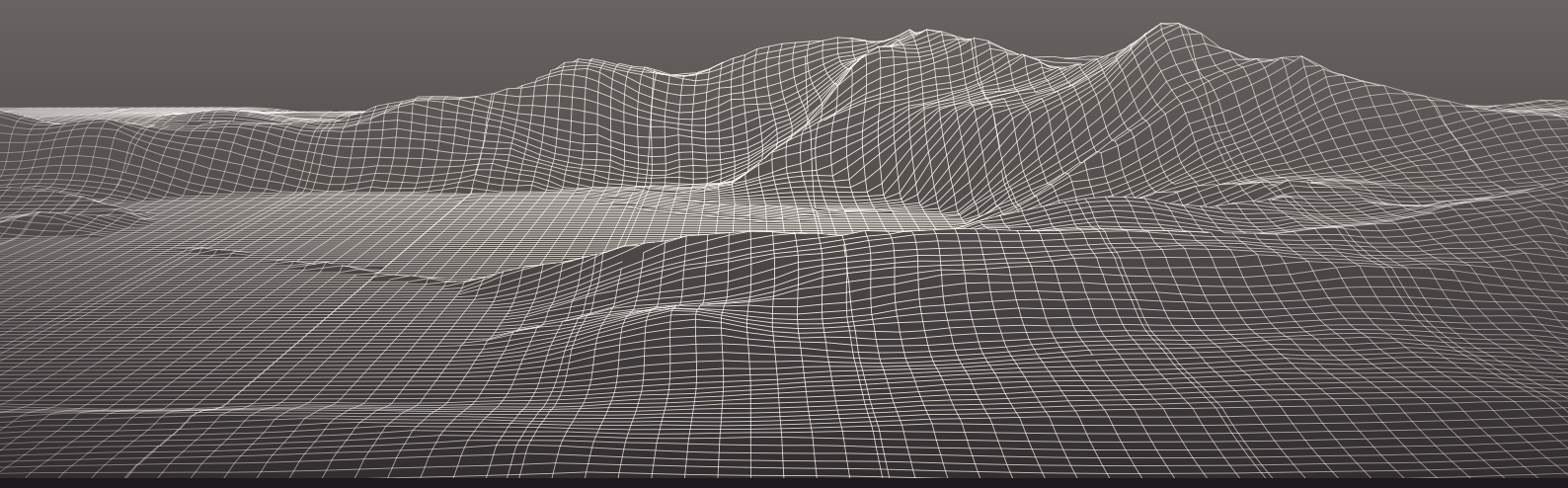
SPATIAL WEB
FOUNDATION

SWF STD-3:2025 (0.1.0)

The Spatial Web

Hyperspace Transaction Protocol (HSTP) Implementation Specification

September 26, 2025



Contents

List of tables

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

E-mail: info@spatialwebfoundation.org

Abstract

This document is the Implementation Specification for the Hyperspace Transaction Protocol (HSTP), produced by the Spatial Web Foundation. It provides the detailed technical requirements developers need to build and test HSTP-conformant components. It maps the system requirements defined in the IEEE P2874 Spatial Web Protocol, Architecture and Governance reference model into concrete implementation clauses and serves as the basis for conformance testing.

Introduction

This specification is one of a series produced by the Spatial Web Foundation to realize the IEEE P2874 Spatial Web system design. IEEE P2874 defines the reference model for the Spatial Web's architecture, governance and core protocols. Here, we focus on HSTP—its structure, message formats, interaction patterns and compliance targets—to enable practical implementations.

The Spatial Web Foundation develops these lightweight implementation specifications in collaboration with reference specification implementers and working groups, iterating between draft text, code examples and feedback to ensure clarity and adoption.

1. Scope

This Implementation Specification defines the Hyperspace Transaction Protocol (HSTP) as the core transactional layer of the Spatial Web. It specifies the following:

1. Message Envelope: An RDF graph, primarily serialized as a JSON-LD document, containing HSML payloads
2. HSTP OPERATIONs: list of OPERATIONs that HSTP nodes may support
3. Protocol Bindings & Encodings: Normative transports and formats
4. Core data structure
   - Core spatial web entities
   - Representation within the message payload
5. Conformance Criteria: Testable compliance targets are grouped in Annex A; each target is traced back to IEEE P2874 clauses in Annex B.

Conversely, the following items are out of scope:

- Higher-level application protocols that utilize HSTP
- Specific implementation technologies or programming languages
- Hardware or infrastructure deployment requirements
- Business logic or application-specific transaction types
- Governance and policy mechanisms (covered in IEEE P2874)
- HSML encoding rules (see HSML Implementation Spec)
- UDG query APIs (see UDG Implementation Spec)
- SWID document formats (see SWID Implementation Spec)

1.1. Word Usage

This specification reuses the generic normative keywords from IEEE P2874 Section 1.5 (see Annex C) and adds these HSTP-specific conventions:

- HSTP OPERATION: ALL_CAPS verb naming for protocol actions (e.g. LOOKUP, COMMIT).
- Envelope Fields: lowerCamelCase names (hstpVersion, messageId, etc.).
- Enumerations: UPPER_SNAKE_CASE for status and error codes (SUCCESS, INVALID_SIGNATURE, etc.).
- Ontology Terms: TitleCase for high-level concepts (Domain, Agent, Activity, Contract, etc.).
- Any other capitalization or wording in examples is non-normative.

1.2. Motivating Scenarios and Need for HSTP

This specification provides the detailed technical requirements for building and testing HSTP-conformant components, targeting engineers and architects implementing Spatial Web systems, mapping system requirements from the IEEE P2874, mapping system requirements from the IEEE P2874 Spatial Web Protocol, Architecture and Governance reference model into concrete implementation clauses.

The Hyperspace Transaction Protocol (HSTP) is the core transactional layer of the Spatial Web. Its foundational purpose is to underwrite the automated contracting necessary for building a coherent, decentralized, secure, and privacy-respecting Spatial Web. It is crucial to understand that HSTP is an application-layer protocol and a transactional protocol, designed to facilitate the execution of functionality and the sharing of structured data through defined operations and transactions, rather than being a general data transfer protocol. This specification provides the detailed technical requirements for building and testing HSTP-conformant components, mapping system requirements from the IEEE P2874 Spatial Web Protocol, Architecture and Governance reference model into concrete implementation clauses.

1.3. The Foundational Need for HSTP

The need for HSTP stems from the complex requirements of the Spatial Web for coherent, secure, and interoperable interactions that go beyond simple data exchange. The following lists several examples of these needs:

- Enabling Automated Transactions and Functionality Execution: HSTP is fundamental for automated contracting within a decentralized Spatial Web. It operates as a request/response protocol, enabling HSTP-compliant systems to execute specific functionalities and exchange relevant data. Its transactional focus is evident in OPERATIONs such as REGISTER_DOMAIN for creating domains, EXECUTE_ACTIVITY for requesting activity execution, and CREATE_ENTITY, UPDATE_ENTITY, and DELETE_ENTITY for managing entities in the Universal Domain Graph (UDG). HSTP

messages encapsulate Hyperspace Modelling Language (HSML) Activity Schemas, forming the structural basis for protocol commands and allowing for complex interaction flows like contract negotiation.

- Operations and Semantics at the Operational Layer: HSTP addresses the critical need for standardized and semantically consistent communication among diverse systems and endpoint devices. By defining a common semantic layer, HSTP ensures that the meaning of exchanged information and executed functionalities remains consistent, regardless of the underlying network technology. This enables independent implementations to interoperate effectively.
- Enhancing Security and Trust Models: A core need for HSTP is to establish robust security and trust within the Spatial Web. It incorporates mechanisms for authentication, authorization, and encryption to safeguard interactions. HSTP explicitly implements a zero-trust security model, requiring all users and entities to be authenticated, authorized, and continuously validated. This is enforced by mandating that HSTP message payloads be encrypted, with decryption access gated behind a claim in a verifiable credential. OPERATIONs like ENCRYPT_PAYLOAD and DECRYPT_PAYLOAD are integral to this security framework.
- Structuring and Managing Complexity in Projects (Technical Specification Perspective): The development of the HSTP technical specification itself is a crucial process. Writing such a document forces engineers to thoroughly examine problems before implementation, helping to identify overlooked aspects of a solution. It provides a comprehensive view of the proposed solution, serving as vital documentation during both implementation and post-project review. For teams, it offers an efficient way to communicate design ideas, fostering collaborative problem-solving, shared ownership, and responsibility, which limits complications from overlapping work and aids in onboarding new members. This investment ultimately leads to a superior product by ensuring team alignment, faster progression of large projects, effective management of complexity, and the prioritization of impactful components.
- Interacting with Core Spatial Web Entities: HSTP's OPERATIONs are deeply integrated with the fundamental Spatial Web entities defined by the Hyperspace Modelling Language (HSML) and managed within the Universal Domain Graph (UDG). It defines OPERATIONs for Create, Read, Update, and Delete (CRUD) actions on HSML entities and for executing activities. The payload field in HSTP messages is specifically designed to carry HSML Entities or Activity Schemas, enabling the protocol to transport the rich, structured data models of HSML. This tight integration ensures HSTP facilitates direct interaction with the dynamic data models and relationships (e.g., hierarchical, heterarchical, nested) established by HSML and managed within the UDG.
- Protocol Bindings and Encodings: HSTP is designed to operate over various normative transport protocols (e.g., HTTP/2, QUIC, MQTT) while using multiple data serialization formats (e.g., JSON-LD, Turtle, OData, GraphQL) to ensure broad interoperability.

1.4. Motivating Scenarios: HSTP in Action

The transactional nature of HSTP is best illustrated through its application in various Spatial Web scenarios, where it orchestrates complex interactions and state changes. HSTP implements use cases from the IEEE P2874 standard. Outlines of several scenarios are shown in the following:

- Urban Digital Twin / Smart City: HSTP enables the foundational transactions for managing a smart city. This includes REGISTER_DOMAIN for city government and energy utilities, ISSUE_CREDENTIAL to authorize employees, CREATE_CHILD_DOMAIN for adding energy plans or digital twin models to the Spatial Web, and UPDATE_DOMAIN status for changes based on usage. It also facilitates MONITOR_CHANNEL for discussions on energy goals and progress.
- Global Supply Chain: In logistics, HSTP facilitates transactions such as CREATE_CHILD_DOMAIN to manifest goods, CREATE_ACTIVITY to initiate shipments and plan routes, and TRANSFER_DOMAIN when goods are received at their destination. It also enables MONITOR_CHANNEL for real-time updates on shipments, progress, and events, allowing for re-planning and announcements via the channel.
- Warehouse Robot Operations: HSTP underpins automated warehouse processes, allowing a robot agent to negotiate ACTIVITY CONTRACT for tasks like pick/delivery. It handles the update DOMAIN status as physical objects are transferred and picked.
- Entertainment XR: For immersive experiences, HSTP enables ISSUE_CREDENTIAL for game privileges, UPDATE_DOMAIN status to track a player's location in a virtual space, and CREATE_ACTIVITY for game challenges. It also manages TRANSFER_DOMAIN when virtual assets are exchanged between players.
- Digital Earth: Greenhouse Gas Monitoring: HSTP supports environmental management by enabling CREATE_DOMAIN for GHG goals and monitoring networks. It facilitates CREATE_ACTIVITY for governments to define guidance on GHG reduction channels and MONITOR_CHANNEL for communities to discuss actions, with AI mediation nodes recommending agreements as contracts.

These examples demonstrate how HSTP, as a transactional protocol, orchestrates the dynamic, state-changing interactions essential to the Spatial Web's functionality across diverse domains.

1.5. Differentiation and Unique Value Proposition

The unique value proposition of HSTP lies in its specific design as a transactional protocol for the Spatial Web, distinguishing it from general data transfer protocols. Several dimensions of differentiation are shown in the following:

- Transactional Core: Unlike basic data transfer protocols, HSTP is explicitly designed as the "core transactional layer" that underwrites "automated contracting". It focuses on defining the actions systems perform and the state changes they effect, rather than just raw data movement.

- Semantic Interoperability Across Transports: HSTP provides a "common semantic layer" that ensures consistent meaning of information and executed functionalities regardless of the underlying network technology (e.g., HTTP/2, QUIC, MQTT, GraphQL, Kafka, NATS, MCP, A2A). This contrasts with protocols that merely move bits without providing inherent semantic understanding across disparate systems.
- Zero-Trust Security by Design: HSTP mandates encryption of payloads and access control via verifiable credentials, implementing a "zero-trust security model". This baked-in security is integral to its transactional integrity, rather than being an add-on.
- Direct Interaction with Spatial Web Entities: HSTP's OPERATIONs are specifically tailored to interact with HSML Entities (e.g., Activities, Domains) and the UDG, enabling direct CRUD OPERATIONs and execution of complex behaviors within the Spatial Web's structured data model.
- Support for Complex Agent Interactions: HSTP provides protocols for agents with "diverse intelligences to communicate and interact effectively". It supports interactions across varying levels of contextual and syntactic complexity, facilitating machine learning and cognitive computing OPERATIONs. This goes beyond simple message passing to enable sophisticated, goal-oriented agent behaviors.

1.6. Scalability and Performance Justification

HSTP is designed with the inherent scalability and performance requirements of a global, distributed Spatial Web in mind. Several examples of these considerations are shown in the following:

- Support for Diverse Network Conditions: HSTP is engineered to "operate with varied communication network performance," accommodating both high latency/low connectivity and high-bandwidth scenarios (hundreds of gigabits to several terabits per second).
- Scalability of Agent Interactions: The protocol is built to "support scalability, enabling increased AGENT interactions without compromising performance," which is crucial for large-scale operations.
- Resilience to Faults: HSTP is designed to be "resilient to faults," ensuring that the network of Spatial Web nodes experiences only transient loss of function if a single node becomes non-operational.
- Distributed Architecture Compatibility: It is capable of working seamlessly in both "centralized or distributed computing environments", acknowledging the heterogeneous nature of the Spatial Web.
- Flexible Transport Negotiation: HSTP messages can include information that allows communicating systems to "negotiate their preferred or required transport method on a per HSML activity basis". This dynamic negotiation enables optimal performance tailored to the specific activity and available network conditions.
- Optimized Routing and Discovery: HSTP messages include the sender's position in the UDG (SWID, known neighbors, spatial hints) and information to locate networked addresses. This provides the Distributed UDG System with necessary

information for efficient routing and discovery, reducing latency in locating entities and services.

