

TRUST Over IP FOUNDATION

ToIP Technical Architecture Specification

Create technical specifications in markdown. Based on the original Spec-Up, extended with Terminology tooling

Trust over IP (ToIP) Technology Architecture Specification

Revision History

- Public Review Draft 02 (PR2) — 02 September 2024

For the most recent version of this specification, please see [this ToIP web page](#) ↗.

Editors

[Daniel Bachenheimer](#) ↗ (Accenture)

[Wenjing Chu](#) ↗ (Futurewei Technologies, Inc)

[Darrell O' Donnel](#) ↗ (Continuum Loop)

[Andor Kesselman](#) ↗ (Benri)

[Antti Kettunen](#) ↗

[Drummond Reed](#) ↗ (Gen)

[Jo Spencer](#) ↗ (460degrees / Sezoo)

Contributors

[Jacques Bikoundou](#)

[Tim Bouma](#) (CIO Strategy Council)

[Kevin Dean](#) (GS1 Global Office)

[Judith Fleenor](#) (Trust Over IP Foundation)

[Sid Haniff](#) (Datasoc)

[Daniel Hardman](#) (Provenant)

[Isaac Henderson](#)

[John Jordan](#) (Province of British Columbia)

- [Vikas Malhotra](#) (WOPLLI Technologies)
- [Christine Martin](#) (Continuum Loop)
- [Sankarshan Mukhopadhyay](#) (Dhiway Networks)
- [Sumabala Nair](#) (IBM)
- [Vinod Panicker](#) (Wipro)
- [Scott Perry](#) (Digital Governance Institute)
- [Vladimir Simjanoski](#) (Blokverse)
- [P. A. Subrahmanyam](#) (CyberKnowledge)
- [Bart Suichies](#)
- [Samuel Smith](#) (ProSapien LLC)
- [Neil Thomson](#) (QueryVision)
- [Allan Thomson](#)
- [Alex Tweeddale](#) (cheqd)
- [Mattia Zago](#) (Monokee)
- [Vlad Zubenko](#) (ETS)

Copyright: 2024 Trust Over IP Foundation

Introduction

The mission of the [Trust over IP \(ToIP\) Foundation](#) is to define an overall architecture for Internet-scale digital trust that combines cryptographic assurance at the machine layers (technology) with human accountability at the business, legal, and social layers (governance). Together these two halves form a complete four-layer architecture for decentralized digital trust infrastructure known as the [ToIP Stack](#). Figure 1 is a conceptual diagram of the basic structure of this “dual stack”:

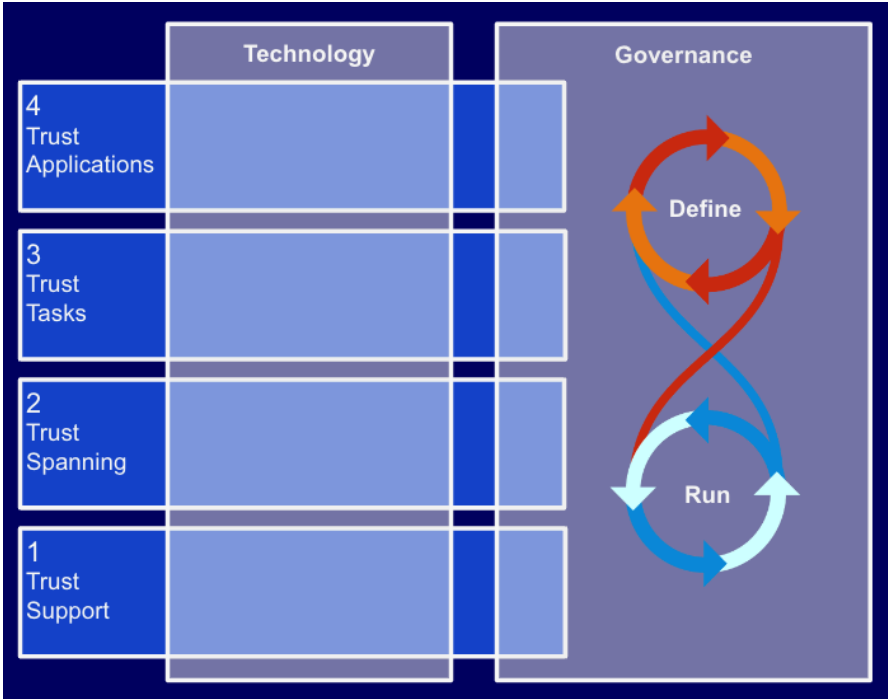


Figure 1: Conceptual diagram of the ToIP stack

The ToIP stack is a model for implementing interoperable [digital trust ecosystems](#). Each ecosystem implements the elements it requires from the [ToIP Technology Stack](#) and publishes an [ecosystem governance framework](#) based on the [ToIP Governance Architecture Specification](#). Figure 2 illustrates the relationship of the ToIP model with an ecosystem instance.

