Ecosystem Toolkit

Technical Specification

Building a Trusted Environment: Event-based Attribute Assurance

Alpha Project artefact

September 2020

FACTERN

OPEN IDENTITY EXCHANGE

White Paper

PROJECT PARTICIPANTS

The Open Identity Exchange (OIX) is a non-profit, technology agnostic, collaborative cross sector membership organisation with the purpose of accelerating the adoption of digital identity services based on open standards. OIX's broad membership and independent nature have seen it develop a significant body of digital identity research, and it is a significant influencer working towards the development of a digital identity market.

The following individuals and organisations participated in the OIX Alpha Project that formed the basis of this Report:

- Factern: Ben Helps, Robbie Fraser
- GDS: Chris Allgrove
- Food Standards Agency: Sid Kalita
- Future Borders: Ross McDonald, Stephen Cowx, Tom Robinson, Paul Evans, Jake Luscombe
- HMRC: Nick Davies, Wayne Robinson
- Idemia: David Rennie, Richard Thompson
- Inrupt: Paul Worrall
- Oxford Semantic Technologies: Peter Crocker
- University of Lincoln: Steve Brewer

OIX and the authors would like to thank all the Alpha Project team members for their considerable contribution to the work.

OXFORD SEMANTIC TECHNOLOGIES

CONTENTS

1. INTRODUCTION

1.1 PURPOSE OF THIS DOCUMENT

The Ecosystem Toolkit aims to make it easier for entities to collaborate by producing and/or consuming tokens of value that relate to real world 'things' of common interest. The Toolkit is comprised of the Technical Specification and the Rules of Engagement[1], which outline the technical and non-technical components respectively.

This document is the Technical Specification: it specifies the technical considerations that must be deployed in order to implement an Expert System. An Expert System can be implemented by any entity that sees value in using that Expert System – for example: to collaborate with other entities also deploying Expert Systems.

Throughout this document, we make reference to the concept of an "ecosystem". At a technical level, an ecosystem is simply a function of the entitlements that have been administered through a network of Expert Systems. The configuration of the entitlements that are deployed through an Expert System implement the patterns of behaviour (either assumed or agreed) that define the way two or more entities can, must or ought to collaborate with one another.

An "ecosystem" indicates that those patterns of behaviour have been formalised under a collaboration framework to which each Participant has explicitly agreed, typically as a multilateral agreement (i.e. when more than two entities have agreed to interact under a common governance framework).

The Rules of Engagement provide a default framework for collaboration between two or more entities, and a template for developing that collaboration framework further to meet

[1] See separate document

the needs of any given “ecosystem”. This document should therefore be read in conjunction with the Rules of Engagement.

Entities that use the Ecosytem Toolkit to collaborate are referred to as Ecosystem Participants (or just Participants). Two or more Participants collaborating together are collectively referred to as an Ecosystem.

1.2 CONVENTIONS USED IN THIS DOCUMENT

Much of our thinking is informed by Systems Thinking. It is our belief that both code and configuration are static or even stagnant constructs. Our goal is to create a system that executes both dynamically and is also highly resilient while doing so.

This has led to us to take a particular exploration approach in our conversations, workshops and design sessions, and we have focused heavily on specifying the minimum set of “responsibilities” that need to be deployed to sustain such a system. We refer to a deployment of the these responsibilities as an Expert System.

We use the C4 model to visualise architecture within this Technical Specification and limit visualisations to level 1 and level 2 (system context and container respectively), so as not to prejudice implementation choices by specifying individual components or code.

1.3 STRUCTURE OF THIS DOCUMENT

Section 2 provides an overview of an Expert System and the minimum set of responsibilities that must be implemented to deploy an Expert System in compliance with this Technical Specification.

Section 3 details the Architecture Constraints that must be factored into the deployment of these responsibilities. These constraints form part of the Technical Specification.

Section 4 details each responsibility in turn, describing the requirements that must be implemented. These requirements form part of the Technical Specification.

Section 5 sets out Architecture Recommendations and aims to educate on best practice implementation. These recommendations are not part of the Technical Specification

Section 6 highlights two responsibilities that do not form part of this version of the Technical Specification, but which represent obvious potential extensions of it. These responsibilities are referred to as the Directory and Reasoning Engine.