1.7. Adoption Strategy and Pathway

The adoption of HSTP is strategically planned through collaboration with established standards bodies, iterative development, and a focus on interoperability. Several steps and factors on this path are outlined in the following:

- IEEE Standards Collaboration: HSTP is part of a series of "lightweight implementation specifications" produced by the Spatial Web Foundation in collaboration with IEEE SA, to realize the IEEE P2874 Spatial Web system design. This collaboration follows rigorous requirements to become a globally adopted IEEE standard.
- Iterative Development and Feedback: The development process emphasizes iterating between draft text, code examples, and feedback from working groups to ensure clarity and adoption. The current stage is a Preliminary Design Review (PDR), where the goal is to produce actionable initial drafts sufficient to initiate collaboration and gather essential feedback from various teams (HSML, UDG, Agent). This iterative approach focuses on clarity over completeness in early drafts.
- Conformance and Interoperability: The HSTP specification defines "Conformance Criteria" with "Testable compliance targets". This is critical for enabling independent implementations to be interoperable.
- Reference Implementations: Reference implementations are developed concurrently with the specification and test suite. These are vital for discovering errors or ambiguities in the specification, validating the correct functioning of test suites, serving as a "Gold Standard" for other implementations, and clarifying the intent of the specification where conformance tests are insufficient.
- Open Standards Philosophy: HSTP promotes interoperability between diverse ecosystems by adhering to Internet standards and allowing the use of multiple common Internet payload formats. This open approach fosters a balanced ecosystem of innovation and competition.
- Profile Definition: HSTP will define specific profiles for common protocols like HTTP, MQTT, GraphQL, and other protocols, further aiding adoption by providing clear guidelines for integration within existing ecosystems.

2. 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.

ISO 8601-1:2019, International Organization for Standardization. *Date and time — Representations for information interchange — Part 1: Basic rules*. First edition. 2019. Geneva. https://www.iso.org/standard/70907.html.

ISO/IEC 20802-1, International Organization for Standardization and International Electrotechnical Commission. *Information technology — Open data protocol (OData) v4.0 — Part 1: Core*. First edition. Geneva. https://www.iso.org/standard/69208.html.

ISO/IEC 20802-2, International Organization for Standardization and International Electrotechnical Commission. *Information technology — Open data protocol (OData) v4.0 — Part 2: OData JSON Format*. First edition. Geneva. https://www.iso.org/standard/69209.html.

IETF RFC 4122, P. LEACH, M. MEALLING and R. SALZ. *A Universally Unique IDentifier (UUID) URN Namespace*. 2005. RFC Publisher. https://www.rfc-editor.org/info/rfc4122.

IETF RFC 9000, J. IYENGAR and M. THOMSON (eds.). *QUIC: A UDP-Based Multiplexed and Secure Transport*. 2021. RFC Publisher. https://www.rfc-editor.org/info/rfc9000.

IETF RFC 9113, M. THOMSON and C. BENFIELD (eds.). *HTTP/2*. 2022. RFC Publisher. https://www.rfc-editor.org/info/rfc9113.

OASIS mqtt-v5.0, COPPEN, Richard, Andrew BANKS, Ed BRIGGS, Ken BORGENDALE and Rahul GUPTA (Chair). *MQTT Version 5.0*. 2019. OASIS. http://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html.

[1] The Linux Foundation. *Aries Agent to Agent Communication Protocol*. https://a2a-protocol.org/latest/.

[2] The Apache Software Foundation. *Kafka Protocol Guide, 4.1, Apache Kafka Documentation*. https://kafka.apache.org/documentation/.

[3] Joint Development Foundation Projects, LLC. *The GraphQL Specification Project*. 2021. https://spec.graphql.org.

[4] Cloud Native Computing Foundation. *NATS Server Protocol Specification*. https://docs.nats.io/nats-protocol/nats-protocol.

[5] Tom Preston-Werner. *Semantic Versioning Specification (SemVer) 2.0.0*. https://semver.org.

HSML Specification, Spatial Web Foundation. *Hyperspace Modelling Language (HSML) Specification*. 2025. HSML Specification.

UDG Specification, Spatial Web Foundation. *Universal Domain Graph (UDG) Specification*. 2025. UDG Specification.

SWID Specification, Spatial Web Foundation. *Spatial Web Identifier (SWID) Specification*. 2025. SWID Specification.

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

W3C json-ld11, World Wide Web Consortium. *JSON-LD 1.1*. https://www.w3.org/TR/json-ld11/.

W3C shacl, World Wide Web Consortium. *Shapes Constraint Language (SHACL)*. https://www.w3.org/TR/shacl/.

W3C sparql11-entailment, World Wide Web Consortium. *SPARQL 1.1 Entailment Regimes*. https://www.w3.org/TR/sparql11-entailment/.

W3C sparql11-federated-query, World Wide Web Consortium. *SPARQL 1.1 Federated Query*. https://www.w3.org/TR/sparql11-federated-query/.

W3C sparql11-overview, World Wide Web Consortium. *SPARQL 1.1 Overview*. https://www.w3.org/TR/sparql11-overview/.

W3C sparql11-protocol, World Wide Web Consortium. *SPARQL 1.1 Protocol*. https://www.w3.org/TR/sparql11-protocol/.

W3C sparql11-query, World Wide Web Consortium. *SPARQL 1.1 Query Language*. https://www.w3.org/TR/sparql11-query/.

W3C sparql11-service-description, World Wide Web Consortium. *SPARQL 1.1 Service Description*. https://www.w3.org/TR/sparql11-service-description/.

W3C sparql11-update, World Wide Web Consortium. *SPARQL 1.1 Update*. https://www.w3.org/TR/sparql11-update/.

3. Terms, definitions and abbreviated terms

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

3.1. Terms and definitions

Only the most HSTP-specific terms, or terms fundamental to the core message structure and payload content, are defined here.

3.1.1. Activity PREFERRED

A temporally-extended process defined by a set of changes an Agent can effect.

Note 1 to entry: An Activity Schema is a template with conditions and variables.

Note 2 to entry: An Activity Instance is the performance of a schema. Activities are central to entity-to-entity execution and state change and can be composed of other Activities.

Note 3 to entry: In HSTP, an ActivityType or Activity Schema is transported as part of an operation.

3.1.2. Addresses PREFERRED

A catalog of a system's publicly listable networked addresses.

3.1.3. Agent PREFERRED

An Entity that senses and responds to its environment, maintains a model of its environment, and performs Activities to achieve a goal.

Note 1 to entry: Agents can communicate using HSTP.

3.1.4. Channel PREFERRED

A Spatial Web Entity that groups a stream of HSML entities related to an activity in a specific context that does not itself warrant a domain or hierarchy. Used for grouping and searching limited contexts.

3.1.5. Contract PREFERRED

Represents contractual agreements governing transactions. Typically an agreement for the performance of an HSML Activity Instance.

Note 1 to entry: HSTP supports contract negotiation.

3.1.6. CorrelationId PREFERRED

A unique identifier used for correlating a message instance with related messages in a conversational sequence.

3.1.7. Credential PREFERRED

Represents verifiable claims about an Entity.

Note 1 to entry: Credentials are used in HSTP for authentication and access control.

Note 2 to entry: HSTP uses Credentials for validating access to decryption keys for message contents.

3.1.8. Domain PREFERRED

A central element representing any Entity with a persistent identity.

Note 1 to entry: Domains are identified by SWIDs. Domains provide a conceptual area to describe reality.

Note 2 to entry: Relationships between Domains are managed in the UDG.

3.1.9. Entity PREFERRED

The base class for all Spatial Web entities.

Note 1 to entry: Within HSTP, Entities are queried, created, updated, or deleted via specific OPERATIONs. All Entities that have a persistent identity are modeled as Domains.

3.1.10. hstpVersion PREFERRED

An Envelope Field specifying the version of the HSTP protocol being used.

3.1.11. HSTP OPERATION PREFERRED

An ALL_CAPS protocol verb that identifies a specific transactional action.

[example] LOOKUP and COMMIT are examples of HSTP operations.

Note 1 to entry: HSTP uses OPERATIONs to route messages and encapsulate HSML Activity Schemas.

3.1.12. Hyperspace PREFERRED

A generalized concept of space, fundamental to the Spatial Web.

Note 1 to entry: HSML defines representations of hyperspace.

3.1.13. Message Envelope PREFERRED

The JSON-LD wrapper carrying an HSTP OPERATION, its parameters, and any metadata

[example] A signature and timestamp are examples of message envelope metadata.

3.1.14. messageId PREFERRED

An Envelope Field providing a unique identifier for a specific message instance.

3.1.15. Neighbors PREFERRED

A list of a system's publicly-listed neighboring SWIDs within the *Universal Domain Graph* (3.1.27) (UDG), at a minimum distance of 1.

3.1.16. operation PREFERRED

An Envelope Field specifying the intended action or HSTP OPERATION to be performed.

Note 1 to entry: This field dictates the primary purpose and semantics of the message.

3.1.17. payload PREFERRED

An Envelope Field containing the data relevant to the operation.

Note 1 to entry: This field carries representations of HSML Entities or ActivityTypes.

3.1.18. Requester PREFERRED

The SWID of the Domain sending an HSTP OPERATION request.

3.1.19. Resolver PREFERRED

A system that responds to a "resolve" flow by proposing values for Activity parameters.

3.1.20. Resolvers PREFERRED

An optional Envelope Field included in an HSTP OPERATION response, containing a mapping that proposes values for parameters defined in the requested HSML Activity schema.

3.1.21. signature PREFERRED

An Envelope Field containing cryptographic signature information to verify the authenticity and integrity of the message.

3.1.22. Spatial Web Identifier SWID PREFERRED

A unique identifier for a Domain.

Note 1 to entry: A SWID conforms to the W3C Decentralized Identifier (DID) core standard.

3.1.23. Target PREFERRED

An Envelope Field identifying the intended recipient or target system for the operation specified in the message.

3.1.24. timestamp PREFERRED

An Envelope Field indicating the time the message was created.

3.1.25. Time PREFERRED

Represents temporal aspects within the Spatial Web. Time may be related to Hyperspace and Activities.

3.1.26. UDG Context PREFERRED

Metadata about the sender's position in the *Universal Domain Graph* (3.1.27) (UDG) SWID, known neighbor SWIDs, spatial or routing hints.

3.1.27. Universal Domain Graph
UDG PREFERRED

A distributed metagraph which contains all relationships between all known entities in the Spatial Web.

3.2. Abbreviated terms

CRUD	Create, Read, Update, Delete
DID	Decentralized Identifier
HSML	Hyperspace Modelling Language
HSTP	Hyperspace Transaction Protocol
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
JDF	Joint Development Foundation
JSON-LD	JSON for Linking Data
OData	Open Data Protocol
PDR	Preliminary Design Review
QUIC	Quick UDP Internet Connections
RFC	Request for Comments
SWID	Spatial Web Identifier
UDG	Universal Domain Graph
W3C	World Wide Web Consortium

4. Conceptual Model

4.1. Overview

This clause specifies the fundamental structure of HSTP messages and the core data entities they transport and operate upon, as required for building HSTP-compliant systems that implement the IEEE P2874 Spatial Web system design.

4.2. Architectural Principle: Symmetric Communication

HSTP is designed to support symmetric communication between Spatial Web nodes. Any node can act as both a requester and a responder, enabling peer-to-peer interactions rather than being limited to a strict client-server architecture. This principle is fundamental to the collaborative and decentralized nature of the Spatial Web.

4.3. HSTP Message Syntax

4.3.1. General

This clause defines the standardized structure of the Hyperspace Transaction Protocol (HSTP) message envelope and how these messages are conveyed over various normative transport protocols. The message syntax is designed to encapsulate and exchange the rich, structured data defined by the Hyperspace Modelling Language (HSML).

4.3.2. HSTP OPERATIONS

4.3.2.1. Request message envelope structure

The operations defined in this section are the core verbs of the protocol. Unlike generic transport protocol methods (e.g., HTTP GET/POST), each HSTP OPERATION carries specific semantic value within the context of the Spatial Web. They are designed to manage the lifecycle of HSML entities and orchestrate the complex, stateful interactions required by agents and domains.

The HSTP Operation Request Message Envelope **shall** include the following required fields:

1. *hstpVersion*: Specifies the version of the HSTP protocol being used
   - **Required data type**: String
   - **Required format**: Must follow Semantic Versioning 2.0.0 — `<MAJOR>.<MINOR>.<PATCH>`, optionally with pre-release and build metadata (e.g., `1.0.0`, `2.1.0-beta.1`, `3.0.0+build.45`)
2. *messageId*: A unique identifier for this specific message instance
   - **Required data type**: String

- **Required format**: UUID, version 4 (random-based) as defined in RFC4122
- **Requirements for uniqueness**: Implementations **shall** ensure sufficient uniqueness for message correlation and idempotency

3. *timestamp*: The time the message was created, according to the local clock of the entity generating the message envelope
   - **Required data type**: String
   - **Required format**: Must conform to ISO 8601 date-time format (e.g., `2025-05-27T14:23:30Z` or `2025-05-27T14:23:30+00:00`)
4. *operation*: Specifies the intended action or operation to be performed by the target system, including the operation name and any operation-specific parameters. This field dictates the primary purpose and semantics of the message: HSTP uses these operations to route messages and encapsulate, among other things, HSML Activity Schemas, which form the structural basis for protocol commands.
   - **Required data type**: Object
   - **Required format**: JSON object with the following structure:
   - **name** (required): String specifying the operation name
   - **Format**: Uppercase letters (A–Z), digits (0–9), and underscores (_) only (e.g., CREATE_ENTITY, EXECUTE_ACTIVITY)
   - **References**: Must reference a value from the defined set of HSTP Operations/Verbs listed in Section $1params (optional): Object containing operation-specific parameters
   - **Format**: JSON object where keys are parameter names (strings) and values can be any valid JSON type as defined by the specific operation's parameter schema
   - **References**: Valid parameters and their schemas are defined in the individual operation specifications referenced in Section 4.3
   - **Default**: Empty object {} if no parameters are required or specified
5. *target*: Identifies the intended recipient or target system for the operation specified in the message. This field is included in an HSTP OPERATION request.
   - **Required data type**: String
   - **Required format**: 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. SWIDs **shall** conform to the W3C Decentralized Identifier (DID) core standard.
   - NOTE: should this be just a SWID? This section is under development and will be detailed in a future draft.
6. *requester*: The SWID of the Domain sending the HSTP OPERATION request.
   - **Required data type**: String
   - **Required format**: Must be a SWID. SWIDs **shall** conform to the W3C Decentralized Identifier (DID) core standard.
7. *payload*: Contains the data relevant to the operation. While its primary purpose is to carry HSML Entities, the payload is transport-agnostic. Depending on the operation and transport binding negotiations, it may contain any data format, identified

by its media type (e.g., application/pdf, image/jpeg). For simple results, such as a single measurement or status, the payload should contain a minimal, semantically appropriate HSML entity (e.g., an hsml:Observation). The specific structure and constraints for a given operation's payload are defined by that operation's SHACL schema.