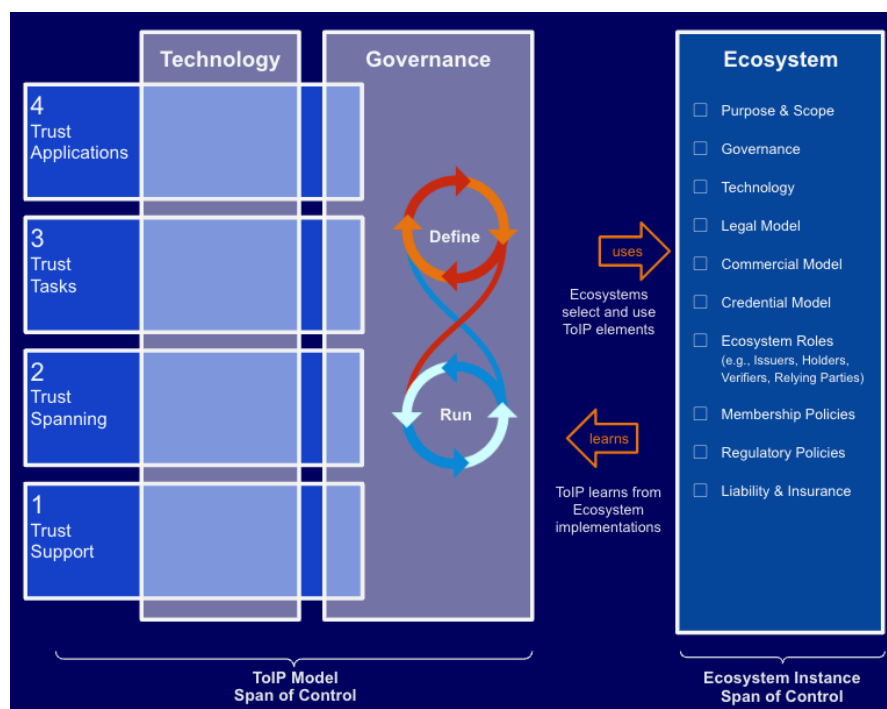


Figure 2: The relationship of the ToIP model with an ecosystem instance

Each ecosystem will have its own specific policies defined by its purpose, context, and objectives. However digital trust ecosystems do not stand alone in the digital world any more than biological ecosystems stand alone in the real world. People, businesses, and even governments operate within and across many different ecosystems. As shown in figure 3, the purpose of the ToIP model is to enable the interconnection and interoperability of many different digital trust ecosystems around the world the same way the Internet enabled the interconnection of many different local data networks around the world.

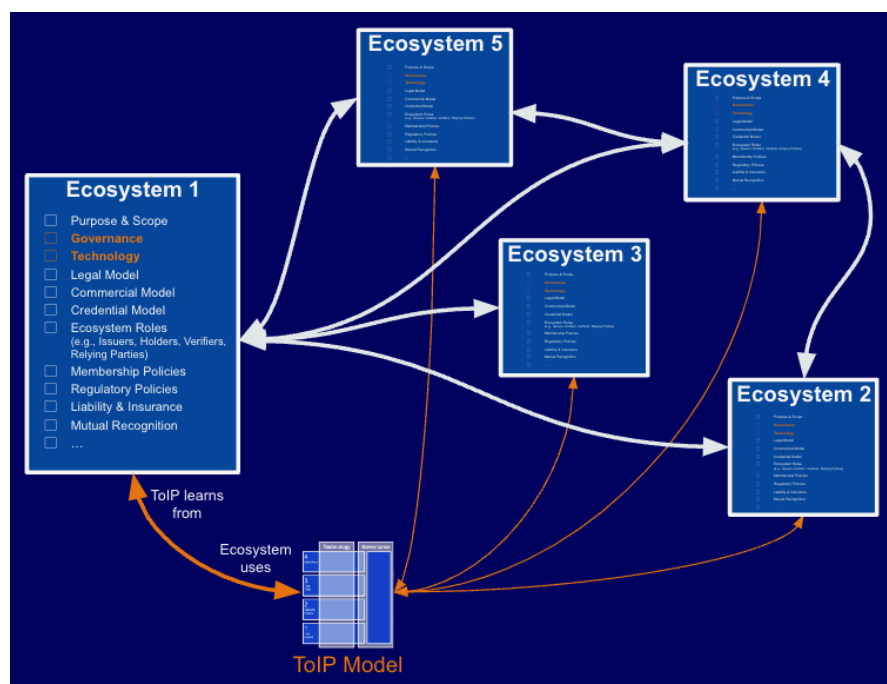


Figure 3: The ToIP model enables interoperability of digital trust ecosystems

The mission of the ToIP Foundation is two fold: 1) develop (or reference) the technical specifications required for the four layers of the ToIP Technology Stack, and 2) develop the [governance framework](#) models and artifacts required for the ToIP Governance Stack. Figure 4 illustrates examples of key components on both sides.

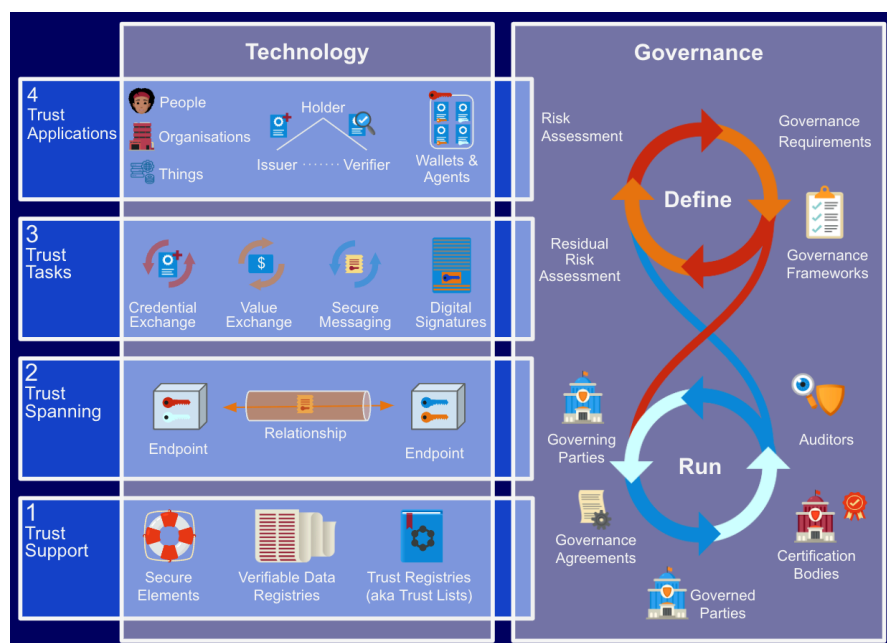


Figure 4: A more detailed view of key components on both sides of the ToIP stack

This document is the normative specification for the high-level architecture of the [ToIP Technology Stack](#) (the left half of Figure 4). It is a deliverable of the [Technology Stack Working Group](#) at the ToIP Foundation. It is recommended to read this document in conjunction with these other documents from the ToIP Foundation in the following order:

1. [Introduction to ToIP](#) is our white paper that provides an overall introduction to the emergence of decentralized digital trust infrastructure. It explains the origin and basic structure of the [ToIP stack](#) together with the mission and activities of the ToIP Foundation.
2. [Evolution of the ToIP Stack](#) is a companion document to this specification that explains the overall process the ToIP Foundation is following in the development of the [ToIP stack](#). It is recommended for anyone seeking to understand how the work of the ToIP Foundation relates to that of adjacent non-profit organizations such as the [Decentralized Identity Foundation](#), the [OpenID Foundation](#), the [Open Wallet Foundation](#), and others including established SDOs such as W3C, IETF, ISO, etc. See [Appendix B](#) for more.
3. [Design Principles for the ToIP Stack](#) is the immediate predecessor to this specification (see the development tracks described in [Section 4](#)). It enumerates the set of design principles informing, guiding, and constraining the design of the [ToIP stack](#). We especially recommend this document for a complete understanding of this specification.