Section 7 lists other technical considerations that may be relevant to a given ecosystem / domain of interest / use case but which do not form part of the Technical Specification, and are therefore considered to be separate concerns.

Section 8 provides some more detail and further links on Systems Thinking and C4, as referenced within this Technical Specification.

A separate Glossary of Terms[2] provides a common language across the Ecosystem Toolkit artefacts including this Technical Specification.

[2] See separate document

2. AN EXPERT SYSTEM

An Expert System refers to a deployment of the responsibilities required under this Technical Specification. The individual, organisation or collective that deploys the Expert System is the owning entity or owning legal entity (referred to through this document as Owning Entity) of that Expert System and the Expert System is an Agent for the Owning Entity that deploys it. The Expert System acts on behalf of its Owning Entity in all its interactions.

An entity **may** act as Owning Entity to more than one Expert System. A single Expert System **must not** be Agent for more than one Owning Entity at any one time. An entity **may** relinquish its role as Owning Entity to a given Expert System, allowing another entity to claim the role of Owning Entity.

The Owning Entity configures and instructs the Expert System by using the Exchange to write administrative events which draw on the Core Ontology and are referred to as Core Ontology Events. The Owning Entity executes its business processes and undertakes its business by using the Exchange to read and write events which do not draw on the Core Ontology - these are referred to as Events.

Core Ontology Events configure the basis against which the Expert System operates, in terms of the functionality it affords any given entity (including the Owning Entity) based on their entitlements. Whenever it interacts with other entities, the Expert System **must** operate in line with the configuration at that point in time.

A subset of terms within the Core Ontology are each associated with an Administration Protocol. These terms are referred to as Action Terms. Whenever the Owning Entity writes a Core Ontology Event containing an Action Term, the Expert System **must** implement the Administration Protocol that is associated with that term - i.e. the Expert System does exactly what its Owning Entity has asked it to do, in line with the Protocol set out in the Core Ontology.

Events which do *not* draw on the Core Ontology (i.e. “normal” Events) constitute the facts/assertions pertaining to body(s) of knowledge over which an Ecosystem collaborates.

Events have very deliberately been selected as the single most fundamentally important building block of Expert Systems and therefore ecosystems. With Events we may reason over a sound model of state and a sound model of time.

Expert Systems and ecosystems are not interested in a river of data within which it cannot be determined where one structure ends and another begins. It **must** be possible to associate metadata with Events, where this metadata pertains to provenance, permissions and other critical concerns. The objective is to join precise representations of expertise across the boundaries of organisations rather than asking participants for data dumps.

At times the term “Signal” may be used interchangeably with Event. A Signal represents an Event with carefully qualified and quantified properties or dimensions. These dimensions may include priority, category and assurance level amongst others. Put simply, the intention is to boost the signal and reduce the noise.

2.1 SUMMARY OF REQUIRED RESPONSIBILITIES

The Technical Specification specifies that an Expert System must deploy the following responsibilities:

– Exchange: the gateway that connects the Expert System to the outside world (including other Expert Systems), and through which the Owning Entity configures and instructs the Expert System

– Administration: the instruction set used to provision the Expert System, configure the Owning Entity’s entitlements, instantiate an Ecosystem and specify a knowledge base

– Refinery: the transformation of information (including Administration instructions) into a machine-readable representation in the form of an Event

– Event Store: the persistent storage of Events written to the Expert System

– Registry: a registry of all the ‘things’ that have been declared (i.e. nodes), including the Participants that form part of an Ecosystem and their entitlements

– Ontology: a conceptual model of the domain of interest shared by the Participants of an Ecosystem, expressed in a machine-readable logic-based language

These responsibilities may be implemented to varying levels depending on the requirements of any given Owning Entity and/or Ecosystem. With the exception of Architecture Constraints, implementation details and architectural style are not part of this Technical Specification.

2.2 INTEROPERABILITY BETWEEN EXPERT SYSTEMS

The Technical Specification creates a default standard (represented by an Expert System) for the accumulation and distribution of knowledge and insight in the form of facts/assertions on a machine-to-machine basis.

An Expert System takes information of all forms - including the outcomes of distributed operations and reasoning - as inputs, and 'refines' these inputs into tokens of value represented by Events.

A Owning Entity configures its Expert System to afford access to those Events under its control according to the entitlements the Owning Entity has given to other Participants (each of whom is operating their own Expert System, and may - or may not - afford reciprocal access). This configuration enables the instantiation of an Ecosystem.