- Required data type/Format*: JSON-LD document is specified as the primary format for the Message Envelope and HSML payloads. The specific encoding format for HSML entities and ActivityTypes within the payload is defined by the HSML Implementation Specification. HSTP relies on these defined formats (e.g. JSON-LD, RDF, SHACL) to ensure interoperability.
- General Structure: The payload structures shall conform to the encoding rules defined in the HSML Implementation Specification, which may include formats like JSON (ISO/IEC 21778), OData (ISO/IEC 20802-1 and ISO/IEC 20802-2), JSON-LD (W3C json-ld11). The payload contains HSML Entities (either a single Entity or a list of Entities)

8. *signature*: Cryptographic signature information to verify the authenticity and integrity of the message.
   - Note: Specify the Required data type/format (e.g., JSON Web Signature structure) and the signature algorithm(s) to be used. This section is under development and will be detailed in a future draft.

The HSTP Message Envelope may include the following required fields:

1. *correlationId*: An optional unique identifier used for correlating this message instance with related messages in a conversational sequence.
   - Required data type: String.
   - Required format: The value shall be a 16-byte array encoded as 32 lowercase hexadecimal characters, conforming to the trace-id format defined in the W3C Trace Context specification.
   - Other requirements: Used for message correlation. When present, implementations shall propagate it across service boundaries.
2. *addresses*: An optional catalog of the sending system's (requester's in a request, target's in a response) publicly listable networked addresses. HSTP messages shall include information needed to locate networked addresses, be it TCP/IP addresses or other. Can be included in both HSTP OPERATION requests and responses.
   - Required data type: Note: The sources describe this as a "catalog", implying a structured data type like an array or object, but the specific format is not detailed. This needs to be defined. This section is under development and will be detailed in a future draft.
   - Required format: Note: Must be a format that allows for locating networked addresses. The specific structure (e.g., list of strings, map of address types to values) needs specification. This section is under development and will be detailed in a future draft.

3. *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).
   - Required data type: List of Strings
   - Required format: Each string in the array must be a SWID. SWIDs shall conform to the W3C Decentralized Identifier (DID) core standard
   - Requirements: Can be included in both HSTP OPERATION requests and responses. The list includes neighbors to at a minimum a distance of 1 on the UDG. The graph distance reported may be specified by the HSTP request or response.
4. *resolvers*: An optional field included in an HSTP OPERATION response. It is a mapping that proposes values for parameters defined in the requested HSML Activity schema.
   - Required data type: Map.
   - Required format: A mapping structure where keys correspond to parameters defined in the Activity schema and values are proposed HSML Entities or other relevant data. The example given includes the SWID and signatures of the Resolving Activities.
   - Note: The specific structure and expected content of this mapping need to be fully detailed. This section is under development and will be detailed in a future draft.
   - Requirements: This field is optional in responses.

4.3.2.2. SHACL Schema for HSTP OPERATION Request

This schema describes the data of which an HSTP Operation Request is composed, but note that the details of how this is transmitted are delegated to the bindings section of this document.

FIGURE 1

```
@prefix hstp: <https://example.org/hstp#> .
@prefix hsml: <https://spec.hsml.dev.verses.build/ns/
hsml#> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
 .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

hstp:MessageEnvelopeShape
    a sh:NodeShape ;
    sh:targetClass hstp:MessageEnvelope ;
    sh:property [
        sh:path hstp:hstpVersion ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
```

```
        sh:pattern "^[0-9]+\\.[0-9]+\\.[0-9]+(-[a-zA-Z0-9.
-]+)?(\\+[a-zA-Z0-9.-]+)?$" ;
        sh:description "Must follow Semantic Versioning
2.0.0 format"
    ] ;
    sh:property [
        sh:path hstp:messageId ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:pattern "^[0-9a-f]{8}-[0-9a-f]{4}-4[0-9a-f]{3}-
[89ab][0-9a-f]{3}-[0-9a-f]{12}$" ;
        sh:description "Must be a valid UUID version 4"
    ] ;
    sh:property [
        sh:path hstp:timestamp ;
        sh:datatype xsd:dateTime ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:description "Must conform to ISO 8601 date-
time format"
    ] ;
    sh:property [
        sh:path hstp:operation ;
        sh:node hstp:OperationShape ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:description "Operation specification with name
and optional parameters"
    ] ;
    sh:property [
        sh:path hstp:target ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:pattern "^did:swid:" ;
        sh:description "Must be a valid SWID conforming
to W3C DID standard"
    ] ;
    sh:property [
        sh:path hstp:requester ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:pattern "^did:swid:" ;
        sh:description "Must be a valid SWID conforming
to W3C DID standard"
    ] ;
    sh:property [
        sh:path hstp:payload ;
        sh:node hstp:PayloadShape ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:description "Payload containing data with
mimetype specification"
    ] ;
    sh:property [
        sh:path hstp:signature ;
        sh:node hstp:SignatureShape ;
```

```
            sh:minCount 1 ;
            sh:maxCount 1 ;
            sh:description "Cryptographic signature
    information"
        ] ;
        sh:property [
            sh:path hstp:correlationId ;
            sh:datatype xsd:string ;
            sh:minCount 0 ;
            sh:maxCount 1 ;
            sh:pattern "^[0-9a-f]{32}$" ;
            sh:description "Optional correlation identifier.
    Conforms to the 32-character lowercase hex W3C Trace
    Context trace-id format."
        ] ;
        sh:property [
            sh:path hstp:addresses ;
            sh:node hstp:AddressListShape ;
            sh:minCount 0 ;
            sh:maxCount 1 ;
            sh:description "Optional list of network
    addresses"
        ] ;
        sh:property [
            sh:path hstp:neighbors ;
            sh:datatype xsd:string ;
            sh:minCount 0 ;
            sh:pattern "^did:swid:" ;
            sh:description "Optional list of neighboring
    SWIDs in the UDG"
        ] ;
        sh:property [
            sh:path hstp:resolvers ;
            sh:node hstp:ResolversShape ;
            sh:minCount 0 ;
            sh:maxCount 1 ;
            sh:description "Optional mapping of resolver
    parameters"
        ] .

    hstp:OperationShape
        a sh:NodeShape ;
        sh:property [
            sh:path hstp:name ;
            sh:datatype xsd:string ;
            sh:minCount 1 ;
            sh:maxCount 1 ;
            sh:pattern "^[A-Z0-9_]+$" ;
            sh:description "Operation name using uppercase
    letters, digits, and underscores"
        ] ;
        sh:property [
            sh:path hstp:params ;
            sh:minCount 0 ;
            sh:maxCount 1 ;
            sh:description "Optional operation-specific
    parameters"
        ] .
```

```
    # Updated PayloadShape - now supports mimetype and
flexible content

hstp:PayloadShape
    a sh:NodeShape ;
    sh:property [
        sh:path hstp:mimetype ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:pattern "^[a-zA-Z0-9][a-zA-Z0-9!#$&\\-\\^_]*\\/
[a-zA-Z0-9][a-zA-Z0-9!#$&\\-\\^_.]*$" ;
        sh:description "MIME type of the payload content
(e.g., 'application/ld+json', 'text/plain', 'application/
json')"
    ] ;
    sh:property [
        sh:path hstp:content ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:description "The actual payload content - can
be any type based on mimetype"
    ] .

hstp:SignatureShape
    a sh:NodeShape ;
    sh:property [
        sh:path hstp:protected ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:description "Protected header of the signature"
    ] ;
    sh:property [
        sh:path hstp:signature ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:description "The signature value"
    ] .

hstp:AddressListShape
    a sh:NodeShape ;
    sh:property [
        sh:path hstp:addressEntry ;
        sh:node hstp:AddressShape ;
        sh:minCount 0 ;
        sh:description "Individual address entries"
    ] .

hstp:AddressShape
    a sh:NodeShape ;
    sh:property [
        sh:path hstp:type ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:in ( "tcp" "websocket" "http" "https" ) ;
        sh:description "Address type"
```

```
        ] ;
        sh:property [
            sh:path hstp:address ;
            sh:datatype xsd:string ;
            sh:minCount 1 ;
            sh:maxCount 1 ;
            sh:description "Network address"
        ] .

    hstp:ResolversShape
        a sh:NodeShape ;
        sh:property [
            sh:path hstp:resolverEntry ;
            sh:node hstp:ResolverShape ;
            sh:minCount 0 ;
            sh:description "Individual resolver entries"
        ] .

    hstp:ResolverShape
        a sh:NodeShape ;
        sh:property [
            sh:path hstp:swid ;
            sh:datatype xsd:string ;
            sh:minCount 1 ;
            sh:maxCount 1 ;
            sh:pattern "^did:swid:" ;
            sh:description "SWID of the resolver"
        ] ;
        sh:property [
            sh:path hstp:signatures ;
            sh:datatype xsd:string ;
            sh:minCount 0 ;
            sh:description "List of signatures"
        ] .
```

4.3.2.3. JSON payload for an HSTP OPERATION request

The following example presents a JSON payload for an HSTP OPERATION request.

FIGURE 2

```
{
  "hstpVersion": "1.0.0-draft",
  "messageId": "550e8400-e29b-41d4-a716-446655440000",
  "timestamp": "2025-05-23T10:41:19Z",
  "operation": {
    "name": "EXECUTE_ACTIVITY"
  },
  "target": "did:swid:example:spatial-domain-456",
  "requester": "did:swid:example:client-domain-789",
  "payload": {
    "@context": {
      "hsml": "https://spec.hsml.dev.verses.build/ns/
hsml#",
      "xsd": "http://www.w3.org/2001/XMLSchema#",
```

```
        "time": "http://www.w3.org/2006/time#"
      },
      "@type": "hsml:Activity",
      "hsml:name": "Inspect Solar Panel",
      "hsml:hasStatus": "hsml:Planned",
      "hsml:performedBy": {
        "@type": "hsml:Person",
        "hsml:name": "Technician A"
      },
      "hsml:performedIn": {
        "@type": "hsml:SpatialFeature",
        "hsml:name": "Solar Farm Zone 3"
      },
      "hsml:performedOn": {
        "@type": "hsml:Thing",
        "hsml:name": "Solar Panel #4521"
      },
      "hsml:hasActivitySchema": {
        "@type": "hsml:AtomicActivitySchema",
        "hsml:name": "Solar Panel Inspection Schema",
        "hsml:hasVariable": [
          {
            "@type": "hsml:Variable",
            "sh:path": "hsml:inspectionTime",
            "sh:datatype": "xsd:dateTime"
          },
          {
            "@type": "hsml:Variable",
            "sh:path": "hsml:resultStatus",
            "sh:datatype": "xsd:string"
          }
        ]
      },
      "hsml:precondition": {
        "@type": "hsml:Condition",
        "hsml:name": "Access Permission Verified"
      },
      "time:hasBeginning": {
        "@type": "time:Instant",
        "time:inXSDDateTime": "2025-07-15T08:00:00Z"
      },
      "time:hasEnd": {
        "@type": "time:Instant",
        "time:inXSDDateTime": "2025-07-15T09:00:00Z"
      }
    },
    "signature": {
      "protected": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjEyMzQ1NiJ9
",
      "signature": "dBjftJeZ4CVP-mB92K27uhbUJU1p1r_
wW1gFWFOEjXk"
    },
    "correlationId": "0af7651916cd43dd8448eb211c80319c",
    "addresses": [
      {
        "type": "tcp",
        "address": "192.168.1.100:8080"
      },
      {
```

```
          "type": "websocket",
          "address": "wss://example.com/hstp"
        }
      ],
      "neighbors": [
        "did:swid:example:neighbor-domain-001",
        "did:swid:example:neighbor-domain-002",
        "did:swid:example:neighbor-domain-003"
      ],
      "resolvers": {
        "activity_executor": {
          "swid": "did:swid:example:executor-domain-555",
          "signatures": [
            "sig-abc123",
            "sig-def456"
          ]
        },
        "validation_service": {
          "swid": "did:swid:example:validator-domain-777",
          "signatures": [
            "sig-ghi789"
          ]
        }
      }
    }
```

4.3.3. Response message envelope structure

All HSTP OPERATION Responses are encapsulated within a standardized message envelope. This envelope is specified as an RDF graph. The envelope contains the necessary metadata to route the message, identify the intended OPERATION, provide context, and ensure authenticity.

The HSTP Operation Response Message Envelope **shall** include the following required fields:

1. *hstpVersion*: Specifies the version of the HSTP protocol being used.
   - **Required data type**: String
   - **Required format**: Must follow Semantic Versioning 2.0.0 — <MAJOR>.<MINOR>.<PATCH>, optionally with pre-release and build metadata.
2. *messageId*: A unique identifier for this specific response message instance.
   - **Required data type**: String
   - **Required format**: UUID, version 4 (random-based) as defined in RFC4122.
   - **Requirements for uniqueness**: Implementations **shall** ensure sufficient uniqueness for message correlation and idempotency.
3. *timestamp*: The time the response message was created, according to the local clock of the entity generating the message envelope.
   - **Required data type**: String

- **Required format**: Must conform to ISO 8601 date-time format (e.g., 2025-08-29T17:51:02Z).

4. *target*: Identifies the intended recipient of this response, which is the original requester.
   - **Required data type**: String
   - **Required format**: Must be a SWID. SWIDs **shall** conform to the W3C Decentralized Identifier (DID) core standard.
5. *responder*: The SWID of the Domain sending this response.
   - **Required data type**: String
   - **Required format**: Must be a SWID. SWIDs **shall** conform to the W3C Decentralized Identifier (DID) core standard.
6. *response*: An object containing the status of the response.
   - **Required data type**: Object
   - **Required format**: A JSON object containing a required status field from a predefined enumeration (e.g., SUCCESS_0, FAILURE_1) and an optional failureMessage string if the status is a failure .
7. *signature*: Cryptographic signature information to verify the authenticity and integrity of the message.
   - Note: The specific data type/format (e.g., JSON Web Signature structure) and the required signature algorithm(s) will be detailed in a future revision.

4.3.3.1. SHACL Schema for HSTP Operation Response Envelope

This schema describes the data of which an HSTP Operation Response is composed, but note that the details of how this is transmitted are delegated to the bindings section of this document.