As with all ToIP deliverables, the ToIP Foundation invites your feedback and suggestions. Please contact us via the [ToIP Foundation website](#).

Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL", when appearing in ALL CAPITALS, are to be interpreted as described in [IETF RFC 2119](#).

All other defined terms are linked to their definitions in the [ToIP Glossary](#). This glossary is maintained on behalf of all ToIP Working Groups and the wider decentralized digital trust community by the [ToIP Concepts and Terminology Working Group](#) (CTWG).

Terms especially important to this specification are also explained further inline.

Motivations

This section is informative.

The goal of this specification is to define the overall requirements for a layered system architecture that enables interoperable [trust relationships](#) between any set of endpoints on the Internet. This is directly analogous to the [role](#) the [TCP/IP](#) stack plays in enabling interoperable data exchange between any set of endpoints on the Internet. The design patterns applicable to solving these interoperability challenges, and the motivations for each, are detailed at length in [Design Principles for the ToIP Stack](#).

Whether from the perspective of an implementer, a customer, or a policymaker, there are many benefits to a well-defined layered architecture:

- **Engineering stability.** The abstraction of bundling technologies and policies within distinct layers isolates changes within a layer from interactions and dependencies between layers. The result is a framework more resilient to structural changes than when the layers are not separated.
- **Wide adoption through a principled trust spanning layer.** The [TCP/IP stack](#) features a simple minimalistic protocol (the [Internet Protocol](#) [↗]) as a universal [spanning layer](#). More diverse task-specific protocols are built on top of this spanning layer (e.g., TCP for connection-oriented, UDP for connectionless). In the case of ToIP, a [trust spanning layer](#) based on a well-grounded set of design principles can maximize adoption, interoperability and reachability.
- **Reliability.** A well-defined architecture enables the development of software components and applications that can be trusted to act in predictable, reliable ways—and that can expect other components and applications to do the same.
- **Interoperability and vendor independence.** As with the [TCP/IP stack](#), the Bluetooth stack, the NFC stack, or other [protocol stacks](#), implementations from multiple vendors can and should be interoperable, and customers should be able to switch between them while maintaining standardized functionality. In addition, whenever practical, ToIP should leverage existing technologies provided they are consistent with the [ToIP design principles](#) [↗].
- **Development communities.** A well-designed architecture stack helps spawn a robust, diverse community of developers building solutions whose interoperability depends on a core stack. More development attracts more innovation, more innovation attracts more adoption, producing a network effect benefiting the entire ecosystem.
- **Commoditization.** Standardization of a stack for mass adoption commoditizes it, reducing both the cost of implementation and the time to market. It also frees vendors to focus on their proprietary differentiation in their service offerings.
- **Public policy.** A well-defined architecture with clear and concise [terminology](#) helps policymakers and legal experts define coherent policies and regulations in a manner that serves the needs of society without constraining technical innovation and competition.

Audience, Purpose and Scope

This section is informative.

The audience for this specification is protocol designers, system architects, software developers and product managers who wish to understand, influence, design, develop, or implement interoperable decentralized digital trust infrastructure, services, or applications.

The purpose of this specification is to define a reference architecture for the technology side of the [ToIP stack](#), known formally as the [ToIP Technology Stack](#), including the functions and behaviors required for each of the four layers and the functional and behavioral inter-dependencies between the layers:

- What each layer **must** do.
- What each layer *should* and *may* do.
- What behaviors are expected to support interoperability.
- What interactions each layer supports for other layers.

The goal of these architectural [requirements](#) is to inform subsequent development stages as summarized in Figure 5:

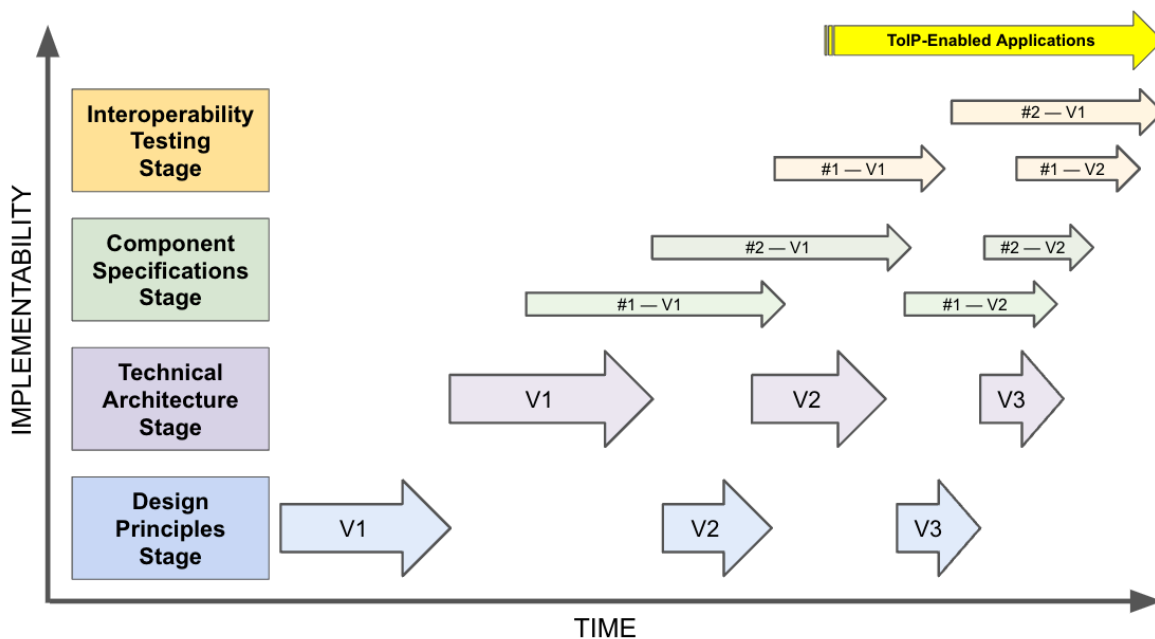


Figure 5: The planned progression of development stages for the ToIP Technology Stack

For more information about the interrelationship and progression of these four stages, please see this page on the ToIP website: [Evolution of the ToIP Stack](#) [↗].