3. ARCHITECTURE CONSTRAINTS

Architecture constraints **do** form part of the Technical Specification.

3.1 INFORMATION MUST BE CONVEYED USING IMMUTABLE ASSERTIONS

Ecosystems **do not** transport data in a traditional footprint. It is better to minimise data movement due to well recognised problems around privacy, security, legislation and governance.

Instead, Participants use Events both to reference uniquely identifiable ‘things’, including real world entities and data objects, and to make assertions about those ‘things’, including how they relate to each other. Ecosystems aim to convey these references and assertions, without transporting underlying data about the ‘things’ themselves.

Best practice is to ensure that the information entering any Ecosystem is minimally “hydrated” – i.e. transferring just enough information from the system that acts as an External Feed for the resulting Events to be useful within the Ecosystem itself.

This approach is particularly applicable in the context of data assurance: a Participant may assert a relationship between an information set and an entity (e.g. that the former is representative of the latter) without disclosing underlying data about either.

Other Participants may subsequently re-affirm the relationship, or attest that the state of the entity is unchanged. These signals can be used by other Participants to infer a level of trust about the information set.

Events (i.e. the references and assertions) **must** be immutable, minimally hydrated and **may** be indelible.

3.2 EVENT INFORMATION EXCHANGE MUST BE ECOSYSTEM TOPOLOGY AGNOSTIC

An Expert System **must** be agnostic of the model or mechanism used to exchange information with other Expert Systems. This Technical Specification does not favour any specific topology of ecosystem deployment.

An Expert System may operate independently, with access only afforded to its Owning Entity. Alternatively, an Expert System may connect to other Expert Systems via their own network of point-to-point connections. Equally, Participants within a given Ecosystem may mutually agree to use a hub to facilitate connectivity within their particular use case.

3.3 EXPERT SYSTEMS AND ECOSYSTEMS MUST BE OBSERVABLE AT RUN TIME

In order to measure performance and establish and adjust the kinds of feedback loop that are essential in permitting resilience which in turn helps a system to live long enough to evolve, we **must** be able to observe running systems. A combination of runtime observability and feedback loops moves us towards anti-fragility as a design goal.

Within a system whose primary purpose is to join expertise across a network of collaborating participants the primary facet of observability we **must** concern ourselves with is metrics on the routing of signals. Given that the options for Expert System signal routing are flexible and configurable, there is an element of complexity present. The ability to trace signals is therefore important. It **must** be possible to determine if signals are being delivered successfully.

Routing metrics **may** also provide useful information on reciprocation – a healthy ecosystem demonstrates a balance of contribution versus consumption.

Metrics detailing the types of interaction participating entities (Owning Entities and other Expert Systems) are transacting over **must** be available. This permits the observer to understand the value of contributed signals. If accretion is observable over a signal or there is evidence of an association with other signals for corroboration the observer is somewhat informed as how valuable individual signals are.

This Technical Specification provisions for the design of Expert Systems which may operate as configurable collaborating building blocks for a large set of disparate ecosystems.

The Rules of Engagement artefact specifies a collaboration framework which comprises of several components or dimensions which compose to define how participants will collaborate. Expert Systems record collaboration framework component values at a transactional level. The value to which these components may be set is configurable over a spectrum of possible values in some cases.

In order to optimise these values to ensure healthy and useful collaboration can be the status quo in a given ecosystem, the observer **must** be able to observe at least the metrics outlined in this section.

3.4 API FIRST

Expert Systems **must** be architected to cope with today's rapidly evolving technical landscape, provisioning for a multitude of devices and channels without endlessly duplicating code and services.

Automation is playing an ever more important role in the information economy. Application Programming Interfaces (APIs) are at the heart of this. Expert Systems will take on an increasing role in working on our behalf, making recommendations where they are not permitted to go further. For some use cases, machine-to-machine communication is arguably more important than human-to-machine.

True machine-to-machine communication can only exist where an API is self-describing, negotiable and evolvable. Both forms and links **must** be supported, these are elements of a RESTful flavour of API. This Technical Specification does not prescribe that HATEOAS be implemented against. However, round trips to the server and the 'sub-resource problem' can both be avoided. We are therefore aiming for a 'REST-ish' API flavour.