FIGURE 3

```
@prefix hstp: <https://spec.hstp.dev.verses.build/schemas/
v1/> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
 .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

hstp:ResponseEnvelopeShape
    a sh:NodeShape ;
    sh:targetClass hstp:ResponseEnvelope ;
    sh:property [
        sh:path hstp:hstpVersion ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
```

```
        sh:pattern "^[0-9]+\\.[0-9]+\\.[0-9]+(-[a-zA-Z0-9.
-]+)?(\\+[a-zA-Z0-9.-]+)?$" ;
        sh:description "Must follow Semantic Versioning
2.0.0 format"
    ] ;
    sh:property [
        sh:path hstp:messageId ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:pattern "^[0-9a-f]{8}-[0-9a-f]{4}-4[0-9a-f]{3}-
[89ab][0-9a-f]{3}-[0-9a-f]{12}$" ;
        sh:description "Must be a valid UUID version 4
for this response"
    ] ;
    sh:property [
        sh:path hstp:timestamp ;
        sh:datatype xsd:dateTime ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:description "Must conform to ISO 8601 date-
time format"
    ] ;
    sh:property [
        sh:path hstp:target ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:pattern "^did:swid:" ;
        sh:description "SWID of the original requester
(recipient of this response)"
    ] ;
    sh:property [
        sh:path hstp:responder ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:pattern "^did:swid:" ;
        sh:description "SWID of the node sending this
response"
    ] ;
    sh:property [
        sh:path hstp:response ;
        sh:node hstp:ResponseShape ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:description "Response details including status
and optional metadata"
    ] ;
    sh:property [
        sh:path hstp:payload ;
        sh:node hstp:ResponsePayloadShape ;
        sh:minCount 0 ;
        sh:maxCount 1 ;
        sh:description "Optional response payload"
    ] ;
    sh:property [
        sh:path hstp:signature ;
        sh:node hstp:SignatureShape ;
```

```
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:description "Cryptographic signature
information"
    ] ;
sh:property [
        sh:path hstp:correlationId ;
        sh:datatype xsd:string ;
        sh:minCount 0 ;
        sh:maxCount 1 ;
        sh:pattern "^[0-9a-f]{32}$" ;
        sh:description "Correlation ID from the original
request, conforming to the W3C Trace Context trace-id
format." ] ;
    sh:property [
        sh:path hstp:addresses ;
        sh:node hstp:AddressListShape ;
        sh:minCount 0 ;
        sh:maxCount 1 ;
        sh:description "Responder's publicly listable
network addresses"
    ] ;
    sh:property [
        sh:path hstp:neighbors ;
        sh:datatype xsd:string ;
        sh:minCount 0 ;
        sh:pattern "^did:swid:" ;
        sh:description "Responder's neighboring SWIDs in
the UDG"
    ] ;
    sh:property [
        sh:path hstp:resolvers ;
        sh:node hstp:ResolversShape ;
        sh:minCount 0 ;
        sh:maxCount 1 ;
        sh:description "Optional activity resolver
mapping"
    ] .

hstp:ResponseShape
    a sh:NodeShape ;
    sh:property [
        sh:path hstp:status ;
        sh:in ( "SUCCESS_0" "FAILURE_1" "DEFERRED_TO_TELL_
MAILBOX_2" ) ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:description "Response status from predefined
enumeration"
    ] ;
    sh:property [
        sh:path hstp:failureMessage ;
        sh:datatype xsd:string ;
        sh:minCount 0 ;
        sh:maxCount 1 ;
        sh:description "Optional failure message when
status is FAILURE_1"
    ] .
```

```
    hstp:ResponsePayloadShape
        a sh:NodeShape ;
        sh:property [
            sh:path hstp:payloadData ;
            sh:datatype xsd:string ;
            sh:minCount 0 ;
            sh:maxCount 1 ;
            sh:description "The actual payload content as
    text, binary blob (base64), or external reference"
        ] ;
        sh:property [
            sh:path hstp:mimeType ;
            sh:datatype xsd:string ;
            sh:minCount 1 ;
            sh:maxCount 1 ;
            sh:description "MIME type of the payload content"
        ] ;
        sh:property [
            sh:path hstp:hsml_content ;
            sh:datatype xsd:string ;
            sh:minCount 0 ;
            sh:maxCount 1 ;
            sh:description "HSML content as JSON-LD string
    (temporary stopgap)"
        ] .

    hstp:SignatureShape
        a sh:NodeShape ;
        sh:property [
            sh:path hstp:protected ;
            sh:datatype xsd:string ;
            sh:minCount 1 ;
            sh:maxCount 1 ;
            sh:description "Protected header of the signature"
        ] ;
        sh:property [
            sh:path hstp:signature ;
            sh:datatype xsd:string ;
            sh:minCount 1 ;
            sh:maxCount 1 ;
            sh:description "The signature value"
        ] .

    hstp:AddressListShape
        a sh:NodeShape ;
        sh:property [
            sh:path hstp:addressEntry ;
            sh:node hstp:AddressShape ;
            sh:minCount 0 ;
            sh:description "Individual address entries"
        ] .

    hstp:AddressShape
        a sh:NodeShape ;
        sh:property [
            sh:path hstp:type ;
            sh:datatype xsd:string ;
            sh:minCount 1 ;
            sh:maxCount 1 ;
```

```
            sh:in ( "tcp" "websocket" "http" "https" ) ;
            sh:description "Address type"
        ] ;
        sh:property [
            sh:path hstp:address ;
            sh:datatype xsd:string ;
            sh:minCount 1 ;
            sh:maxCount 1 ;
            sh:description "Network address"
        ] .

    hstp:ResolversShape
        a sh:NodeShape ;
        sh:property [
            sh:path hstp:resolverEntry ;
            sh:node hstp:ResolverShape ;
            sh:minCount 0 ;
            sh:description "Individual resolver entries"
        ] .

    hstp:ResolverShape
        a sh:NodeShape ;
        sh:property [
            sh:path hstp:swid ;
            sh:datatype xsd:string ;
            sh:minCount 1 ;
            sh:maxCount 1 ;
            sh:pattern "^did:swid:" ;
            sh:description "SWID of the resolver"
        ] ;
        sh:property [
            sh:path hstp:signatures ;
            sh:datatype xsd:string ;
            sh:minCount 0 ;
            sh:description "List of signatures"
        ] .
```

4.3.3.2. Example JSON payload for HSTP OPERATION Response: Success case

FIGURE 4

```
{
  "hstpVersion": "1.0.0-draft",
  "messageId": "resp-8400-e29b-41d4-a716-446655440001",
  "timestamp": "2025-05-23T10:45:30Z",
  "target": "did:swid:example:client-domain-789",
  "responder": "did:swid:example:spatial-domain-456",
  "response": {
    "status": "SUCCESS_0"
  },
  "payload": {
    "mimeType": "application/json",
```

```
      "payloadData": "{\"@context\":\"https://example.org/
    hsml/context\",\"@type\":\"Entity\",\"id\":\"created-
    entity-456\",\"name\":\"Successfully Created Entity\",
    \"createdAt\":\"2025-05-23T10:45:30Z\"}",
      "hsml_content": "{\"@context\":\"https://example.org/
    hsml/context\",\"@type\":\"Activity\",\"id\":\"create-
    activity-789\",\"status\":\"completed\"}"
    },
    "signature": {
      "protected": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjY3ODkwMSJ9
    ",
      "signature": "kBjftJeZ4CVP-mB92K27uhbUJU1p1r_
    wW1gFWFOEjXl"
    },
    "correlationId": "0af7651916cd43dd8448eb211c80319c",
    "addresses": [
      {
        "type": "tcp",
        "address": "10.0.0.50:8080"
      },
      {
        "type": "websocket",
        "address": "wss://spatial-domain.example.com/hstp"
      }
    ],
    "neighbors": [
      "did:swid:example:neighbor-spatial-001",
      "did:swid:example:neighbor-spatial-002",
      "did:swid:example:neighbor-spatial-003"
    ],
    "resolvers": {
      "validation_service": {
        "swid": "did:swid:example:validator-domain-888",
        "signatures": ["sig-xyz123", "sig-abc789"]
      },
      "storage_service": {
        "swid": "did:swid:example:storage-domain-999",
        "signatures": ["sig-def456"]
      }
    }
  }
```


4.3.3.3. Example JSON payload for HSTP OPERATION Response: Failure case

FIGURE 5


```
  {
    "hstpVersion": "1.0.0-draft",
    "messageId": "resp-8400-e29b-41d4-a716-446655440002",
    "timestamp": "2025-05-23T10:45:35Z",
    "target": "did:swid:example:client-domain-789",
    "responder": "did:swid:example:spatial-domain-456",
    "response": {
```

```
      "status": "FAILURE_1",
      "failureMessage": "Invalid entity schema: missing
    required 'name' field"
    },
    "payload": {
      "mimeType": "application/json",
      "payloadData": "{\"error\":\"SCHEMA_VALIDATION_
    ERROR\",\"details\":\"Entity validation failed: required
    property 'name' is missing\",\"errorCode\":\"HSTP_4001\"}"
    },
    "signature": {
      "protected": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjY3ODkwMSJ9
    ",
      "signature": "lCjftJeZ4CVP-mB92K27uhbUJU1p1r_
    wW1gFWFOEjXm"
    },
    "correlationId": "0af7651916cd43dd8448eb211c80319c",
    "addresses": [
      {
        "type": "tcp",
        "address": "10.0.0.50:8080"
      }
    ],
    "neighbors": [
      "did:swid:example:neighbor-spatial-001"
    ]
  }
```


4.3.3.4. Example JSON payload for HSTP OPERATION Response: Deferred case

FIGURE 6


```
  {
    "hstpVersion": "1.0.0-draft",
    "messageId": "resp-8400-e29b-41d4-a716-446655440003",
    "timestamp": "2025-05-23T10:45:40Z",
    "target": "did:swid:example:client-domain-789",
    "responder": "did:swid:example:spatial-domain-456",
    "response": {
      "status": "DEFERRED_TO_TELL_MAILBOX_2"
    },
    "payload": {
      "mimeType": "application/json",
      "payloadData": "{\"message\":\"Request has been
    queued for processing\",\"mailboxId\":\"mailbox-
    actor-123\",\"estimatedProcessingTime\":\"2025-05-
    23T11:00:00Z\"}"
    },
    "signature": {
      "protected": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjY3ODkwMSJ9
    ",
      "signature": "mDjftJeZ4CVP-mB92K27uhbUJU1p1r_
    wW1gFWFOEjXn"
```

```
      },
      "correlationId": "0af7651916cd43dd8448eb211c80319c",
      "addresses": [
        {
          "type": "tcp",
          "address": "10.0.0.50:8080"
        }
      ],
      "neighbors": [
        "did:swid:example:neighbor-spatial-001"
      ]
    }
```

4.4. Payload Encryption

As specified in 4.3.2.1 and 4.3.3, the payload field of an HSTP Message Envelope contains the data relevant to the operation, typically representations of HSML Entities. The content of the payload field shall be encrypted. Domain-specific architectures implementing HSTP shall encrypt HSTP payloads. This requirement is part of the broader need for HSTP to incorporate mechanisms for encryption to safeguard interactions.

The Hyperspace Transaction Protocol (HSTP) includes OPERATIONs such as ENCRYPT_PAYLOAD and DECRYPT_PAYLOAD, which fall under the category of OPERATIONs for "Encrypt / Decrypt Communications". The sender utilizes the ENCRYPT_PAYLOAD operation, which encrypts the payload so it can be forwarded and decrypted using a claim in a verifiable credential.

Recipients of an HSTP message requiring decryption utilize the DECRYPT_PAYLOAD operation. The ability to successfully execute this operation on a specific payload is gated behind a claim in a verifiable credential. Credentials represent verifiable claims about an entity and are used in HSTP for authentication and access control.

The recipient's system first validates the authenticity and integrity of the received message, including the sender's identity specified in the `REQUESTER` field and verified via the `signature` field. As part of the zero-trust security model implemented by HSTP, the recipient's system's access control policies evaluate the validated sender's identity and associated claims (potentially presented in credentials or verifiable by the recipient based on the sender's identity) to determine if the recipient possesses the necessary authorization, represented by the required credential claim, to decrypt this specific payload. Thus, authorization to decrypt is controlled by the recipient's policy engine evaluating credentials and claims, leveraging the sender's validated identity and credentials as inputs to that decision process, rather than the sender explicitly specifying a decryption credential within the message envelope.

NOTE 1 Specify the required encryption algorithm(s) and format for the encrypted payload. This specification might consider the `ENCRYPTION` identifier mentioned in SWID Documents as a potential mechanism for indicating encryption capabilities or

requirements and ensure consistency with the ENCRYPT_PAYLOAD operation. This section is under development and will be detailed in a future draft.

NOTE 2 Detail the technical process by which the recipient's system policies use validated sender credentials (obtained and verified via the message envelope fields REQUESTER and signature) to satisfy the "gated behind a claim in a verifiable credential" requirement and gain access to the necessary decryption key(s) for the payload. This may involve detailing interaction with key management systems and policy enforcement points within the recipient's system architecture. This section is under development and will be detailed in a future draft.

NOTE 3 Cross-reference relevant sections in the IEEE P2874 standard [21-23] that mandate payload encryption (e.g., Section 7.1.2.11 [4], Section 7.1.3 [5], requirements like HSTP-15 [6, 7]), define the use of credentials for access control and decryption key access (e.g., Section 6.5 [11, 12], Section 7.1.2.11 [4, 5], requirements like HSTP-15 [6, 7] and HSTP-23 [17-19]), and specify the ENCRYPT_PAYLOAD/DECRYPT_PAYLOAD OPERATIONs (e.g., Section 7.3.3.1 [8, 24], Section 7.3.5 [8, 25]). This section is under development and will be detailed in a future draft.

4.5. HSTP OPERATIONs

4.5.1. General

The operations defined in this section are the core verbs of the protocol. Unlike generic transport protocol methods (e.g., HTTP GET/POST), each HSTP OPERATION carries specific semantic value within the context of the Spatial Web. They are designed to manage the lifecycle of HSML entities and orchestrate the complex, stateful interactions required by agents and domains.

4.5.2. HSTP OPERATION components

An HSTP OPERATION consists of the following components:

1. Bootstrapping
   - *REGISTER_DOMAIN*: Create/register a new DOMAIN; server returns a SWID and private key
   - *ISSUE_IDENTITY_CREDENTIAL*: Issue an identity credential for an actor under a DOMAIN
2. Domain management
   - *CREATE_CHILD_DOMAIN*: Creates a child domain within the domain of the spatial web node handling the operation
   - *TRANSFER_DOMAIN*: Transfers ownership of a domain from one SWID (domain) to another SWID (domain)
3. Activity and actor management
   - *REGISTER_ACTIVITY_EXECUTOR*: Register an activity handler in the UDG; works closely with FIND_ACTIVITY_EXECUTOR

- — *EXECUTE_ACTIVITY*: Request that an instance of an Activity be executed
- — *CHECK_ACTOR_MAILBOX*: Checks the mailbox of the requesting entity (obtained via the 'requester' field of the HSTP OPERATION Request); if there is an ExecuteActivity Operation Request in the mailbox, removes the request from the mailbox and returns it in the response to CHECK_ACTOR_MAILBOX so that the actor can act on it. Implicitly supports only TELL (not ASK) actor semantics.
- — *GET_SITUATION_OF_ACTOR_AT_POINT_IN_TIME*: Queries for a specific SITUATION associated to a specific actor (e.g. an "agent") at a specific point in time.
  - A SITUATION relationship is an entity composed of all DOMAINS that can be perceived and reasoned about by an AGENT. Situation is to be understood as defined in IEEE 7007™-2021: a situation is an entity composed of participating entities and relationships that represent the limited parts of reality that can be perceived and reasoned about by agents.

