dominant paradigm within the World Wide Web since its inception was based on the principle of publishing, and can best be articulated as: *"How do I find and access published content?* This in turn was related to *"How do I publish content?"*
- The dominant paradigm for the Spatial Web, however, is different. It can be expressed as: "*Where are the things in the world that I interact with?*" with the correlative question, "*How do things in my world publish where things are and what they do*?" While similar in scope, this is different in terms of the overall mission of this technology.
- The Spatial Web does not replace the World Wide Web. Rather, it provides another layer to the *noosphere* or knowledge sphere, providing not only the context of philosophical thought but also of epistemological thought.
- The spatial web has three different layers that ask the question "Where?":
- *Domain or application layer.*
- *Place or conceptual space layer.*
- *Location, or positional space layer.*
- The *domain layer* is an existential context layer. By itself, a domain does not necessarily specify where things are, but instead, it indicates what process or system things are apart of. This systemic view is purely abstract, though it may have implicit hierarchies that arise because such hierarchies make it easier to compress process nested subroutines.
- The *place layer* is a conceptual space layer, and generally identifies a partition of a "hyperspace" into discrete, interconnected nodes within a lattice of links. A hyperspace can be thought of as the set of all relevant places within the broader domain, and will vary from domain to domain in terms of breakdown and structure. Places by themselves have relationships, but do not necessarily have the notion of a metric.
- The *location layer* contains specific metrics and the notion of distance. The specific mechanisms for describing that distance will vary from place to place. For instance, one *place* might be a particular gallery within a museum, but within that gallery, there may be an ability to indicate location relative to the defined layer (possibly using some kind of a global coordinate system, or at a minimum level a local coordinate system that is common to that place). If you wanted to specify, for instance, that you are in a hexagon within a set of hexagons that identify the extent of that place as an index, then the location would be a single index value.
- A place will always have, at a minimum, one location — in the case where there are no effective degrees of freedom, this becomes the implicit location. As you increase the number of degrees of freedom within that place, you can better specify location if it is necessary.
- A place does have a location for mapping purposes, but it is defined as a bounding box within a unit hypercube (something called an *object coordinate system*), with the assumption that the domain represents the the maximal extent of this

cube. This hypercube is then passed through a defined *projection filter* (typically, but not always, a tensor) to create a visual representation of the domain in the target mapping dimensions. In most cases, this will be a two dimensional planar representation, even if the hypercube itself is of higher dimension and curved.

- It is important to recognize that the hyperspace envelope of places does not usually completely tile (cover) the object coordinate system. The set of places is contextual and topological (a graph) and is internally connected by links, *holes* (areas that are in the map but not actually within the model) are inevitable. Another way of putting this is that the map shows the relevant areas of the models, but anything that is not relevant (negative space) is simply an undefined region in the map. This is another way of stating the famous dictum "*The map is not the territory.*".
- An *avatar* is a representation of an *entity on a map within a given medium (or content-type). It is typically represented in object coordinates, depending on the place's positional system and the medium in question. An entity, including an agent, may have multiple different avatars, with the best one for the mapping context being chosen prior to rendering. More on Avatars TBD.*
- Movement itself is managed by the udg.d daemon, which refreshes the state of the system at regular, frequent intervals, then renders this movement via hstp.d requests to the mapping service.
- The *UDG.d daemon* also manages communication between the UDG and the HSTP layers. For instance, if a domain template has been previously defined (which will be covered in the next phase), then the HSTP will pass messages to the udg.d daemon to initiate a domain instantiation. The UDG.d itself is responsible for the creation of that instance, but it also communicates with HSTP when it has successfully completed the instantiation. Most of the operational logic that is initiated by the *hstp.d is actually performed by the udg.d, then transmitted back to other nodes via the hstp.d messaging system.*

C.1.1.5. 5.3.5.5. Museum Discovery

This supports other 5.3.5.x use cases.

C.1.1.5.1. Scenario

Jane wants to find other museums in the Smithsonian Institute complex to virtually visit, utilizing the same agent avatar that she had previously, including retaining knowledge and assets.

C.1.1.5.2. What Does This Test?