3.5 RESOURCE REPRESENTATIONS AND CONTENT NEGOTIATION MUST BE SUPPORTED

Every protocol designed to last on the order of decades and keep pace with the evolution of IT supports negotiation, and a flexible and extensible way to represent information resources.

A combination of resource representations and content negotiation ensure Ecosystems will permit a lowering of the barrier to entry for potential Participants, while at the same time enabling specialised bilateral collaborations.

Ecosystems must support extensibility and evolvability.

Ecosystems may offer differing resource states or quality of service depending on the dynamics of collaboration. Equally Participants may choose to send and receive their Events or signals via a preferred format over the wire.

3.6 ENTITLEMENTS AND AFFORDANCES MUST BE DISCOVERABLE

In any Ecosystem, different types of Participants **may** interact with each other indistinguishably for some types of transaction. Consuming Participants **must** understand how they may interact with an Expert System and **must** only be presented with entitlements and affordances they are permitted to act upon.

Since Expert Systems **must** be self-describing, there **must not** be any need for out of band API documentation or other such forms of collateral.

3.7 EXPERT SYSTEM FUNCTIONAL EVOLUTION MUST BE SAFE AND UNSURPRISING

Over time, many different Ecosystems will be instantiated. Within these, many Expert Systems will be collaborating, and around them many organisations will build client software to interact with the Expert Systems for a multitude of reasons.

Expert Systems will inevitably evolve functionally in terms of their protocols and resource representations. Whenever functional evolution is implemented, an Expert System **must not** break the software clients and Expert Systems that interact with it.

The following prime directives **must** be obeyed when functionally evolving Expert Systems:

- do not use versions
- require nothing new
- accept what was accepted before
- yield unsurprising responses

Functional evolution in this instance means all evolution that is within our control for example the addition of new features. Legislative change or security upgrades are examples of concerns which may place the evolution of an Expert System out of our control.

3.8 EXTERNAL FEED INTEGRATION MUST BE TECHNOLOGY AGNOSTIC

External systems push information into an Expert System in order to surface domains and sub-domains of common interest to other Ecosystem Participants. These external systems are referred to as External Feeds, and they will be many and varied.

Ecosystems must support a wide range of technologies from traditional relational database backed RESTful APIs to Queue based systems or distributed ledger platforms.

While this Technical Specification does not prescribe any given level of sophistication pertaining to how External Feeds support audit or provenance over the data they provide, it is acknowledged that for some ecosystems this will be a critical factor.

4. REQUIRED RESPONSIBILITIES

An Expert System **must** deploy all of the responsibilities set out in this section.

4.1 EXCHANGE

The Exchange represents the gateway into a given Expert System, connecting it to its Owning Entity, to other Entities who have permission to interact with it, to other Expert Systems (acting on behalf of other Participants) and to the outside world in general.

In particular, the Exchange provides a feedback loop through which the Owning Entity ‘discovers’ the full functionality of the Expert System. Incremental functionality is revealed as the Owning Entity progresses - or iterates - through the different forms of configuration that constitute Administration. In short, the Expert System operates as a form of state machine.

The Exchange **must** present to the Owning Entity/Participant only that which the Owning Entity/Participant is entitled to at a given point in time, based on the configuration of the Expert System at that point in time.

If the configuration of the Expert System changes such that the Owning Entity/Participant is afforded a greater or lesser scope of action, the Exchange **must** present a set of links to the Owning Entity/Participant that reflects the Owning Entity/Participant’s new entitlements.

The Exchange **must** be deployed as an API, such that the Expert System is addressable on a machine-to-machine basis, either by other Expert Systems or by external systems.

4.2 ADMINISTRATION

The Administration responsibilities enable:

– Provisioning of the Expert System

- Configuration of the entitlements of the Owning Entity
- Access to Expert System functionality
- Instantiation of an Ecosystem
- Creation of a knowledge base

The fulfilment of these responsibilities collectively enables the Owning Entity to collaborate with other Participants in an Ecosystem.

4.2.1 Provisioning the Expert System

Initially, the Expert System must afford an entity the ability to write Core Ontology Events that provision the Expert System as follows:

- Declare their own existence and name themselves with a human readable label
- Claim the role of Owning Entity to the Expert System
- Give the URI of the Expert System's Exchange (and therefore the Expert System) a human readable label or accept an auto-generated one

The Expert System is provisioned to a minimum level necessary when the Events that it stores have configured it as an Agent for a known Owning Entity. Provisioning is an iterative process and may involve evolving the Core Ontology itself via data driven interactions with the Administrative responsibilities.