4. Query Entities — use UDG syntax and semantics to query the UDG : HSTP provides query operations to retrieve data from a node. While all data retrieval is technically a "query" of the underlying data store, HSTP defines specific query patterns to simplify common use cases. To avoid exposing a raw query interface (e.g., SPARQL) and to ensure controlled, secure data access, nodes shall expose these queries as "named canonical queries" or templates. This allows clients to perform both simple 'read' actions and complex queries through a consistent, secure mechanism.
   - — *GET_SHAPES*:: returns an rdf graph (turtle format) enumerating all of the shapes supported by the Domain(s) of the node in the context of the Credential used to query
     - *QUERY_SYNC*: synchronous request/response query pattern (including optional pagination syntax). This OPERATIONs supports multiple query types, including:
     - *READ_ENTITY named query*: A simple, canonical query pattern used to retrieve attributes from a single, specified entity. This pattern serves as the primary "Read" function within HSTP's CRUD operations, abstracting the complexity of a full query for simple value retrieval.
   - — *QUERY_ASYNC*: asynchronous request/response query pattern. This OPERATIONs supports multiple query types, including:
     - *QUERY_STREAM query type*: use a stream to transport sequences of entities; not as robust as a queue; allows for e.g. at most once delivery semantics
     - *QUERY_QUEUE query type*: use a queue to transport sequences of entities; more robust than a stream; allows for e.g. at least once delivery semantics
5. Perform CRUD on Entities
   - — *CREATE_ENTITY*: creates an entity in the UDG
   - — *UPDATE_ENTITY*: updates an entity in the UDG (idempotent)
   - — *DELETE_ENTITY*: deletes an entity in the UDG
6. Authenticate / Verify Entity
   - — *VERIFY_IDENTITY_CREDENTIAL*: Verifies that a Credential is valid
7. Verify Credentials / Capabilities

- *VALIDATE_CREDENTIAL_SIGNATURE*: Verifies that something was signed by a specific Credential (identity)
- *VALIDATE_VERIFIABLE_CREDENTIAL*: encompasses the following
  - Issuer Verification: Confirm the credential was issued by a legitimate and authorized entity
  - Subject Confirmation: Ensure the credential pertains to the correct subject and that the presenter has the right to present it
  - Signature Validation: Check the digital signature to confirm the credential's integrity and authenticity
  - Status and Date Checks: Verify the credential is currently valid and has not been revoked
  - Schema and Context Assessment: Ensure the credential's structure and semantics align with expected standards

8. Authorize / Manage Access
   - *ISSUE_VERIFIABLE_CREDENTIAL*: Generates and delivers a cryptographically signed credential asserting claims about an entity
   - *REVOKE_VERIFIABLE_CREDENTIAL*: Marks a verifiable credential as invalid by updating its status in the associated registry or source materials
9. Discover Entities / Services
   - *FIND_ACTIVITY_EXECUTOR*: to a SWID that is capable of performing it (semantic sugar over QUERY)
   - *DISCOVER_DOMAINS_USING_UDG*: query across the UDG to discover domains (N steps deep)
   - *DISCOVER_DOMAINS_USING_SPACE*: query the UDG using spatial constraints
   - *LIST_SHAPES*: Returns a representation of the shapes (graph) used by the spatial web node
   - *GET_MAP*: Returns a conceptual map of a target domain, providing an agent with situational awareness.
10. Declare Capabilities / Resolvers
    - *GET_SUPPORTED_OPERATIONS*: enumerates the OPERATIONs supported by the spatial web node
    - *GET_SUPPORTED_ACTIVITIES*: enumerates the Activities supported by the spatial web node
11. Encrypt / Decrypt Communications
    - *ENCRYPT_PAYLOAD*: encrypts the payload of an HSTP request so the request can be forwarded and decrypted using a claim in a verifiable credential
    - *DECRYPT_PAYLOAD*: decrypts a payload (gated behind a claim in a verifiable credential)
12. Channels: NOTE: — Channel operations group messages related to a specific, ongoing Activity instance. The HSML Activity definition itself defines and constrains

the Channels that are used within the context of that Activity. Channels are not a prerequisite for all HSTP interactions.

- *LIST_CHANNELS*: list channels owned by this spatial web node;
- *QUERY_CHANNELS*: query for a subset of channels owned by this spatial web node
- *PUBLISH_TO_CHANNEL*: publishes something to a channel owned by this spatial web node
- *SUBSCRIBE_TO_CHANNEL*: subscribe to a channel owned by this spatial web node

4.5.3. HSTP OPERATION: EXECUTE_ACTIVITY

4.5.3.1. Overview

Provided as an example of how I suggest we flesh out the details of each OPERATION type.

4.5.3.2. Summary

Indicates that the requester would like the target to take action that results in conditions of completion of the Activity are met.

4.5.3.3. Behavior

Activity dependent.

4.5.3.4. Parameters

none

4.5.3.5. HSTP OPERATION Request SHACL schema

FIGURE 7

```
# Specific shape for EXECUTE_ACTIVITY payloads
hstp:ExecuteActivityPayloadShape
    a sh:NodeShape ;
    sh:property [
        sh:path hstp:mimetype ;
        sh:hasValue "application/ld+json" ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:description "EXECUTE_ACTIVITY payloads must be
JSON-LD"
    ] ;
    sh:property [
        sh:path hstp:content ;
        sh:node [
            a sh:NodeShape ;
            sh:property [
                sh:path rdf:type ;
                sh:hasValue hsml:Activity ;
                sh:minCount 1 ;
                sh:maxCount 1 ;
```

```
                sh:description "Content must be
explicitly typed as hsml:Activity"
            ]
        ] ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:description "Content must be an HSML Activity"
    ] .

# Shape for envelope when operation.name is EXECUTE_
ACTIVITY
hstp:ExecuteActivityHSTPOperationShape
    a sh:NodeShape ;
    sh:targetClass hstp:MessageEnvelope ;
    sh:property [
        sh:path hstp:operation ;
        sh:node [
            a sh:NodeShape ;
            sh:property [
                sh:path hstp:name ;
                sh:hasValue "EXECUTE_ACTIVITY" ;
                sh:minCount 1 ;
                sh:maxCount 1 ;
                sh:description "Operation name must be
EXECUTE_ACTIVITY"
            ]
        ] ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:description "Operation must be EXECUTE_
ACTIVITY"
    ] ;
    sh:property [
        sh:path hstp:payload ;
        sh:node hstp:ExecuteActivityPayloadShape ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:description "Payload must be hsml:Activity
when executing activity"
    ] .
```

4.5.4. HSTP OPERATION: GET_MAP

4.5.4.1. Summary

Provides an agent with a conceptual map of a target domain. This operation is fundamental for an agent to gain situational awareness when first interacting with an unfamiliar domain. The response provides a synopsis of the domain's contents, including the types of entities it contains and their high-level relationships, acting as a legend for what is available.

4.5.4.2. Behavior

The responding node returns a representation of the domain's conceptual model or "metamodel". This is distinct from a query for specific entity instances; it describes the kinds of things that can exist or be interacted with in the domain. The level of detail

returned depends on the requester's credentials and permissions. The response payload is a structured graph representation (e.g., in HSML) of the domain's available types.

4.5.4.3. Parameters

none

4.5.4.4. HSTP OPERATION Request SHACL schema

FIGURE 8

```
hstp:GetMapHSTPOperationShape
    a sh:NodeShape ;
    sh:targetClass hstp:MessageEnvelope ;
    sh:property [
        sh:path hstp:operation ;
        sh:node [
            a sh:NodeShape ;
            sh:property [
                sh:path hstp:name ;
                sh:hasValue "GET_MAP" ;
                sh:minCount 1 ;
                sh:maxCount 1 ;
                sh:description "Operation name must be
GET_MAP"
            ]
        ] ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:description "Operation must be GET_MAP"
    ] .
```

5. Core Data Structure Representation

5.1. Overview

This clause specifies the fundamental structure of HSTP messages and the core data entities they transport and operate upon, as required for building HSTP-compliant systems that implement the IEEE P2874 Spatial Web system design.

5.2. Core Spatial Web Entities

The Spatial Web ontology is composed of Entities that are the primary concepts used across the Spatial Web. All ENTITIES that have a persistent identity are modeled as DOMAINS. HSTP OPERATIONs are designed to interact with these entities and their relationships as maintained within the Universal Domain Graph (UDG). Core Spatial Web Entities that are fundamental to HSTP interactions include the following:

- ENTITY: The base class for all Spatial Web entities.
- AGENT: Represents an entity capable of action and interaction within the Spatial Web. Agents can communicate using HSTP.
- ACTIVITY: Represents a partially ordered set of changes, defining intended conditions and presupposed conditions. Activities are central to entity-to-entity execution and state change, and can be composed of other Activities. HSTP OPERATIONs support executing and resolving Activities. The UDG keeps a record of executed Activities as part of Contracts.
- DOMAIN: A central element representing any entity with a persistent identity. Domains are identified by SWIDs. Relationships between Domains are managed in the UDG. Domains provide a conceptual area to describe reality from multiple perspectives.
- CONTRACT: Represents contractual agreements governing transactions, typically agreements for the performance of an HSML Activity Instance. HSTP supports contract negotiation.
- CHANNEL: Provides a means of grouping HSML elements together, used to parse functionality and data and as a means of searching a limited context of a spatial graph.
- CREDENTIAL: Represents verifiable claims about an entity. Credentials are used in HSTP for authentication and access control.
- HYPERSPACE: A generalized concept of space, fundamental to the Spatial Web. HSML defines representations of hyperspace. Spaces define boundaries, locations, positions, and geometries.
- TIME: Represents temporal aspects within the Spatial Web.

NOTE 1NOTE 1Ensure all relevant core entities required for fundamental HSTP OPERATIONs are listed and provide their initial, high-level definitions based on the HSML/P2874 conceptual model. Ensure consistency with the main Terminology section (Section 3). This section is under development and will be detailed in a future revision.
NOTE 2NOTE 2Cross-reference the clauses in the IEEE P2874 document (e.g., Clause 6.6) where these entities are defined in detail. This section is under development and will be detailed in a future revision.

5.3. Representation within the Message Payload

The payload field of the HSTP Message Envelope is designed for flexibility. While its primary role is to transport the rich, structured data models defined by HSML, it is not exclusively limited to HSML entities.

As established in Section 4.2.1, and supported by the protocol bindings in Section 6, the payload may contain any data format as negotiated between systems. The specific format and structure of the payload are determined by the operation being performed. For example, an EXECUTE_ACTIVITY operation might return a binary image/jpeg payload, while a CREATE_ENTITY operation will contain an application/ld+json payload representing that entity.

This approach ensures that HSTP can serve as a robust transactional protocol for the diverse needs of the Spatial Web, from managing structured semantic data to exchanging media and other arbitrary data types. All payloads, regardless of format, must be described and constrained by the SHACL shapes associated with the specific operation being invoked.

6. Protocol Bindings and Encodings

6.1. Overview

This section defines how HSTP messages Clause 4 are transported over specific protocols and how the message envelope and payload are encoded. It includes normative transport bindings for HTTP/2 and MQTT, and defines supported encodings for both envelope metadata and payload content. Bindings for additional transports will be added in future revisions.

6.2. General Principles

HSTP (Hyperspace Transaction Protocol) is an application-layer protocol designed to be transport-agnostic and extensible across multiple serialization formats. Every HSTP message is defined by a structured envelope (Section 4.2) that includes metadata fields such as `operation_name`, `messageId`, `timestamp`, `target`, and `requester`, alongside a protocol-agnostic `payload`.

Each transport binding in this section defines how HSTP envelope fields and payloads are mapped to specific transport-layer constructs (e.g., HTTP headers, MQTT topics). These bindings also define protocol-specific behaviors for security, error handling, and message routing.

HSTP messages MAY include transport-specific extensions to negotiate preferences such as encoding, content types, compression, or delivery guarantees. In cases where negotiation is supported (e.g., via HTTP content negotiation or MQTT metadata), the response payload MAY be a media type other than JSON-LD (e.g., PDF, JPEG, MP4), especially for `EXECUTE_ACTIVITY` OPERATION.

To ensure baseline interoperability for discovery across the Spatial Web, all HSTP-compliant nodes shall support the HTTP/2 Binding Profile (Section 6.3). This ensures

that any agent can perform initial discovery operations on any node, regardless of other specialized bindings the node may offer.

6.3. Encodings

6.3.1. Overview

The HSTP envelope is always transmitted as structured metadata per transport-specific conventions. The `payload` is encoded independently and MAY take on different content types depending on the operation, schema, and negotiated preferences.

6.3.2. Supported Media Types

Supported media types are shown in Table 1.

TABLE 1: Table of Media Types

Type	Media Type	Notes
Structured JSON	`application/json`	Used for basic structured data.
JSON-LD	`application/ld+json`	Default serialization format for HSML payloads.
Binary	e.g., `image/jpeg`, `application/pdf`, `application/zip`	Used for activity responses or artifacts negotiated through `EXECUTE_ACTIVITY`.
Textual	`text/plain`, `text/html`	Used for plain text or HTML results.

6.3.3. Negotiation

- HTTP: Content negotiation SHALL use the standard `Accept` header. The `Content-Type` header in responses SHALL reflect the actual media type returned.
- MQTT: Content type MAY be specified in the message properties (e.g., user-defined headers or MQTT v5 content-type property).
- Systems MUST be prepared to handle unexpected content types when explicitly negotiated or default fallbacks are not met.

6.4. HTTP/2 Binding Profile

6.4.1. Purpose

This section defines how HSTP messages are conveyed over HTTP/2, including mappings to HTTP elements, supported encodings, and transport-specific behaviors.

6.4.2. Envelope Mapping

In the HTTP/2 binding, the HSTP message envelope is mapped to standard HTTP constructs in a manner that aligns with web standards. The HSTP payload is sent as the HTTP message body.

For this binding, the HSTP protocol version shall be specified within the standard Content-Type header (for requests) and Accept header (for responses) using a profile parameter. This approach enables standard HTTP content negotiation.

For example:

FIGURE 9

```
Content-Type: application/ld+json; profile="https://spec.
hstp.dev/v1.0.0".
```

To avoid redundancy and ambiguity, the custom HSTP-Version header shall not be used in this binding. Other metadata fields from the envelope are placed into custom HSTP- prefixed headers, as detailed in the table below.

Table Table 2 details the specific mappings.