- Cross Domain Discovery and Linking
- Directories
- Spatial Web Domain Registries
- Agent Persistence

C.1.1.5.3. Observations

- *Cross Domain Linking*. There are multiple layers of linking that exist within the spatial web. One of these is *cross-domain linking*. Such a link moves an agent from one domain to another, rather than simply from one place to another. This is roughly analogous to an external link in HTML that takes you outside of the document, albeit one that has more complexity.
- *Intranode Domain Agent Linking*. Unlike HTML links, cross domain links are stateful —you are in essence transferring an agent from one graph to another, potentially outside of the Spatial Web Node itself. If the *originating domain* (which has the link to the agent) is within the same node, this becomes a fairly simple matter of delinking the agent from one node and relinking it to the *destination domain*.
- *Internode Domain Agent Linking*. If the destination domain, on the other hand, is outside of the spatial web node, then the agent on on the originating domain must in effect be *frozen* or deactivated, while the agent's information and assets are transferred to the *destination spatial web node*. Additionally, a forwarding address is added to the frozen agent on the initial node to the active agenton the destination nnode. This makes it possible to search the evolution of agents across nodes. If an agent returns to the originating node, it's associated asset metadata is *appended to* the previous agent, allowing the agent to learn information over time.
- *Affiliation Networks*. Note additionally that transferring of agents can only occur if the domains have contracts allowing the transfer of agents, which in general means that they have created an affiliated network. This would be like multiple museums each agreeing to honor the contracts of other museums in the network. The exact mechanisms for doing so are currently to be determined.
- *Directories and Landing Domains*. Each Spatial Web Node has a specific landing domain for that node. This identifies the domain templates that are supported on that node, and for each template, the active and completed domains for those templates. This landing Place is generated dynamically, and can be thought of as being analogous to a train station that allows agents to go to a particular domain. This is not a registry per se, but more akin to a directory.
- *Domain Registries*. A *Domain Registry* is a way for organizations to register the _*domain templates* that are publicly available. Domains themselve may be very effervescent (though they can also be long lived), but domain templates are generally stable. In the museum scenario, for instance, there may be any number of active SI-NHM domains active at any given moment, with individual agents or teams of agents interacting with a given domain, but the SI-NHM domain template that informs these domains will remain fixed as a stable point of reference.
- The domain registry, consequently, can tell you which spatial web nodes contain the relevant domain templates, allowing you to search these nodes to see if the domains (the games, simulations, IoT environments, and scenarios) are of interest. This domain registry is managed by (or delegated from) the *Spatial Web Registration Authority. The mechanics of registration are being worked out.*
- *Associated Metadata*. There is a core taxonomy being worked out for helping search and discovery for the domain registry. This is associated with both a *Place* registry for registering places (distinct from domain, though interrelated) and for conceptual registries for identifying *Topics*, where a topic in general is used primarily

for descriptive metadata (find me all domains that focus on global warming, for instance). There will similarly be a such registries for *Personages* and *Organizations*.

C.1.1.6. Design Considerations on 5.3.5

- The model presented here within the UDG looks at the environment of a Spatial Web Node as being a collection of applications built around domains, supported by secondary components, with this particular application being an example of what the author would consider an *Exploration pattern.*
- *Exploration Pattern.* This pattern works on the assumption that one or more agents, acting as proxies for various external actors, are navigating a space (geophysical, conceptual, organizational, etc.), retrieving information, interacting with other agents and things within the system, and gaining respective keys (tokens) that can be used both for "unlocking" specific places within the system and for exchanging as stores of value inside and outside the system.
- *Maps and Properties.* What is most significant here is that the spatial web uses a knowledge graph as its store (and for now is assumed to query and update through a KG layer) but that it's not really a graph in the traditional sense. When you want to *query the property of a given thing or agent in the system, what you are doing is retrieving a map of the thing within one or more domains, expressed in RDF (as JSON-LD, most likely), that will retrieve a representation of the object containing just that property, more than likely as a time-series unless you specify a temporal constraint. For instance, you can get a map of the museum domain showing each of the exhibits, or only those exhibits focused on animals, or the exhibits within a gallery, and so forth.*
- *Maps as Data Structures.* Maps are data structures first — they can be rendered into other forms, but every map is at its core a query against the UDG graph to retrieve representations of entities within the context of a given domain. It should also be noted that a map can be tabularized (this is what the SELECT statement in both SQL and SPARQL do) to provide a slice of this information in tabular form, but even so, the underlying query will be retrieving the subgraph containing relevant entities before applying this transformation to a table.

C.1.1.7. Suggested Use Cases

- Adding a new exhibit
- Adding a new gallery
- Creating a Museum Domain Template
- Moving an Agent From One Domain to Another

Document contributors

George Percivall (lead editor), Kurt Cagle, Christine Perey, Reese Plews

Spatial Web Foundation leadership

Gabriel René	Executive Director / Founder
Dan Mapes	Managing Director / Founder
Michael Wadden	Managing Director / Outreach
Capm Petersen	Director of Innovation & Strategy
Bastiaan den Braber	Director of Operations
George Percivall	Distinguished Engineering Fellow
Dr. Jacqueline Hynes	Research Engineer
Dan Richardson	Director of Market Analysis
Dr. Sarah Grace Manski	Senior Ethics Advisor

Comments about the Spatial Web and this document can be sent to the Spatial Web Foundation at info@spatialwebfoundation.org