Once provisioned, the Owning Entity writes information into the Expert System, to be transformed into Events by the Refinery and stored in the Event Store.

4.2.2 Configuration of Owning Entity entitlements

Once the Expert System has been provisioned, it must permit the Owning Entity to write the Core Ontology Events that configure entitlements by specifying a permitted scope of action, based on:

- Filters which define a possible set of Core Ontology Events (similar to a view or query)
- Conditions that can be evaluated as true or false (i.e. a Boolean)
- Scopes comprised of both a Filter and Condition

Until the Owning Entity has configured their own entitlements, the Expert System **must** prohibit the Owning Entity from writing any other Core Ontology Events (beyond those needed to provision the Expert System).

Once the Owning Entity has configured their own entitlements, the Expert System **must** associate the Owning Entity with a default set of four Scopes relating to:

- Access to Expert System functionality
- Instantiate an ecosystem
- Create a knowledge base
- Collaborate with other Participants

The Condition associated with each of these Scopes **must** simply be an assertion by the Owning Entity of the Expert System that they permit themselves to be afforded the ability to write the relevant Core Ontology Events. Each of the four Filters are described in more detail below.

Best practice is for the Exchange to present the Owning Entity with each Filter in turn, prompting the Owning Entity to fulfil the Condition that permits them to access the associated functionality. This default setting ensures that the Owning Entity actively ‘discovers’ functionality (in the form of Scopes) incrementally and that there is an explicit record of the Owning Entity permitting themselves a defined scope of action.

4.2.2.1 Access to Expert System functionality

The only functionality that is afforded to the Owning Entity is the ability to write Events. However, until they have permitted themselves to do so, the Owning Entity is not afforded the ability to write Action Terms that trigger Expert System functionality.

The Filter that gives the Owning Entity access to Expert System functionality is comprised of the Action Terms in the Core Ontology. By writing Core Ontology Events that contain Action Terms, the Owning Entity triggers a set response from the Expert System that must allow the Owning Entity to:

- **Read** Events that they have written: i.e. the Expert System executes an Administration Protocol which presents the Event information
- **Retract** Events that they have written: i.e. the Expert System executes an Administration Protocol which removes the ability to permit an Event to be read

This base functionality is the minimum set required to support collaboration within an Ecosystem. Additional functionality can be added to an Expert System by extending the Action Terms within the Core Ontology, where each Action Term is associated with a defined Administration Protocol.

4.2.2.2 Instantiate an Ecosystem

The Filter to instantiate an Ecosystem includes Core Ontology Events that are used to provision the Expert System. These terms allow the Owning Entity to:

- Declare the existence of other entities with whom they wish to collaborate
- Classify these entities according to their role (e.g. Expert System, Participant, External Feed)
- Label these entities with human readable names
- Associate the identifiers of each entity with authentication, authorisation and provenance mechanisms
- Associate the identifiers of each Expert System with its URI

The use of these terms from the Core Ontology allows the Owning Entity to instantiate an Ecosystem, by identifying and qualifying the members of that Ecosystem. The details of each entity are recorded in the Registry.

The Owning Entity configures the scope of action afforded to each Participant by defining Scopes that relate to the shared domain of interest, and associating these Scopes with the resource representing each Participant. Initially, the default scope of action afforded to each Participant is zero (i.e. they can do nothing until given the entitlement to do something by the Owning Entity).

Every Scope that configures the entitlements of a given Participant **must** be associated with the governance framework under which the scope of action is being afforded. The default governance framework must comply with Rules of Engagement.

4.2.2.3 Create a knowledge base

The Filter that is used to create a knowledge base includes Core Ontology Events that allow the Owning Entity to create an ontology that is relevant for their domain(s) of interest.

– Define new classes of 'things' (i.e. beyond the members of the Ecosystem)
– Define new relationships between 'things'
– Define new labels for 'things' (i.e. properties)

The new terms that are defined by a Owning Entity are referred to as an Ecosystem Ontology, to differentiate them from the Core Ontology. The Owning Entity uses the Ecosystem Ontology to create a knowledge base over which to collaborate with other Participants.