The scope of this specification is limited to the Technical Architecture Stage above, i.e., to defining the normative architectural requirements needed to guide the Component Specification Stage. Success will be achieved if these requirements are sufficient to produce the component specifications needed to implement the architecture and prepare for the Interoperability Testing Stage.

By focusing solely on the Technical Architecture Stage, the following are explicitly out-of-scope:

1. The definition of specific protocols or interfaces at each layer (these will be produced in the Component Specifications Stage).
2. The definition of specific interoperability profiles and test cases—including both vertical and horizontal interoperability—that can be used for commercial-grade test harnesses and testing labs (these will be produced in the Interoperability Testing Stage).
3. The definition of specific intermediary systems or supporting systems for any layer.
4. The definition of specific applications (and their user interfaces) that run on top of the [ToIP stack](#).
5. The definition of ToIP Governance Stack components such as [trust frameworks](#) or [governance frameworks](#) for usage of the ToIP stack within specific [digital trust ecosystems](#).

NOTE: We do not expect all of these additional deliverables, especially the component specifications, to be produced entirely by the ToIP Technology Stack Working Group (or other ToIP Working Groups). Some of these specifications have already been produced—and others are in development—by other standards development organizations (such as the Decentralized Identity Foundation, W3C, IETF, ETSI, and ISO), independent governing authorities, and independent developers.

*NOTE: Due to the public policy implications, the ToIP Foundation is committed to producing a companion document called **ToIP Primer for Policymakers**. This document will guide policymakers, governing authorities, analysts, and other non-technical audiences who need to deeply understand the purpose, uses, and implications of the [ToIP stack](#) but do not need (or want) to dive into technical details.*

Example Use Cases

This section is informative.

Documenting all the example use cases for the [ToIP stack](#) could be as exhaustive as trying to document all the use cases for the TCP/IP stack (i.e., the Internet). Therefore the purpose of this section is simply to list a set of representative use cases that are diverse enough to illustrate the requirements driving ToIP architecture.

They are grouped into five general categories:

1. Discovery, Connection, and Authentication
2. Digital Wallets and Credentials
3. Payments and Value Exchange
4. Secure Messaging, File Sharing, and Digital Signing
5. Cross-Domain Workflows

Discovery, Connection, and Authentication

Use Case	Capsule Description
Generating and Registering a Verifiable Identifier (VID)	The holder of a digital wallet generates a c
Querying a trust registry	A holder or a verifier sends a query to a trust registry to determine if a
Forming a New Relationship (Offline and Online)	A party and counterparty perform an out-of-band introduction (OOBI).
Authenticating to an Existing Relationship	A party who has already established a ToIP relationship with a counter
Exchanging Electronic Business Cards	A party and counterparty use a ToIP relationship to send each other cr

Digital Wallets and Credentials

Use Case
Obtaining and Presenting a Foundational Digital Credential (e.g., Government ID, Birth Certificate, C
Obtaining and Presenting a Functional Digital Credential (e.g., Employment, Diploma, Certification, Provenance, Digital Access
Obtaining and Presenting a Delegated Digital Credential
Issuing and Presenting a Peer-to-Peer Digital Credential
Presenting a Digital Credential Offline
Revoking (and Optionally Replacing) a Digital Credential

Payments & Value Exchange

Use Case	Capsule Description
Paying with a Payment Credential (e.g., credit card, debit card, stored value card)	The holder of a c
Transferring a Digital Currency	The holder of a digital wallet storing the cry
Leaving a Digital Tip	The same use case as above except the val
Receiving an eReceipt	Following the value exchange in any of the i
Buying a Digital Ticket	The same use case as above, except the e-
Issuing a Purchase Order, Sending an Invoice, Remitting Payment	A purchaser issues a purchase order as a c
Bidding in a Digital Auction	The holder of a digital wallet participates as
Placing a Digital Stake	The holder of a digital wallet transfers a dig

Secure Messaging, File Sharing, & Digital Signing

Use Case	Capsule Description
Sending and Receiving Secure Chat Messages (Synchronous)	A sender sends a secure, private cl
Sending and Receiving Secure Mail (Asynchronous)	A sender sends a secure, private rich text message (and option
Sharing Confidential Documents and Files	The controller of a digital resource stored in a repository uses
Signing and Verifying Digital Resources	A party sends one or more counterparties a request to apply a



Scroll to the right

Cross-Domain Workflows

Use Case	Capsule Description
Scheduling and Holding a Secure Video Conference	<p>A meeting host sends a secure, private</p> <p>Once a quorum of attendees have accepted the meeting request, th</p> <p>When the meeting begins, each attendee joins by presenting a proc</p>
Making a Digital Reservation (e.g., Restaurant, Travel)	<p>A traveler who wishes to make a reservation (airplane, rental car, ho</p> <p>To access the reservation, the traveler presents a proof of the acce</p>
Selling a Registered Vehicle	<p>A vehicle owner lists the vehicle for sale in a marketplace using a or</p> <p>Once the owner accepts an offer, the owner sends an escrow reques</p> <p>The buyer accepts delivery, uses the buyer's VID to digitally sign th</p> <p>The original owner sends a copy of the signed certificate of sale to</p>
Orchestrating a Data Supply Chain ("Product Passports")	<p>The original supplier of a product generates a VID for the product a</p> <p>The second supplier in the chain creates a second credential with t</p> <p>The process is repeated until the product reaches the end consum</p>



Scroll to the right

Reference Architecture Overview

This section is informative.

Design Goals

A reference architecture of a complex system is an abstract framework consisting of a list of functional subsystems together with the interfaces and protocols needed to define the potential interactions and dependencies between these systems and/or external systems. This reference architecture provides a logical articulation of these interfaces and protocols which can then be translated into specific component specifications as described in Figure 5.

Such a reference architecture is an exercise in design guided by a set of most significant goals or principles. The overarching goals for the ToIP stack are twofold:

1. Define a general means of establishing trust between any two or more [endpoint systems](#),
2. Achieve universal interoperability among implementations.