TABLE 2: HSTP Envelope Field to HTTP/2 Mapping

HSTP envelope field	HTTP/2 Mapping	Notes
`hstpVersion`	Content-Type header profile parameter	Required. Mapped to the profile parameter of the Content-Type header, as a custom HSTP-Version header shall not be used in this binding. Example: `Content-Type: application/ld+json; profile="https://spec.hstp.dev/v1.0.0`"/>
`messageId`	Header: `HSTP-Message-ID`	UUIDv4 as per RFC 4122.
`timestamp`	Header: `HSTP-Timestamp`	ISO 8601 UTC timestamp.
`operation`	Header: `HSTP-Operation`	All caps (e.g., `CREATE_ENTITY`, `EXECUTE_ACTIVITY`). Not inferred from URI or method.
`target`	Header: `HSTP-Target`	SWID or URI.
`requester`	Header: `HSTP-Requester`	SWID of the sending domain.
`signature`	Headers: Signature and Signature-Input	Uses the IETF HTTP Message Signatures standard (RFC 9421). Signature-Input defines the signed components and parameters, while Signature contains the signature value.
`correlationId`	Header: `traceparent`	Optional. W3C Trace Context header. The trace-id portion of this header's value corresponds to the correlationId from the message envelope.

HSTP envelope field	HTTP/2 Mapping	Notes
		Implementations shall forward this header to enable distributed tracing.
`addresses`	Header: `HSTP-Addresses`	Optional. List of sender network addresses.
`neighbors`	Header: `HSTP-Neighbors`	Optional. List of adjacent SWIDs.
`resolvers`	Header: `HSTP-Resolvers`	Optional. Proposed resolver values in responses.
`payload`	HTTP Body	Content-type declared using `Content-Type`. Media type MAY vary (e.g., JSON-LD, JPEG, etc.).

NOTE All `HSTP-` headers SHOULD be registered with IANA.

6.4.3. Patterns

Table Table 3 describes patterns of HSTP Operations and Corresponding HTTP Methods.

TABLE 3: HSTP Operations and Corresponding HTTP Methods

HSTP OPERATION	HTTP Method	Description
`CREATE_ENTITY`	POST	Creates a resource. Response MAY contain location or payload.
`QUERY_ENTITY`	GET	Queries a resource.
`UPDATE_ENTITY`	PUT/PATCH	Updates a resource.
`DELETE_ENTITY`	DELETE	Deletes a resource.
ALL OTHER OPERATIONS	POST	(e.g. `EXECUTE_ACTIVITY` performs a transactional operation, MAY return binary/media data.)

6.4.4. Error Handling

- Status Codes: Use HTTP status codes (e.g., 400, 401, 403, 500) based on error condition.
- Payload Format: Use RFC 7807 Problem Details for structured errors.
- Fallback Behavior: Clients SHOULD handle unknown status codes gracefully.

6.4.5. Security

TLS 1.2 or higher is REQUIRED.

Message authenticity and integrity shall be provided using the IETF HTTP Message Signatures standard (RFC 9421). The Signature and Signature-Input headers are used for this purpose. At a minimum, the signature must cover the following components:

- All HSTP- prefixed headers (e.g., HSTP-Operation, HSTP-Target, HSTP-Requester).
- The @method and @target-uri components.
- The Content-Digest header to ensure payload integrity.

6.4.6. Examples

Request Example:

FIGURE 10

```
POST /hstp HTTP/2
Host: example.com
Content-Type: application/ld+json; profile="https://spec.
hstp.dev/v1.0.0"
Content-Digest: sha-256=:jLL2fCoBEb0g9nB4jFk2RtqPSLo3Z28z2
X4afl23N/A=:
traceparent: 00-0af7651916cd43dd8448eb211c80319c-
b7ad6b7169203331-01
HSTP-Operation: EXECUTE_ACTIVITY
HSTP-Message-ID: 6c90a248-8792-4de7-bccc-3d2e92e46ae2
HSTP-Target: did:example:service1
HSTP-Requester: did:example:client1
Signature-Input: sig1=("@method" "@target-uri" "content-
digest" "traceparent" "hstp-operation" "hstP-target"
"hstp-requester");created=1756415098;keyid="did:example:
client1#key-1"
Signature: sig1=:aGVsbG8gd29ybGQ=…(example signature
bytes)…=:
{ "@context": ..., "@type": "MyActivityRequest", ... }
```

Response Example (PDF):

FIGURE 11

```
200 OK
Content-Type: application/pdf
Content-Digest: sha-256=:VGhpcyBpcyBqdXN0IGFuIGV4YW1wbGUgU
ERGLg==:
traceparent: 00-0af7651916cd43dd8448eb211c80319c-
b7ad6b7169203331-01
Signature-Input: sig1=("@status" "content-type" "content-
digest" "traceparent");created=1756415099;keyid="did:
example:service1#key-1"
Signature: sig1=:ZXhhbXBsZSBvZiBhIHZhbGlkIHNpZ25hdHVyZSBmb
3IgdGhlIHJlc3BvbnNlLi4u=:
(binary PDF data)
```

6.5. MQTT Binding Profile

6.5.1. Purpose

This cluase defines how HSTP messages are published and received over MQTT, including topic structure, encoding, and QoS guarantees.

6.5.2. Topic Schema

HSTP Operation Types and Topic Patterns are shown in Table 4.

TABLE 4: HSTP Operation Types and Topic Patterns

Operation Type	Topic Pattern	Notes
Requests to SWID	`hstp/{target}/req`	Directed requests to a specific SWID.
Responses from SWID	`hstp/{requester}/re`	Directed responses.
Broadcast	`hstp/broadcast`	Used for announcements or discovery.

6.5.3. Envelope Mapping

- MQTT Payload: Encoded message object containing both envelope metadata and payload.
- MQTT Headers/Properties:
  - `content-type` (MQTT 5.0): MUST reflect actual media type.
  - User properties MAY encode envelope fields (mirroring HTTP headers).

6.5.4. Transaction Semantics

- `correlationId` MUST be included for all request-response OPERATIONs.
- QoS:
  - CREATE/UPDATE/DELETE: QoS 1 (at least once)
  - QUERY/EXECUTE_ACTIVITY: QoS 0 or 1 depending on reliability needed
- Clean Sessions: RECOMMENDED for stateless agents

6.5.5. Error Handling

- Error information encoded in payload using structured schema (e.g., JSON Problem Details)
- MQTT Reason Codes MAY be used where applicable
- Unexpected disconnects MAY use MQTT Will messages

6.5.6. Security

- TLS (or DTLS) REQUIRED for all external connections
- Authentication via client certificates or token-based schemes
- Access Control via broker ACLs on topic patterns

6.5.7. Examples

Request (JSON-LD):

- Topic: hstp/did:example:service1/req
- Content-Type: application/ld+json
- Payload:

FIGURE 12

```
{
  "hstpVersion": "1.0.0",
  "operation": "EXECUTE_ACTIVITY",
  "messageId": "...",
  "target": "did:example:service1",
  "requester": "did:example:client1",
  "payload": { "@context": ..., "@type":
"MyActivityRequest", ... }
}
```

Response (JPEG):

FIGURE 13

```
Topic: hstp/did:example:client1/resp
Content-Type: image/jpeg
Payload: (binary image data)
```

Annex A
(normative)

Compliance

A.1. General

- This clause specifies how implementations shall be shown to be compliant with the implementation specification. This clause shall group requirements from the body of the specification into compliance clauses.

NOTENOTEThe clause may contain multiple compliance clauses, e.g., ranging from basic to full compliance.

- This is a normative annex.

A.2. Message Envelope Compliance

Implementations claiming compliance with this specification shall meet the following criteria regarding the HSTP Message Envelope:

1. Implementations shall demonstrate that all transmitted HSTP Message Envelopes include the mandatory fields hstpVersion, messageId, timestamp, operation, payload, and signature.
2. Implementations shall demonstrate that the hstpVersion field conforms to the specified format.
3. Implementations shall demonstrate that the operation field uses an ALL_CAPS verb naming convention as defined in this specification.

NOTENOTEcontinue / refine; This section is under development and will be detailed in a future revision.

A.3. Protocol Binding Compliance

Implementations claiming compliance with the HTTP/2 binding of this specification shall meet the following criteria:

1. Implementations shall demonstrate that the HSTP Message Envelope can be correctly conveyed over HTTP/2.
2. Implementations shall demonstrate that HSTP OPERATIONs map correctly to specified HTTP methods or features within the HTTP/2 binding.

NOTENOTEcontinue / refine; This section is under development and will be detailed in a future revision.

Annex B
(normative)

System Requirements Mapping

B.1. General

This annex:

- provides traceability of the applicable requirements listed in the Spatial Web architecture standard (IEEE P2874) to clauses in the implementation specification where the requirement is implemented.
- can simply contain a table that lists the system requirements in the first column followed by a column listing clauses of the implementation specification.

This is a normative annex.

NOTE This is a work in progress. These requirement codes (e.g. HSTP-1, HSTP-23) refer to keys used to identify the system spec statements, recorded in a shared Notion table (internal to SWF).

B.2. Requirements

B.2.1. HSTP Req. 1: IoT System Interoperability

B.2.1.1. Statement

HSTP shall be interoperable with IoT systems in such a way that the entities are able to exchange information and mutually use the information in an efficient way consistent with IEEE 2413 Architectural Framework for IoT.

B.2.1.2. System requirements coverage

1. The HSTP Implementation Specification defines HSTP as providing a common semantic layer for interoperability, designed to operate over normative transport protocols including HTTP/2, QUIC, JSON-LD, OData, GraphQL, and MQTT. This design facilitates consistent meaning for exchanged information and executed functionalities, enabling independent implementations to interoperate.

2. The implementation specification ensures its transport-agnostic nature and extensibility across multiple serialization formats.
3. It aligns with open standards by supporting common Internet payload formats.
4. The implementation specification outlines the future definition of specific profiles for HTTP, MQTT, and GraphQL, which will assist in integration with existing ecosystems.
5. The adoption strategy for HSTP includes iterative development and defines conformance criteria with testable compliance targets to ensure clarity and adoption for independent implementations.

B.2.1.3. System requirements gaps

1. While the overarching architecture is designed for interoperability and HSTP aims to operate over various normative transport protocols (HTTP/2, QUIC, JSON-LD, OData, GraphQL, MQTT), the HSTP Implementation Specification has not yet fully detailed all required protocol bindings, including exact version numbers and the precise mapping of all HSTP OPERATIONs to specific protocol features within each binding for all listed protocols. Furthermore, Annex A (Compliance), which specifies testable compliance targets crucial for ensuring interoperability of independent implementations, is still marked with "Note: continue / refine". Milestone A on the roadmap explicitly highlights the ongoing work to "Complete Initial Bindings" and "Start of alternative protocols".

B.2.2. HSTP Req. 2: Physical Sensor Data Integration

B.2.2.1. Statement

HSTP shall provide interoperability of observations coming from physical sensors based on considerations of ogc_sensorweb, w3c_wot, one_m2m, iot_vocab.

B.2.2.2. System requirements coverage

1. The HSTP Implementation Specification includes OPERATIONs like `EXECUTE_ACTIVITY` and CRUD OPERATIONs (`CREATE_ENTITY`, `UPDATE_ENTITY`, `DELETE_ENTITY`), which can be applied to sensor data for managing entities within the Universal Domain Graph (UDG).
2. The `payload` field in HSTP messages is designed to carry Hyperspace Modelling Language (HSML) Entities or Activity Schemas, enabling the transport of structured data models suitable for sensor observations.
3. The implementation specification defines a 'Channel' as a Spatial Web Entity to group streams of HSML entities related to an activity, with OPERATIONs such as `PUBLISH_TO_CHANNEL` and `SUBSCRIBE_TO_CHANNEL`.

B.2.2.3. System requirements gaps

1. The HSTP Implementation Specification does not currently detail specific mechanisms or integrations with `ogc_sensorweb`, `w3c_wot`, `one_m2m`, or `iot_vocab` beyond general data OPERATIONs.
2. The `EXECUTE_ACTIVITY` operation needs to become more clear as Activity and Channel semantics come out of the HSML spec.

B.2.3. HSTP Req. 3: Robotics and Actuator Device Support

B.2.3.1. Statement

HSTP shall provide interoperability of robotics and other physical actuator devices.

B.2.3.2. System requirements coverage

1. The HSTP Implementation Specification describes HSTP as an application-layer and transactional protocol that facilitates the execution of functionality and sharing of structured data through defined OPERATIONs.
2. The `EXECUTE_ACTIVITY` operation enables HSTP-compliant systems to execute functionalities, which can encompass actions performed by robotics and actuator devices.
3. HSTP messages encapsulate HSML Activity Schemas, which form the structural basis for protocol commands that can define actions for such devices.

B.2.3.3. System requirements gaps

1. The HSTP Implementation Specification does not provide specific implementation details for robotics and actuator device interoperability beyond the general `EXECUTE_ACTIVITY` operation.

B.2.4. HSTP Req. 4: Entity Discovery Services

B.2.4.1. Statement

HSTP shall enable discovery of physical and virtual entities via discovery services.

B.2.4.2. System requirements coverage

1. The HSTP Implementation Specification notes that HSTP messages include sender's UDG position (SWID, known neighbors, spatial hints) and networked address information, which supports optimized routing and discovery.
2. The implementation specification explicitly lists `DISCOVER_DOMAINS_USING_UDG` and `DISCOVER_DOMAINS_USING_SPACE` OPERATIONs for entity discovery.
3. Optimized routing and discovery are cited as key aspects of HSTP's scalability and performance justification.

B.2.4.3. System requirements gaps

1. Milestone C, "UDG and Discovery," in the HSTP roadmap identifies the need for a "Fully specified protocol for spatial discovery / graph traversal" and "Integration mechanisms with UDG and agent registry".

B.2.5. HSTP Req. 5: Cognitive Computing Requests

B.2.5.1. Statement

HSTP shall enable requests for cognitive computing such as for inference (perception, learning, action selection).

B.2.5.2. System requirements coverage

1. The HSTP Implementation Specification is designed to facilitate the execution of functionality and structured data sharing through defined OPERATIONs, supporting various cognitive computing tasks.
2. The implementation specification's transactional nature and its ability to encapsulate HSML Activity Schemas make it a suitable layer for initiating such requests.
3. The `EXECUTE_ACTIVITY` operation provides a general mechanism for requesting complex computations or inferences.

B.2.5.3. System requirements gaps

1. While HSTP is intended to facilitate cognitive computing requests, the current specification lacks detailed protocol mechanisms and specific implementation guidance for such complex interactions. The EXECUTE_ACTIVITY operation, which is fundamental for "requesting activity execution" for tasks like "inference (perception, learning, action selection)", informed by future clarity on Activities and Channels in the HSML workstream. The general extensibility alone may not sufficiently specify the protocol's role in enabling these nuanced cognitive processes at the PDR stage.

B.2.6. HSTP Req. 6: Machine Learning on Sensor Data

B.2.6.1. Statement

HSTP shall enable machine learning OPERATIONs on sensor data, i.e., observations and measurements, accessible in the Spatial Web.