4.2.2.4 Collaborate with other Participants

The fulfilment of these Administrative responsibilities also enables the Owning Entity to populate a data set with data instances (sometimes referred to as 'individuals') that are described using the Ecosystem Ontology:

– Declare the existence of instances of 'things' (creating nodes)
– Classify 'things' according to the relevant Ecosystem Ontology
– Assign relationships from the Ecosystem Ontology between 'things' that have previously been declared

– Assign properties to 'things' that have previously been declared by using labels from the Ecosystem Ontology

When the Owning Entity (or any other Participant) populates a data set, the information is captured as a set of Events in the Event Store. As a default, the Expert System **must** ensure that only those Participants that are entitled to act on these Events (e.g. to read them), or to accrete additional Events to the original Events, are afforded the ability to do so.

The Owning Entity interacts with the Expert System to configure the entitlements of other Participants such that they are permitted to read or retract Events that have already been written, or to write additional Events.

Events are written to a location that is dependant on the topology and configuration of a given set of collaborating Expert Systems. The contextual needs of a participating set of entities and the potential sensitivity of the expertise and signals they wish to share will shape topology and configuration concerns.

The Expert System **must** ensure that only the Event Producer and Rights Owners of an Event (see Refinery below) is able to grant or remove the permissions of other Participants to read or retract that Event.

The Expert System **must** ensure that only its Owning Entity is able to grant or remove the permissions of other Participants to write an Event using that Expert System.

The Expert System **must** reference the governance framework that governs the permissions granted to other Participants to read or retract Events for which they are not the Event Producer. The default governance framework is set out in the Rules of Engagement.

4.3 REFINERY

N.B. this section outlines the inner workings of Expert Systems from a machine to machine perspective. While it is important to implementors of the architecture and infrastructure of Expert Systems it is not relevant to those who are using Expert Systems.

Information **must** be conveyed between Participants using immutable assertions that are uniformly structured as Events. The Refinery converts information entering the Expert System into the standardised structure of an Event.

An Expert System **must** contain the capability to refine 'raw' information into Events. This capability ensures that the barrier to contributing information into an Expert System is as low as possible.

The standardised structure **must** be applied to information at an atomic level. This ensures that the barrier to interoperability between Expert Systems is as low as possible

The provenance of an Event **must** be integrated into its structure. This ensures that every Event (and therefore the assertion which it contains) is trustable, governable and auditable.

An Event **must** be structured as an assertion:

- **Body**: the content of the assertion (i.e. what is being asserted)
- **Header**: the provenance of the assertion (e.g. who is asserting / how are they asserting / when are they asserting)

4.3.1 Body

The body of an Event **must** comprise of at least one three tuple or triple:

- **Subject**: the 'thing' that the assertion is about
- **Predicate**: the property of the subject that is being asserted
- **Object**: the property value (or related 'things') being asserted

The assertion expressed by an Event is the collection of one or more triples, where the meaning of those triples can be entirely arbitrary or nonsensical.

A collection of Events can be thought of as a graph, where 'things' are represented by nodes and the relationship between those nodes and the properties associated with those nodes are represented by vertices.

4.3.2 Header

The header of an Event **must** contain the following elements:

- **Identification mechanism** (e.g. UID, DID or other): the way to identify one Event from all others
- **Timestamp**: the time/date at which the assertion contained in the Event entered into the Expert System (i.e. was 'refined')
- **Event Producer**: the identifier of the entity that generated the assertion being made[3]

The Event header provides information about the basis on which an Event enters into an Ecosystem, describing key aspects of its provenance which govern its treatment within the Ecosystem.

[**Optional - best practice to add in where appropriate, context specific.**]

- **External Feed**: the identifier of the external source used to input the assertion into the Expert System
- **Rights Owners**: the identifier of any legal entity or entities (other than the Event Producer) that exercise rights over access to the assertion[4]

[**Implicit - only needs to be explicit when Events are shared with another Expert System**]

- **Event Provider**: the Expert System managing access to the Event on behalf of the Event Producer (and other Rights Owners) within the Ecosystem[5]

[3] In the context of the OIX Trust Framework, the Event Producer can cover the Identity Provider, Evidence Verifier or Evidence Issuer roles, depending on the assertion being made

[4] In the context of the OIX Trust Framework, the "user" (e.g. the holder of an identity or attribute) may be a Rights Owner. Entitlements can be configured to require consent of the Rights Owner to allow access.