These twin objectives led the ToIP Foundation to begin the work with the Design Principles Stage in Figure 25. In 2021, we developed a set of 17 [Design Principles for the ToIP Stack](#) [↗](#) that are the basis for the design choices reflected in this specification. For the full rationale behind each design principle, please see that document.

With regard to the first design goal, establishing trust between [parties](#) requires that each [party](#) develop confidence in the following properties of their relationship:

- 1. **Authenticity:** is the receiver of a communication able to verify that it originated from the sender and has not been tampered with?^[^1]
- 2. **Confidentiality:** is the contents of a communication protected so only authorized [parties](#) have access?
- 3. **Metadata Privacy:** is the metadata of a communication protected so that unauthorized parties can not collect metadata for tracking or correlating with other identifying data?^[^2]

Note that, in some [trust relationships](#), confidentiality and metadata privacy may be optional. Thus our design goal with the [ToIP stack](#) is to achieve these three properties in the order listed.^[^3]

^[^1]: With respect to this design goal, authenticity includes **message integrity**, i.e., a communication is not authentic if it has been tampered with in any way.

^[^2]: In addition to confidentiality and metadata privacy, additional notions of privacy can be built with trust tasks and applications above the trust spanning layer.

^[^3]: Another important property of the architecture is **availability**. This is a concern with the design and implementation of operational deployments of the ToIP stack and should be addressed in the associated operational governance frameworks.

With regard to the second design goal, the ToIP reference architecture shares the same goal of global scalability as the original Internet architecture. This involves several intertwined considerations that overlap and reinforce each other as summarized by the first four [Design Principles for the ToIP Stack](#) ^[^4]:

- 1. The End-to-End Principle
- 2. Connectivity Is Its Own Reward (Universal Interoperability)
- 3. The Hourglass Model
- 4. Decentralization

The Four Layer Pattern

Together these considerations lead to the general *four-layer pattern* of a protocol stack summarized in Table 1.

Layer ##	Generic Hourglass Model Layer Name	ToIP Layer Name
4	Applications	Trust Applications
3	Supported protocols	Trust Tasks
2	Spanning protocol	Trust Spanning
1	Supporting protocols	Trust Support

Table 1: The four layer pattern of protocol stacks that follow the Hourglass Model

The best-known example of this four-layer pattern is the TCP/IP Internet protocol stack, where any number of local area networking protocols at Layer 1 support a single [spanning layer](#) protocol—the IP protocol—at Layer 2. This spanning layer in turn supports multiple higher-level protocols at Layer 3 (e.g., TCP, UDP, HTTP, SMTP) designed to meet the needs of many different applications at Layer 4.

Much of the success of the Internet is attributed to this [Hourglass Model](#) in which the [spanning layer protocol](#) maximizes interoperability by providing a common way for all the higher level layers to communicate with all the lower level layers. This is why the design of the [trust spanning layer](#) should be “as simple as possible but no simpler”. Figure 6 illustrates how this same hourglass design applies to the four ToIP layers.

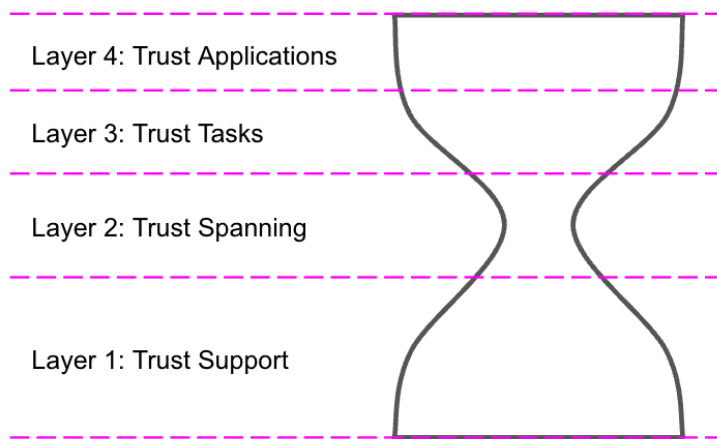


Figure 6: The four layer pattern of the Hourglass Model

For additional overviews of how the ToIP technology stack implements the [Hourglass Model](#), see [Appendix B](#).

High-Level System Architecture

The reference architecture of the ToIP stack provides a generalization of different solutions to trust establishment over the Internet (or over other digital networks). This section introduces the basic concepts, requirements and vocabulary with which to consider: a) each functional component, b) the interface definitions and protocols between these components, and c) interoperability of solutions built upon those components. Subsequent sections will describe these components and protocols in more detail.

At the highest level, ToIP interactions occur between three basic types of interacting systems delineated by *locus of control*.

1. **Endpoint systems** (often simply referred to as **endpoints**): the [ToIP systems](#) between which end-to-end trust is enabled following the End-to-End Principle. See [Section 7.1](#).
2. **Intermediary systems** may be used to assist in the interactions between the [endpoint systems](#). In that context, intermediary systems are involved in the [ToIP Trust Spanning Protocol](#), and may themselves be [endpoint systems](#). Intermediary systems are not a dependency to the [trust relationship](#) between [endpoint systems](#). See [Section 9](#).
3. **Supporting systems** are typically required to support the definition of endpoints and [trust establishment](#) between [endpoint systems](#). Supporting systems that facilitate the authenticity and autonomy of an [endpoint system](#) are termed “privileged” supporting systems, others are “unprivileged” (see [Section 10.1](#)). Supporting systems are not directly involved in the [ToIP Trust Spanning Protocol](#). See [Section 10](#).

The relationships between these systems is shown in Figure 7.