B.2.6.2. System requirements coverage

1. The HSTP Implementation Specification's capability to execute activities and handle structured data (e.g., sensor observations and measurements) via HSML payloads supports machine learning operations.

2. The EXECUTE_ACTIVITY operation can be utilized to request the execution of activities involving machine learning.

B.2.6.3. System requirements gaps

1. The HSTP Implementation Specification does not yet provide specific protocol details or comprehensive mechanisms for enabling machine learning operations directly on sensor data. The EXECUTE_ACTIVITY operation should be utilized for such operations. Additionally, this gap is compounded by the broader lack of detailed integrations with specific sensor data standards (e.g., ogc_sensorweb, w3c_wot, one_m2m, or iot_vocab) noted under Annex B.2.2.

B.2.7. HSTP Req. 7: Agent Node ML Interoperability

B.2.7.1. Statement

HSTP shall enable machine learning by providing interoperability of agent nodes using information accessible in the Spatial Web.

B.2.7.2. System requirements coverage

1. The HSTP Implementation Specification provides a common semantic layer for interoperability among diverse systems, ensuring consistent meaning of exchanged information.
2. It defines OPERATIONs for managing entities and executing activities, which are integrated with HSML and UDG, facilitating data access for agent nodes.
3. HSTP messages include information about the sender's UDG position, enabling optimized routing and discovery for agent interactions.

B.2.7.3. System requirements gaps

1. While HSTP provides a common semantic layer and mechanisms for agent interactions, the specific protocol details for enabling machine learning interoperability between diverse agent nodes remain largely undefined. As we get clarity from the "agent framework" portion of the spec, we should add necessary detail to core agent-centric OPERATIONs such as REGISTER_ACTIVITY_EXECUTOR, EXECUTE_ACTIVITY, and FIND_ACTIVITY_EXECUTOR, which are essential for coordinating and enabling machine learning tasks across agents.

B.2.8. HSTP Req. 8: AI Node Message Passing

B.2.8.1. Statement

HSTP shall enable message passing between nodes with AI functionality.

B.2.8.2. System requirements coverage

1. The HSTP Implementation Specification describes HSTP as a transactional, application-layer protocol designed for communication between systems.
2. The system specification explicitly requires HSTP to provide interoperability of digital messages between nodes.
3. The implementation specification defines OPERATIONs to route messages and encapsulate HSML Activity Schemas, forming the basis for communication.

B.2.8.3. System requirements gaps

1. No specific gaps directly related to enabling message passing for AI functionality are noted in IEEE P2874, as it is considered a core function of HSTP.

B.2.9. HSTP Req. 9: Multi-Intelligence Agent Communication

B.2.9.1. Statement

HSTP shall provide infrastructure and protocols for AGENTS with diverse intelligences to communicate and interact effectively.

B.2.9.2. System requirements coverage

1. The HSTP Implementation Specification establishes a common semantic layer for interoperability among diverse systems and end-point devices, ensuring consistent meaning of exchanged information and executed functionalities.
2. HSTP OPERATIONs are designed to be deeply integrated with HSML entities like Agents.
3. The system specification explicitly states that HSTP shall provide infrastructure and protocols for Agents with diverse intelligences to communicate and interact effectively.

B.2.9.3. System requirements gaps

1. The HSTP Implementation Specification currently lacks detailed protocols and mechanisms for how AGENTS with “diverse intelligences” are meant to communicate and interact effectively. As we get clarity from the “agent framework” portion of the spec, we should add necessary detail to core agent-centric OPERATIONs such as REGISTER_ACTIVITY_EXECUTOR, EXECUTE_ACTIVITY, and FIND_ACTIVITY_EXECUTOR, which are essential for coordinating and enabling machine learning tasks across agents.

B.2.10. HSTP Req. 10: Skipped as a duplicate of HSTP-9

B.2.10.1. Statement

NOTE This requirement is a duplicate of HSTP-9 in the system specification.

B.2.11. HSTP Req. 11: Reliable Agent Communication Protocols

B.2.11.1. Statement

HSTP shall help ensure communication protocols are reliable and efficient, facilitating information transmission between AGENTs regardless of their underlying implementation or the network over which they communicate.

B.2.11.2. System requirements coverage

1. The HSTP Implementation Specification is designed to operate across varied communication network performance, supporting both high latency/low connectivity and high-bandwidth scenarios.
2. The implementation specification is transport-agnostic and extensible across multiple serialization formats, contributing to efficiency and reliability.
3. Its core purpose as a transactional protocol, facilitating coherent, secure, and interoperable interactions, inherently contributes to reliable information transmission.
4. The implementation specification's design for scalability, enabling increased AGENT interactions without compromising performance, supports efficiency.

B.2.11.3. System requirements gaps

No specific gaps are identified in IEEE P2874 for this requirement.

B.2.12. HSTP Req. 12: Variable Complexity Agent Interactions

B.2.12.1. Statement

HSTP shall support AGENT communication and interactions of varying levels of contextual and syntactic complexity, from simple command and control signals to semantically rich high-dimensional vector representations.

B.2.12.2. System requirements coverage

1. The HSTP Implementation Specification provides a common semantic layer for interoperability, ensuring consistent meaning regardless of the underlying network technology.
2. The `payload` field is specifically designed to carry HSML Entities or Activity Schemas, enabling the transport of rich, structured data models that can represent varying levels of complexity.
3. The use of JSON-LD as the primary format for the Message Envelope and HSML payloads allows for semantically rich data exchange.

B.2.12.3. System requirements gaps

No specific gaps related to supporting varying levels of communication complexity are noted in IEEE P2874.

B.2.13. HSTP Req. 13: Timescale Communication and Rating Framework

B.2.13.1. Statement

HSTP shall enable agents to communicate over necessary timescales for optimal function and integrate an autonomous rating framework to support understanding of agents' capabilities, responsibilities, and interaction behaviors within multi-agent systems.

B.2.13.2. System requirements coverage

1. The system specification explicitly states that HSTP shall enable agents to communicate over necessary timescales and integrate an autonomous rating framework.
2. The "AIS Rating Framework" is identified as a separate compliance target in Annex A of the system specification, indicating its role in addressing this requirement.

B.2.13.3. System requirements gaps

The HSTP Implementation Specification itself does not detail the integration or specification of the autonomous rating framework within HSTP. This is deferred to the separate AIS Rating Framework specification, as implied by the system specification.

B.2.14. HSTP Req. 14: Scalable Agent Interactions

B.2.14.1. Statement

HSTP shall support scalability, enabling increased AGENT interactions without compromising performance, crucial for large-scale OPERATIONs.

B.2.14.2. System requirements coverage

1. The "Scalability and Performance Justification" subsection in the HSTP Implementation Specification explicitly states that HSTP is designed with inherent scalability and performance requirements.
2. The implementation specification confirms that the protocol is built to "support scalability, enabling increased AGENT interactions without compromising performance".

B.2.14.3. System requirements gaps

No specific gaps are identified in IEEE P2874 for this requirement.

B.2.15. HSTP Req. 15: Security and Encryption Mechanisms

B.2.15.1. Statement

HSTP shall incorporate mechanisms for authentication, authorization, and encryption to safeguard interactions.

B.2.15.2. System requirements coverage

1. The HSTP Implementation Specification identifies "Enhancing Security and Trust Models" as a core need, explicitly stating the incorporation of authentication, authorization, and encryption mechanisms.
2. The implementation specification mandates a zero-trust security model, requiring authentication, authorization, and continuous validation for all users and entities, and specifies that HSTP message payloads must be encrypted.
3. The `signature` field is a required component of the Message Envelope, essential for verifying message authenticity and integrity.
4. Authorization to decrypt payloads is controlled by the recipient's policy engine, evaluating credentials and claims.
5. OPERATIONs like `ENCRYPT_PAYLOAD` and `DECRYPT_PAYLOAD` are integral to this security framework.
6. The HTTP/2 binding explicitly requires TLS 1.2 or higher and specifies authentication via verifiable credentials in headers.

B.2.15.3. System requirements gaps

1. The HSTP Implementation Specification has a "Note: Specify the required data type/format (e.g., JSON Web Signature structure) and the signature algorithm(s) to be used" for the `signature` field.
2. For payload encryption, a "Note: Specify the required encryption algorithm(s) and format for the encrypted payload" is noted.
3. A "Note: Detail the technical process by which the recipient's system policies use validated sender credentials... to satisfy the 'gated behind a claim in a verifiable credential' requirement and gain access to the necessary decryption key(s) for the payload" is also noted.
4. Milestone D, "Security & Trust," in the HSTP roadmap aims to finalize the AuthN/AuthZ strategy, encryption model, secure transport, and threat model & risk mitigation notes.

B.2.16. HSTP Req. 16: HSML Entity CRUD OPERATIONs

B.2.16.1. Statement

HSTP shall enable create, read, update and delete on HSML entity relations and attributes.

B.2.16.2. System requirements coverage

1. The HSTP Implementation Specification explicitly defines `CREATE_ENTITY`, `UPDATE_ENTITY`, `DELETE_ENTITY`, and `QUERY_SYNC` (for read) OPERATIONs for CRUD actions on HSML entities.
2. The system specification explicitly states that HSTP shall enable create, read, update, and delete on HSML entity relations and attributes.

B.2.16.3. System requirements gaps

No specific gaps are identified in IEEE P2874 for this requirement.

B.2.17. HSTP Req. 17: HSML Entity Queries

B.2.17.1. Statement

HSTP shall enable queries on HSML ENTITIES.

B.2.17.2. System requirements coverage

1. The `QUERY_SYNC` operation is explicitly listed in the HSTP Implementation Specification for querying entities.
2. `QUERY_STREAM` and `QUERY_QUEUE` are also listed as potential query OPERATIONs in the implementation specification.
3. The system specification implies HSML shall enable various query types, including Bootstrapping/Context, Activity, Hyperspace range, Abstract data type, Graph, Semantic, and Vector queries.

B.2.17.3. System requirements gaps

1. The `QUERY_STREAM` and `QUERY_QUEUE` OPERATIONs in the HSTP Implementation Specification need to be further fleshed out in context of the core HSTP scenarios.

B.2.18. HSTP Req. 18: Location Awareness Support

B.2.18.1. Statement

HSTP shall allow Spatial Web domain architectures to enable location awareness.

B.2.18.2. System requirements coverage

1. The HSTP Implementation Specification notes that HSTP messages include sender's UDG position (SWID, known neighbors, spatial hints) and information to locate networked addresses, supporting spatial context.
2. The `DISCOVER_DOMAINS_USING_SPACE` operation is specifically designed for querying domains based on spatial constraints.
3. The system specification explicitly states that HSTP shall allow Spatial Web domain architectures to enable location awareness.

B.2.18.3. System requirements gaps

No specific gaps are identified in IEEE P2874 for this requirement.

B.2.19. HSTP Req. 19: Physical Equipment Location Discovery

B.2.19.1. Statement

HSTP shall allow Spatial Web domain architectures to enable clients to ascertain the physical location of the distributed computing equipment that is hosting the Spatial Web node that the client is accessing.

B.2.19.2. System requirements coverage

1. The HSTP Implementation Specification is designed to include information for locating networked addresses (TCP/IP or other types).
2. The `addresses` field is an optional part of the message envelope, intended to carry a catalog of publicly listable networked addresses.
3. The system specification explicitly states that HSTP shall allow Spatial Web domain architectures to enable clients to ascertain the physical location of the distributed computing equipment.

B.2.19.3. System requirements gaps

1. The HSTP Implementation Specification has a "Note: IEEE P2874 describe this as a "catalog", implying a structured data type like an array or object, but the specific format is not detailed. This needs to be defined" for the `addresses` field.
2. A "Note: Must be a format that allows for locating networked addresses. The specific structure (e.g., list of strings, map of address types to values) needs specification" is also present for `addresses`.

B.2.20. HSTP Req. 20: Fault Resilience Design

B.2.20.1. Statement

HSTP shall be designed to be resilient to faults: The network of Spatial Web nodes should have only transient loss of function if a single node becomes non-operational.

B.2.20.2. System requirements coverage

1. Statement of Design Principle: Section 1.3.4, "Scalability and Performance Justification," of this HSTP Implementation Specification explicitly states that HSTP is designed to be "resilient to faults," ensuring that the network of Spatial Web nodes experiences only "transient loss of function if a single node becomes non-operational."
2. Related Design Aspects: The design also accounts for operation under "varied communication network performance," including high latency/low connectivity and

high-bandwidth scenarios. This contributes to overall system robustness against network-related disruptions.

B.2.20.3. System requirements gaps

1. While this specification states the design principle of fault resilience, it currently lacks detailed technical mechanisms for how this resilience is achieved and maintained in practice. Specific areas needing further definition include:
    a. Fault Detection and Recovery Protocols: Explicit protocols or mechanisms for detecting node failures and initiating recovery processes are not yet detailed. [This aligns with general technical specification needs for addressing "How will it recover in the event of a failure?" which is currently not detailed in HSTP Implementation Spec WORKING DRAFT]
    b. Data Consistency and Integrity: Further elaboration is needed on specific strategies for ensuring data consistency and integrity across the distributed Universal Domain Graph (UDG) following transient losses or node failures. This includes how potential inconsistencies in relationships or content of nodes are handled.
    c. Prevention of Cascading Failures: Specific measures or design patterns to prevent the failure of one node from adversely affecting other components, services, or systems within the Spatial Web ecosystem are not yet described.
    d. Detailed Operational Procedures: The current draft does not specify operational procedures for data recovery or full solution recovery in fault scenarios. [This also aligns with general technical specification needs for "Operational considerations" on data recovery and solution recovery.]
2. Implementation Details for Zero-Trust: While the overall zero-trust security model is mentioned and crucial for resilience, the detailed implementation of security mechanisms, such as encryption algorithms and the process for decryption key access (as noted in gaps for Annex B.2.15 and Annex B.2.23), are still under development and directly impact the practical resilience against malicious faults or data compromise.
3. Cross-Network Access & Auto-Configuration: Related requirements like "automatic configuration" (Annex B.2.25) and "cross-network node access" (Annex B.2.26), which are currently identified as having gaps in the HSTP Implementation Spec WORKING DRAFT themselves, implicitly impact resilience by enabling dynamic adaptation and reachability in a distributed environment. Their fleshing out will contribute to a more robust fault resilience design.

B.2.21. HSTP Req. 21: Variable Network Performance Support

B.2.21.1. Statement

HSTP shall be designed to operate with varied communication network performance. This includes support for both high latency / low connectivity support and scenarios where bandwidth ranges from hundreds of gigabits per second to several terabits per second (i.e. having latency in the sub-millisecond range).

