Protocol

Knowledge, information, facts about ‘subject’: 4D (where, when, who, state dimensions: categories / profiles). Patterns. Node responses. Index.

Hierarchical contexts query / assertion / reply interfaces. Graph / nested contexts (tree / lists) models. Paths. Node endpoints dynamically allocated into graphs according identity, attributes and contextual comparison dimensional alignments. Registry.

Dialog: ask / assert / reply pattern in context path location. RDF Message based representations. Dialog variables, placeholders. Pattern based subscriptions. Naming.

‘Accounts’ (Consumer / Subscriber context, interactions): backend, user clients and agents consolidated and syndicated dispatch of ‘gestures’ (paths messages plus command statements) translated to any given protocols / IO. Context graph ‘reactive’ dataflow activation.

Flows / scenarios (contexts / roles / state / transitions). Discovery. Subscriptions. Ontology alignment.

Protocol layer (over HTTP):
URI scheme (paths, node patterns, subscriptions HATEOAS HAL / JSONLD browseable applications).
Representations: RDF Message.
Command Statement (via HTTP methods defines how representation messages are handled):

C: Path.
S: Assert.
P: Query.
O: Reply.

Protocol 'semantics' (Discovery, Subscriptions, REST):
Subscription: REST Resource (feed / queue), declaratively stated patterns (Binding).

Nodes. Resources / Patterns. Data (Fact, Event), Information (Kind, Rule), Knowledge (Class, Flow) instance / schema dataflow activation (metamodel reactive IO).

Contexts. Endpoints. Paths. Hierarchical / graph aggregation of node resources identifiers. Resolvable 'patterns' to nodes in hierarchy according context and contents.

Accounts (contexts). Dialogs (interactions). Subscriptions (data). Declaratively stated by 'patterns' (resource templates with 'paths': model, application and domain levels).

Protocol: submit 'paths' to 'paths' (functional semantics). CSPO Statements reified / encoded as 'paths'. Return 'paths', rel discovery by comparison alignment (referrer). CRUD is performed on the requesting side by means of returned results augmenting node metamodel. Intermediate requests augments requested nodes metamodel.

Example: from ‘Peter’ resource in ‘Employment’ (referrer context) to ‘Country’: all Peter’s Countries and the relationship with them (i.e.: countries where Peter has worked in).

CSPO Paths: Patterns. C: Path, S: Assertion, P: Query, O: Answer 'patterns'. Discovery by comparison alignment of obtained resource 'paths' rel with goal 'pattern' (referrer).

Example encoding:

C: Context / Path (instance identifier aggregates SPO into pattern.

SPO: /subjectPath[path]/predicatePath[path]/objectPath[path]

De referenceable resource: aggregated Message (IO).

Domain translation: fulfill (dynamic) template.

Application layer (Message encoded as RDF. Reactive dataflow). Bindings: Client APIs (DOM / JAF / DCI).

Dashboard: actionable domain translation of problem spaces flows.