[5] In the context of the OIX Trust Framework, the Event Provider may be equivalent to the Broker role

4.3.3 External Feed requirements

The External Feed connecting to the Expert System via the Exchange **must** have been previously declared to the Registry, and **must** identify itself to the Refinery.

The UUID and Timestamp are generated by the Refinery at run time, and the identity of the Event Provider (i.e. the Expert System of which the Refinery is a part) is known to the Refinery.

As a default, the Refinery **must** set the owner of the External Feed as the Event Producer, and set other Rights Owners to null.

If the Event Producer and Rights Owners are different, the External Feed **must** identify them as entities that have previously been declared to the Registry.

4.4 EVENT STORE

The Event Store provides the functionality to persist Events over time. Events **must** be immutable, minimally hydrated when sourced from an external feed and **may** be indelible.

The Expert System **must** record all of the Administrative activity that occurs as Events. Similarly, interactions between Expert Systems (on behalf of their respective Owning Entities) **must** be recorded as Events.

This ensures that a full audit trail of activity is observable at run time.

4.5 REGISTRY

The Registry enforces compliance to this Technical Specification for the subset of Events that record the creation of nodes and - in particular - the declaration of roles played by Participants within an Ecosystem.

When an entity is classified as an Expert System, its identifier **must** be associated with:

– Credentials, so that it can be authenticated

- A URI, so that it can be uniquely addressed by other machines (i.e. the Exchange)
- A unique human readable name, so that it can be uniquely and usefully referenced by a human
- An Owning Entity, so that the legal entity on whose behalf it acts is knowable

When an entity is classified as an External Feed, its identifier **must** be associated with:

- Credentials, so that it can be authenticated by the Expert System
- An Event Producer, so that the legal entity who operates it is knowable

When an entity is classified as a Owning Entity, Event Producer or Rights Owner its identifier **must** be associated with:

- Credentials, so that it can be authenticated
- A unique human readable name, so that it can uniquely be referenced by a human
- A set of contact points (such as an email address or telephone number), so that it can be addressed by a human

As noted below, the identification of the entities that control the identifiers is considered as a separate concern. The responsibility for discovery and identification of Participants does not fall within this Technical Specification.

A given Ecosystem **may** choose to specify an identification requirement under the governance framework to which they submit, where that governance framework goes beyond the Rules of Engagement.

When an Event Producer declares the existence of any ‘thing’, it creates a node. The node identifier **must** be associated with a mechanism for resolving to or de-referencing the identity of the ‘thing’ that it represents.

The default mechanism will be the contact points associated with the Event Producer (or Owning Entity). Every Event includes the identity of the entity that generated the assertion being made in the role of the Event Producer. The Registry enforces compliance with the requirement to associate entities in the role of Event Producer with contact points.

Albeit clumsy, the default approach to de-referencing is therefore to ask the entity that created the node. However, it is **best practice** for the Event Producer to associate another identifier with the node identifier. This additional identifier uniquely identifies the node within the architecture of the Event Producer / External Feed. Alternatively, Event Producers may associate the node identifier with a URI that can be resolved and de-referenced directly.

A given Ecosystem **may** choose to specify the ability to resolve the identity of ‘things’ as a requirement, so that the knowledge base over which they collaborate can *always* be resolved and de-referenced to information held outside of the ecosystem.

4.6 ONTOLOGY

The Expert System **must** afford the ability to create and edit an Ecosystem Ontology using a logic-based, machine-readable standard description language (such as RDF, RDFS and OWL).

An Ecosystem Ontology describes the conceptual model of the domain of interest over which the Participants are collaborating. Expert Systems operating within an Ecosystem use the Ecosystem Ontology to reason and infer in a consistent way, regardless of deployment.

5 ARCHITECTURE RECOMMENDATIONS

Architecture recommendations **do not** form part of the Technical Specification.

5.1 DATA DRIVEN

When it pertains to the exchange of intelligence between organisations as has been stated previously we will be working with facts and assertions which are the results of business processes being applied to raw data across a network of sites - in other words we will be borrowing due diligence already undertaken by individuals, organisations and collectives. Here we must minimise the transport and storage of and processing over raw data.