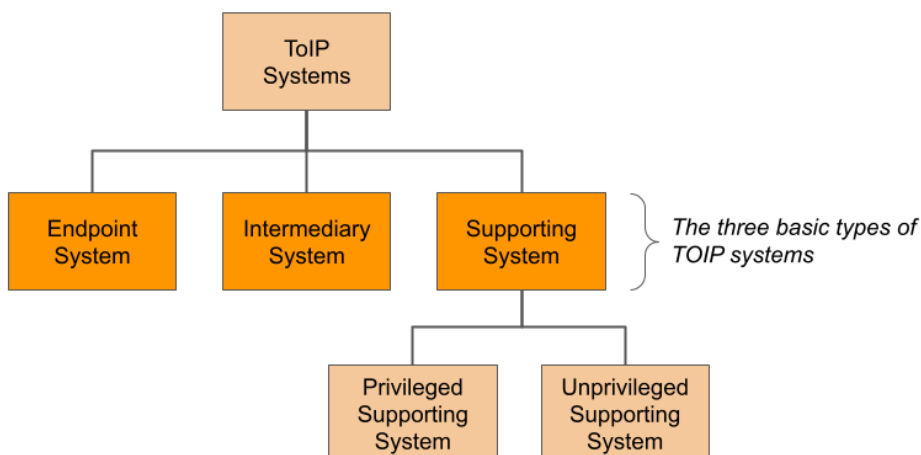


Figure 7: The three basic types of component systems in ToIP architecture

The definition of each system is anchored to its defined (and agreed) locus of control, i.e., who is able to exert control over the operation of that system. Clarity about the locus of control and the dependencies between systems is critical as end-to-end trust is constructed between any two [endpoint systems](#). Each system, whether it is classified as an [endpoint system](#), [intermediary system](#) or [supporting system](#), defines its own locus of control. An [endpoint system](#), for example, may be a tiny IoT device, a personal smartphone, or a large capacity service hosted in a cloud. The terms such as ‘local or remote’ or ‘within, internal or outside’ an [endpoint system](#) should be understood as being with respect to its locus of control rather than physical location. What matters to the architecture is that it exhibits a consistent locus of control and, therefore, consistent interaction protocols with respect to other systems.

These subsystems collaborate with each other through three types of consistent ToIP interactions:

1. [endpoint system](#) to [endpoint system](#)
2. [endpoint system](#) to [supporting systems](#)
3. [endpoint system](#) to [intermediary systems](#)

ToIP [endpoint systems](#) and their interactions follow the 4-layer design pattern described in [Section 6.2](#). As we move up the stack (to Layers 3 and 4), the [roles](#) that may be played by an [endpoint system](#) are often given more context-specific names. For example, at Layer 3, an [endpoint system](#) involved in the [trust task](#) of exchanging [verifiable credentials](#) may be classified as an [issuer](#), [holder](#), or [verifier](#) in that specific interaction context. These higher layer terms are specific to that context and must be consistent with the abstract general terms used in this reference architecture.

Figure 8 shows a high level view of how these three basic types of component systems might interact using the existing infrastructure of the Internet.

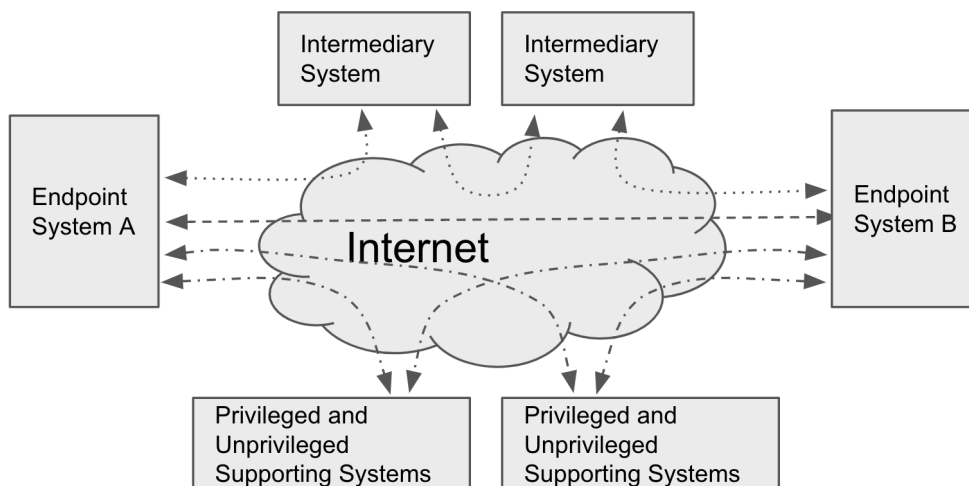


Figure 8: High level view of ToIP consistent system interactions

The normative requirements for each type of subsystem and interaction across the ToIP layers are specified in the following sections.

Verifiable Identifiers

Just as IP addresses are the heart of the Internet [TCP/IP stack](#), [cryptographically verifiable](#) identifiers (VIDs) are the heart of the ToIP stack. Figure 9 illustrates a basic taxonomy of these identifiers.

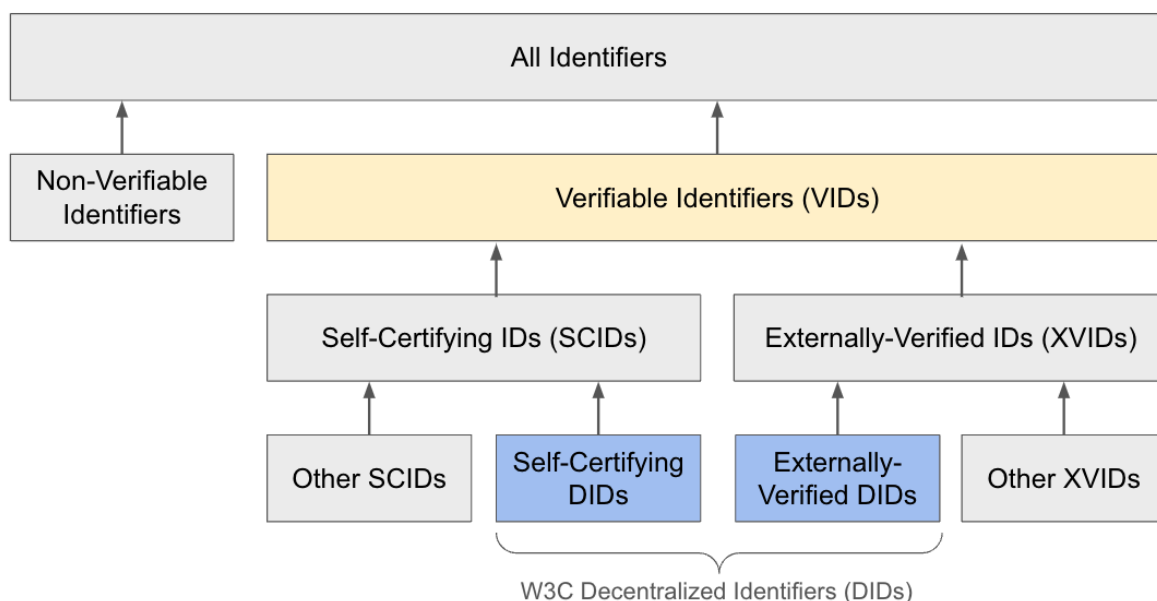


Figure 9: A basic taxonomy of verifiable identifier (VID) types

Design Principle #5 ([Cryptographic Verifiability](#)) states that “messages and data structures exchanged between parties should be verifiable as authentic using standard cryptographic algorithms and protocols”. This requires that [endpoint systems](#) be able to associate, discover and verify the [cryptographic keys](#) associated with the identifier of any other [ToIP endpoint](#). This specification will refer to all such identifiers that meet this basic requirement of cryptographic verifiability as VIDs.