B.2.21.2. System requirements coverage

1. The "Scalability and Performance Justification" section in the HSTP Implementation Specification cites "Support for Diverse Network Conditions" as a primary justification, explicitly covering high latency/low connectivity and high-bandwidth scenarios.
2. The system specification explicitly states that HSTP shall be designed to operate with varied communication network performance.

B.2.21.3. System requirements gaps

No specific gaps are identified in IEEE P2874 for this requirement.

B.2.22. HSTP Req. 22: Independent Implementation Interoperability

B.2.22.1. Statement

HSTP shall enable independent implementations that are interoperable.

B.2.22.2. System requirements coverage

1. "Conformance and Interoperability" is a critical component of HSTP's adoption strategy, with "Testable compliance targets" defined to ensure independent implementations can interoperate effectively.
2. The HSTP Implementation Specification provides a common semantic layer to ensure consistent meaning, facilitating effective interoperability between independent implementations.
3. HSTP promotes interoperability by adhering to Internet standards and allowing multiple common Internet payload formats.
4. The system specification explicitly states that HSTP shall enable independent implementations that are interoperable.
5. Annex A, "Compliance," in the HSTP Implementation Specification outlines how implementations will be demonstrated to be compliant.

B.2.22.3. System requirements gaps

1. The compliance clauses in Annex A of the HSTP Implementation Specification are marked with "Note: continue / refine".

B.2.23. HSTP Req. 23: Zero-Trust Security Model

B.2.23.1. Statement

HSTP shall implement a zero-trust security model.

B.2.23.2. System requirements coverage

1. The HSTP Implementation Specification explicitly implements a zero-trust security model, requiring authentication, authorization, and continuous validation for all users and entities.
2. Payload decryption access is specifically gated behind a claim in a verifiable credential as part of this zero-trust model.
3. The system specification explicitly states that HSTP shall implement a zero-trust security model.

B.2.23.3. System requirements gaps

1. Similar to HSTP-15, the HSTP Implementation Specification has Notes regarding encryption algorithms and the detailed decryption process.
2. Milestone D, "Security & Trust," in the HSTP roadmap aims to finalize the AuthN/AuthZ strategy, encryption model, secure transport, and threat model & risk mitigation notes.

B.2.24. HSTP Req. 24: Digital Message Interoperability

B.2.24.1. Statement

HSTP shall provide interoperability of digital messages between nodes.

B.2.24.2. System requirements coverage

1. The HSTP Implementation Specification describes HSTP as an application-layer protocol designed to facilitate functionality execution and structured data sharing through defined operations and transactions.
2. It provides a common semantic layer for interoperability, ensuring consistent interpretation of messages.
3. The system specification explicitly states that HSTP shall provide interoperability of digital messages between nodes.

B.2.24.3. System requirements gaps

No specific gaps are identified in IEEE P2874 for this requirement.

B.2.25. HSTP Req. 25: Automatic Node Configuration

B.2.25.1. Statement

HSTP shall provide protocols for automatic configuration that allow Spatial Web nodes to react on the addition and removal of nodes such as devices and networks.

B.2.25.2. System requirements coverage

1. The system specification explicitly states this requirement.

B.2.25.3. System requirements gaps

1. The HSTP Implementation Specification does not detail the specific protocols or mechanisms for automatic configuration.

B.2.26. HSTP Req. 26: Cross-Network Node Access

B.2.26.1. Statement

HSTP shall support accessing Spatial Web nodes in the local network from the outside of the local network (the internet or another local network), considering network address translation.

B.2.26.2. System requirements coverage

1. The system specification explicitly states this requirement.

B.2.26.3. System requirements gaps

1. The HSTP Implementation Specification does not detail the specific mechanisms or support for cross-network node access.

B.2.27. HSTP Req. 27: Centralized and Distributed Computing

B.2.27.1. Statement

HSTP shall work in centralized or distributed computing environments.

B.2.27.2. System requirements coverage

1. The HSTP Implementation Specification explicitly states that HSTP is capable of working seamlessly in both “centralized or distributed computing environments”.
2. The system specification explicitly states that HSTP shall work in centralized or distributed computing environments.

B.2.27.3. System requirements gaps

No specific gaps are identified in IEEE P2874 for this requirement.

B.2.28. HSTP Req. 28: Spatial Web Ecosystem Support

B.2.28.1. Statement

HSTP shall support the implementation of a Spatial Web ecosystem, including support for: Spatial Web client nodes, Spatial Web Servers, Spatial Indices, Spatial Web adapters, Distributed consensus and distributed ledger infrastructure, Spatial rendering servers, and Peer-to-peer servers.

B.2.28.2. System requirements coverage

1. The HSTP Implementation Specification explicitly states that HSTP shall support the implementation of a Spatial Web ecosystem, including all listed components.

B.2.28.3. System requirements gaps

No specific gaps are identified in IEEE P2874 for this requirement.

B.2.29. HSTP Req. 29: HSML Activity Schema Routing

B.2.29.1. Statement

HSTP shall use HSTP OPERATIONS to route and encapsulate HSML Activity Schemas as the structure of the protocol commands which can be sent to HSTP-compliant systems.

B.2.29.2. System requirements coverage

1. The HSTP Implementation Specification states that HSTP messages encapsulate HSML Activity Schemas, forming the structural basis for protocol commands.
2. The `operation` field in the message envelope is designed to dictate the primary purpose and semantics of the message, routing messages and encapsulating HSML Activity Schemas.
3. The system specification explicitly states that HSTP shall use HSTP OPERATIONS to route and encapsulate HSML Activity Schemas as the structure of the protocol commands.

B.2.29.3. System requirements gaps

No specific gaps are identified in IEEE P2874 for this requirement.

B.2.30. HSTP Req. 30: MIME-Type Data Identification

B.2.30.1. Statement

HSTP shall use the specified MIME-type-like identifier of the types of data contained, alongside the data, whether it be text, media, binary, and/or encrypted information.

B.2.30.2. System requirements coverage

1. The HTTP/2 binding table in the HSTP Implementation Specification specifies that the `payload's "Content-type" is declared using the Content-Type header, allowing media types to vary (e.g., JSON-LD, JPEG).
2. A table of "Supported Media Types" explicitly lists `application/json`, `application/ld+json`, binary types (e.g., `image/jpeg`, `application/pdf`), and textual types (`text/plain`, `text/html`).
3. The system specification explicitly states that HSTP shall use the specified MIME-type-like identifier of the types of data contained.

B.2.30.3. System requirements gaps

No specific gaps are identified in IEEE P2874 for this requirement.

B.2.31. HSTP Req. 31: System Capability Declarations

B.2.31.1. Statement

HSTP shall use the specified method of providing declarations that alert communicating systems as to the capabilities of each system via providing a catalog of the HSML Activities they can resolve and perform.

B.2.31.2. System requirements coverage

1. The HSTP Implementation Specification includes `GET_SUPPORTED_OPERATIONS` and `GET_SUPPORTED_ACTIVITIES` OPERATIONs under the "Declare Capabilities / Resolvers" category.
2. The system specification explicitly states that HSTP shall use the specified method of providing declarations of capabilities.

B.2.31.3. System requirements gaps

No specific gaps are identified in IEEE P2874 for this requirement.

B.2.32. HSTP Req. 32: Connection Type Designation

B.2.32.1. Statement

HSTP messages shall include information that designates the types of desired and allowed connections from both the requester and responder.

B.2.32.2. System requirements coverage

1. The `addresses` field is an optional part of the message envelope in the HSTP Implementation Specification, intended to carry a "catalog" of publicly listable networked addresses.
2. The implementation specification notes that HSTP messages can include information allowing systems to "negotiate their preferred or required transport method on a per HSML activity basis".

B.2.32.3. System requirements gaps

1. While the `addresses` field and transport negotiation provide some information, the HSTP Implementation Specification lacks explicit detail on how specific "types of desired and allowed connections" are formally designated. Further specification is needed here.

B.2.33. HSTP Req. 33: UDG Network Address Location

B.2.33.1. Statement

HSTP messages shall include information needed by Distributed UDG System, either spatial domains or SWIDs, to locate the networked addresses, be it TCP/IP addresses or other.

B.2.33.2. System requirements coverage

1. The HSTP Implementation Specification is designed for HSTP messages to include sender's UDG position (SWID, known neighbors, spatial hints) and information to locate networked addresses, aiding optimized routing and discovery by the Distributed UDG System.
2. The `addresses` field in the message envelope specifically provides a "catalog" of publicly listable networked addresses.
3. The `neighbors` field provides a list of publicly-listed neighboring SWIDs within the UDG.
4. The system specification explicitly states that HSTP messages shall include information for the Distributed UDG System to locate networked addresses.

B.2.33.3. System requirements gaps

1. For the `addresses` field, the HSTP Implementation Specification has a "Note: IEEE P2874 describe this as a "catalog", implying a structured data type like an array or object, but the specific format is not detailed. This needs to be defined".
2. A "Note: Must be a format that allows for locating networked addresses. The specific structure (e.g., list of strings, map of address types to values) needs specification" is also present for `addresses`.

B.2.34. HSTP Req. 34: Transport Method Negotiation

B.2.34.1. Statement

HSTP messages shall include information that designates the types of desired and allowed connections from both the requester and responder, enabling HSTP-compliant systems to negotiate their preferred or required transport method on a per HSML activity basis.

B.2.34.2. System requirements coverage

1. The "Scalability and Performance Justification" section of the HSTP Implementation Specification highlights "Flexible Transport Negotiation," stating that HSTP messages enable systems to "negotiate their preferred or required transport method on a per HSML activity basis".
2. The `payload` field in the HSTP Implementation Specification carries HSML Entities or Activity Types, with content dependent on the operation and activity definition.

3. The system specification explicitly states that HSTP messages shall include information for transport method negotiation.

B.2.34.3. System requirements gaps

1. Although HSTP is designed to allow communicating systems to "negotiate their preferred or required transport method on a per HSML activity basis", the current specification lacks explicit details on how these specific "types of desired and allowed connections" are formally designated within HSTP messages, or the precise protocol-level steps for this negotiation. This aligns with the existing gap noted in HSTP Req. 32 regarding the unspecified format and structure of the addresses field, which would be crucial for conveying such connection type information. Therefore, despite the stated capability, the implementation details for this negotiation are incomplete.

B.2.35. HSTP Req. 35: Geospatial Location and Status

B.2.35.1. Statement

HSTP messages shall include the last known geospatial or other spatial locations where such information is available, as well as the system's status and readiness for communication.

B.2.35.2. System requirements coverage

1. The HSTP Implementation Specification notes that HSTP messages include information about the sender's position in the UDG, which can contain spatial hints.
2. The `addresses` field, while general, could potentially include spatial locations for the system.
3. The system specification explicitly states that HSTP messages shall include geospatial or other spatial locations, system status, and readiness for communication.

B.2.35.3. System requirements gaps

1. While the `addresses` field is mentioned, the HSTP Implementation Specification does not explicitly detail how "the system's status and readiness for communication" is conveyed within HSTP messages. This aspect requires further specific definition.

B.2.36. HSTP Req. 36: HSML 1.0 Unconditional Compliance

B.2.36.1. Statement

HSTP shall achieve 'Unconditional Compliance' with HSML 1.0 by using HSML CREDENTIALs, Domains, Spaces, Activities, and Channels to be specified in HSML implementation specification when executing or resolving an HSML Activity within an HSTP message.

B.2.36.2. System requirements coverage

1. HSTP OPERATIONs in the HSTP Implementation Specification are deeply integrated with core Spatial Web entities defined by HSML and managed within the UDG.
2. The `payload` field in HSTP messages is designed to carry HSML Entities or Activity Schemas.
3. The system specification explicitly states that HSTP shall achieve "Unconditional Compliance" with HSML 1.0 using specified HSML entities during Activity execution or resolution within an HSTP message.
4. The HSTP Implementation Specification notes a "crucial dependency on and a need for close collaboration with the HSML team" to ensure alignment between HSTP and HSML.

B.2.36.3. System requirements gaps

1. The HSTP Implementation Specification has a "Note: review terminology capitalization once HSML and UDG specs are finalized".
2. A "Note: Highlight the crucial dependency and collaboration needed with the HSML team. Identify specific questions that need HSML team input, such as how different serialization formats are indicated or managed within the payload" is also noted.

B.2.37. HSTP Req. 37: Protocol Profile Definitions

B.2.37.1. Statement

HSTP shall define profiles of HSTP for HTTP, MQTT, and GraphQL.

B.2.37.2. System requirements coverage

1. The "Profile Definition" section of HSTP's adoption strategy states that it will define specific profiles for common protocols like HTTP, MQTT, and GraphQL.
2. The "Protocol Bindings & Encodings" section of the HSTP Implementation Specification defines the HTTP/2 binding.
3. MQTT and GraphQL are listed as normative transport protocols over which HSTP operates.
4. The system specification explicitly states that HSTP shall define profiles for HTTP, MQTT, and GraphQL.

B.2.37.3. System requirements gaps

1. The HSTP Implementation Specification notes a "Note: Create separate subclauses or reference external documents for the detailed binding specifications for each normative transport protocol (e.g., 6.3 HTTP/2 Binding, 6.4 MQTT Binding, etc.)".
2. "Note" items also exist to "Specify the exact version numbers and link to the normative references for each protocol binding" and "Define how HSTP

OPERATIONs map to specific protocol features (e.g., HTTP methods, GraphQL queries/mutations) within each binding".

3. Milestone A, "Complete Initial Bindings," in the HSTP roadmap includes deliverables like the "MQTT binding section" and the "Start of alternative protocols: GraphQL, OData, Kafka, etc.".

B.2.38. HSTP Req. 38: Distributed Computing Use Cases

B.2.38.1. Statement

HSTP shall implement the use cases in section-distributed-computing-use-cases.

B.2.38.2. System requirements coverage

1. The system specification explicitly states that HSTP shall implement the use cases in Section 7.4, titled "Distributed computing use cases" in the P2874 Table of Contents.
2. Section 4.3, "HSTP OPERATIONs," in the HSTP Implementation Specification lists specific OPERATIONs that align with the use cases outlined in P2874 Section 7.4, which includes sequence diagrams for processes like public & top domain SWID registration and creating child domains.

B.2.38.3. System requirements gaps

1. Several OPERATIONs listed in Section 4.3 of the HSTP Implementation Specification, dependent on other workstreams such as UDG and agent framework, require additional detail before this can be fully realized.

Bibliography

Document contributors

Scott Carroll (lead editor), Kurt Cagle, Bastiaan den Braber, Stephane Fellah, Jacqueline Hynes, George Percivall, Christine Perey, Capm Petersen, Reese Plews, Gabriel René, Lior Saar, Markus Sabadello, Hari Thiruvengada, Ronald Tse

Spatial Web Foundation leadership

Gabriel René	Executive Director / Founder
Dan Mapes	Managing Director / Founder
Bastiaan den Braber	Director of Operations
George Percivall	Distinguished Engineering Fellow
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