In stark contrast while it is not part of this Technical Specification we make a strong recommendation here that implementations of this specification do make use of a data driven approach. It is our view that data should be treated as pure data, not locked away in classes or other abstractions of programming languages. This increases extensibility and lets us evolve systems more easily. Our solutions lend themselves to a more general applicability thus satisfying the aim that our toolkits can be used to instantiate a large set of ecosystems supporting different use-cases.

Taking this one step further we recommend that implementors try to promote metadata to be a first class concern on a level with data. Provenance, audit and characteristics of our events and assertions are as important as the attributes of the assertions themselves and while metadata should not impact equality semantics the ability to process over it as if it were data is invaluable.

Working with data first class makes it significantly easier to manage change and reduces the amount of code we need to create in the first place - therefore making the maintenance of our toolkits and solutions easier in the long term.

We are aiming to minimise the amount of code we need to create and hold on to, while maximising the use we put data to. Our code should be viewed as disposable inventory.

This approach pairs nicely with a bottom up stratified design process - providing a larger amount of re-usability over existing and domain specific primitives.

5.2 API STYLE

The Technical Specification does insist the API First approach **must** make use of links and forms. Beyond this it is recommended that other elements of REST may be adopted as this is well suited to the discoverability that sits at the epicentre of the Exchange. Furthermore, using elements of the REST architectural style we benefit from a level of extensibility and evolvability, as it supports machine-to-machine interactions between Ecosystems. See the example API below for reference.

We recommend that the Exchange presents a base set of integrations with existing API-based standards that allow external systems that interact with the Expert System to use negotiable protocols and resource representations. The base set of integrations can be supplemented over time, so as to facilitate easy interaction between the Expert System and external systems.

6. POTENTIAL EXTENSIONS

6.1 DIRECTORY

The discovery, identity proofing and certification of entities that have deployed an Expert System are separate concerns that do not form part of this Technical Specification.

It is conceivable, however, that such functionality could be delivered in a fully machine-readable way by an Expert System, as part of a Directory responsibility.

6.2 REASONING ENGINES

There is no requirement within the current Technical Specification to execute any form of reasoning within the Expert System. The effort and expertise that each Participant uses to generate the insights that may be shared within a given Ecosystem are assumed to be proprietary to each Participant and therefore opaque to others.

However, there are obvious advantages to extending the functionality of the Expert System such that it reasons directly over the Events that it has access to, on behalf of its Owning Entity. At its simplest, this might take the form of a finite state machine or similar.

7. SEPARATE CONCERNS

There are several technical considerations that may be relevant to a given ecosystem / domain of interest / use case but which are not part of the Technical Specification. These include, but are not limited, to the following:

- Security considerations
- Authentication standards
- Proving claims to identity
- Proving claims to entitlements
- Certifying deployment of this Technical Specification
- Reasoning engines (see Potential Extensions)
- Discovery mechanisms (see Potential Extensions)
- Contracting mechanisms
- Value exchange mechanisms
- Issue resolution mechanisms

Participants are free to implement these considerations in a way that best fit their needs. Non-technical considerations that relate to governance are addressed in the Rules of Engagement.

8. REFERENCES

C4 ARCHITECTURE VISUALISATION

C4 stands for context, containers, components, and code — a set of hierarchical diagrams that are used to describe software architecture at different zoom levels, each of which are useful for different audiences.

Details of the C4 model can be found here. In summary:

– Level 1: a **system context diagram** shows the software system you are building and how it fits into the world in terms of the people who use it and the other software systems it interacts with.
– Level 2: a **container diagram** zooms into the software system, and shows the containers (applications, data stores, microservices, etc.) that make up that software system.
– Level 3: a **component diagram** zooms into an individual container to show the components inside it. These components should map to real abstractions (e.g., a grouping of code) in your codebase.
– Level 4: if you really want or need to, you can zoom into an individual component to show how that component is implemented in **code** (e.g. using UML).

SYSTEMS THINKING

There are plenty of resources online to dig into. The below is a good place to start.

A system isn't just any old collection of things. A system is an interconnected set of elements that is coherently organized in a way that achieves something. If you look at that definition closely for a minute, you can see that a system must consist of three kinds of things: elements, interconnections, and a function or purpose.

Thinking in Systems: A Primer, Donella Meadows