VIDs can be divided into two subclasses as shown in figure 9:

1. **Self-certifying identifiers (SCIDs)** are cryptographically bound to the original key pair used to generate the VID and subsequent key material where key rotation is supported. This means the binding between the VID and the key material can be verified purely using cryptography—without reference to any external system. SCIDs have the advantage of being highly secure (as strong as the cryptographic algorithm used), decentralized (because no external system is required), and portable.
2. **Externally-verified identifiers (XVIDs)** are generated by an interaction between the VID controller’s [agent](#) that has access to the [digital wallet](#) holding the cryptographic key pair and some type of external system or authority, such as a [blockchain](#), [distributed hash table](#), or [certificate authority](#) (CA) that is outside of the autonomous boundary of the [VID controller](#). Verification of an XVID requires the [verifier](#) to interact with that [supporting system](#).

Currently, the most common form of VID is a [decentralized identifier](#) (DID) as defined by the [W3C Decentralized Identifiers (DID) V1.0 Specification] [TODO-REFERENCE](#). In general, DIDs fulfill the requirement of Design Principle #4 ([Decentralization by Design and Default](#)). The W3C DID specification defines a generic syntax for DIDs and a standard data model for a [DID document](#)—the artifact obtained through [DID resolution](#) that contains the [cryptographic keys](#) and [service endpoints](#) bound to the DID. The syntax for a specific type of DID and the process for creating, reading, updating, or deactivating the associated DID document is defined by a [DID method](#).

However, as figure 9 illustrates, there are also SCIDs and XVIDs that are not DIDs. An example of the former is an [autonomic identifier](#) (AID) as defined by the [KERI](#) specifications. An example of the latter is the authority portion of an HTTPS URL. An HTTPS URL relies on two types of [supporting systems](#): 1) a CA for issuance of an X.509 digital certificate (to provide the cryptographic binding with a public key), and 2) a DNS registry (for resolution of the domain name).

Requirements for VIDs are covered in [Section 8.2](#).

Endpoint systems and the Layered Stack

This section is normative.

Endpoint systems

[Endpoint systems](#) represent ToIP systems that are under a [party](#)’s direct control. An [endpoint system](#)’s boundary is delineated by its locus of control. A [party](#) means the entity that is evaluating, relying on, and benefiting from a trust relationship. In other words, a [party](#) is any user of the system without regard to their [role](#) in the system. This represents a contrast with the traditional

identity and access management (IAM) distinct [roles](#) of a user who is making trust assertions and a [relying party](#) who is relying on those assertions to make a [trust decision](#). In a ToIP system, [endpoint systems](#) have a symmetric [peer-to-peer trust relationship](#) in Layer 2 — the [trust spanning layer](#).

[Endpoint systems](#) are autonomous in the sense that a party's locus of control is the whole [endpoint system](#) by definition. This means a potential compromise of other [endpoint systems](#), [intermediary systems](#), or [supporting systems](#) will not directly compromise the integrity of a given [endpoint system](#). Each [endpoint system](#) can be simple or very complex, i.e., it may have many further divided functions and/or services, however in this reference architecture, we shall consider the abstract [endpoint system](#) autonomous. Implementers SHOULD ensure autonomy for [endpoint systems](#) [REQ A.1]

Common examples of [endpoint systems](#) include:

- A mobile phone owned and administered by an individual.
- A merchant's web server (on-premise or in the cloud), administered by the merchant.
- A financial institution's distributed digital services, including certain online services for trust functions, administered by the financial institution.
- An IoT pollution sensor device owned and administered by a city.

Befitting Design Principle #1 ([The End-to-End Principle](#) [↗](#)), [endpoint systems](#) are the ultimate targets of the requirements of ToIP architecture. They are likely to be much larger in number — by several magnitudes — compared to [intermediary systems](#) or [supporting systems](#). They implement most of the functions in ToIP architecture and represent the biggest challenge for interoperability and scalability.

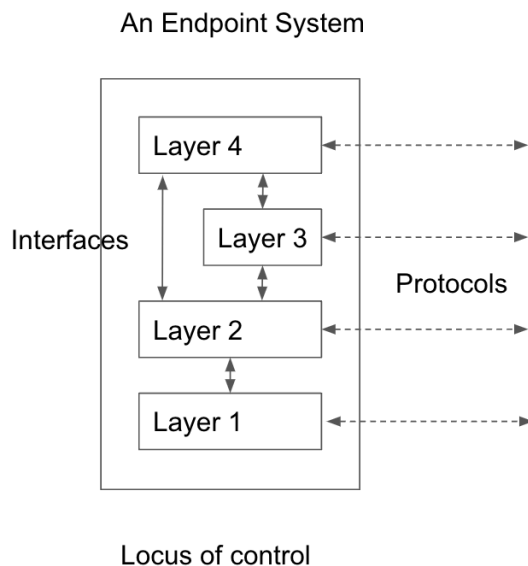


Figure 10: Endpoint system

Within an [endpoint system](#)'s locus of control, a higher layer uses the functions of a lower layer through an **interface**. In ToIP architecture, functions within an [endpoint system](#) are decomposed into layers in a vertical stack where layer boundaries are defined by their corresponding interfaces. In a ToIP [endpoint system](#), the higher layers of the ToIP protocol stack **MUST** communicate with the lower layers via defined interfaces. [REQ A.2]

In addition to the internal layer interfaces implemented by hardware and software resources within the [endpoint system](#)'s locus of control, an [endpoint system](#) may also rely on the services of other [supporting systems](#) that are located outside of the [endpoint system](#)'s locus of control but accessible through the Internet to perform their functions. This type of interaction requires a defined **protocol**.

The distinction between an interface and a protocol is whether the systems communicating over the protocol represent different loci of control. For example, simply distributing the functions within a particular layer over the Internet — such as having some of the functions performed using cloud computing or web services—does not necessarily require a defined protocol if all of the functions are under the same locus of control. However an agreed protocol may be necessary if the communicating systems are under different loci of control. What is essential is delineating who has control over what in order to reason about [trust relationships](#).

The four layer stack within an [endpoint system](#) is defined in the following sections.

Layer 1: Trust Support

If a ToIP [endpoint system](#) includes trust support functions *within its locus of control*, then those functions MUST be included at Layer 1 of the [endpoint system](#). [REQ L1.1] The exact nature of the trust support functions required by any particular [endpoint system](#) may vary significantly depending on the endpoint system's physical manifestation and numerous other design goals (e.g. cost, location, convenience, power usage, reliability and so on). For example the trust support functions required for a full-featured smartphone vs. a cloud server vs. an IoT thermostat may be very different.

Examples of trust support functions designed to specifically support machine-to-machine trust (aka [cryptographic trust](#) or [technical trust](#)):

- Cryptographic hardware modules capable of generating good quality cryptographic key materials.
- Sufficiently secure storage of secrets and cryptographic materials.
- Sufficiently secure computing environment.
- Sufficient communication functions for the intended deployment environment.

NOTE: while this specification generally assumes the Internet as the common networking environment, Internet support is not strictly required. The ToIP stack may be implemented over any communication medium capable of supporting the communication functions.

Examples of trust support functions designed to specifically support human-to-human trust (aka business trust or legal trust) include:

- [Identity binding](#) mechanisms that associate a [natural person](#) to the [endpoint system](#) itself, or to data artifacts on the [endpoint system](#) such as identity [claims](#) or [verifiable credentials](#). One of the strongest identity binding mechanisms is [biometrics](#) — physiological or behavioral characteristics that identify the individual (e.g., facial recognition, fingerprint readers, voice recognition, palm recognition, and so on). Layer 1 would provide the hardware and software support for registering, storing, and processing biometric primitives.
- Hardware-based trust attestation systems, such as those incorporated within [Trusted Platform Modules](#) [↗](#), and other [confidential computing](#) systems that can provide legally valid evidence of the security characteristics of a Layer 1 component.

Diversity of implementations of Layer 1 trust support functions is *intentional* and a key goal of the ToIP stack design.

NOTE: For functional, performance, security, or other reasons, a Layer 1 trust support function implementation may use a remote service outside its locus of control, e.g., a distributed ledger, distributed directory, distributed database, distributed file system, or distributed hash table. These systems are [supporting systems](#) to the Layer 1 implementation; they are not part of Layer 1 itself. See [Section 10.1](#).

Layer 2: Trust Spanning

Layer 2 is the [trust spanning layer](#) of the ToIP stack. In keeping with Design Principle #3 ([The Hourglass Model](#) [↗](#)), this means there is only one requirement for Layer 2: A ToIP [endpoint system](#) MUST communicate with another ToIP [endpoint system](#) using the [ToIP Trust Spanning Protocol](#). [REQ L2.1] No other functions are required.

The requirements for the [ToIP Trust Spanning Protocol](#) are defined in [Section 8](#).

Layer 3: Trust Tasks

Many applications may require more complex trust-building functions than the minimal set offered directly by the [ToIP Trust Spanning Protocol](#). When one of these functions is reusable across multiple contexts that are separated in time, space, or perspective, we call it a [trust task](#). Trust tasks can be standardized as their own higher-level protocols at Layer 3 of the ToIP stack.

A Layer 3 [trust task protocol](#) MUST communicate either over the Layer 2 [ToIP Trust Spanning Protocol](#) or over another Layer 3 [trust task protocol](#) for all communications related to [trust establishment](#) between [endpoint systems](#). [REQ L3.1] This is directly analogous to how TCP and UDP communicate over IP, and how HTTP communicates over TCP. A Layer 3 [trust task](#) MAY use other protocols, but only for other purposes (since short-circuiting Layer 2 when establishing trust with other [endpoint systems](#) would undermine the trust guarantees of the ToIP stack). [REQ L3.2]

Note that because [confidentiality](#) and metadata privacy are optional for the Layer 2 [ToIP Trust Spanning Protocol](#), the following requirement applies: A Layer 3 [trust task protocol](#) intended to communicate private data SHOULD support confidentiality and

MAY also support additional notions of privacy. [REQ L3.3]

There can be as many [trust task protocol](#) as are needed by Layer 4 [trust applications](#). Some examples of [trust tasks](#) include:

- Human authentication (as opposed to cryptographic authentication performed at Layer 2), including [biometric authentication](#)
- Exchanges of [verifiable credentials](#) (e.g., issue, request, offer, present, revoke)
- Provisioning, updating, and verification of [digital identities](#)
- Consent management
- Requesting and signing of digital documents
- Secure messaging
- Secure data sharing
- Digital payment or value exchange in any form
- Digital auctions
- Digital notaries

Layer 4: Trust Applications

Layer 4 is an open-ended application layer for any application that needs to engage in trusted interactions. Layer 4 [trust applications](#) MAY use any number of Layer 3 [\[xref: toip, trust task protocols\]](#). [REQ L4.1].

If a Layer 4 [trust application](#) does not use a Layer 3 [trust task protocol](#) using the Layer 2 [ToIP Trust Spanning Protocol](#). [REQ L4.2]

Layer 4 is the layer where humans “touch” the ToIP stack, so this is where Design Principle #8 ([Trust is Human](#) [↗](#)) and #14 ([Trust and Technology have a Reciprocal Relationship](#) [↗](#)) come into play. The human experience of digital trust is so critical that Layer 4 has one more requirement: A Layer 4 [trust application](#) MUST support any ToIP-defined [trust affordances](#) relevant to that application. [REQ 4.3]

The ToIP Trust Spanning Protocol

This section is normative.

Overview

This section describes the [ToIP Trust Spanning Protocol](#) required at Layer 2 to communicate between any two [endpoint systems](#). The overall protocol operation is shown in Figure 11 below.

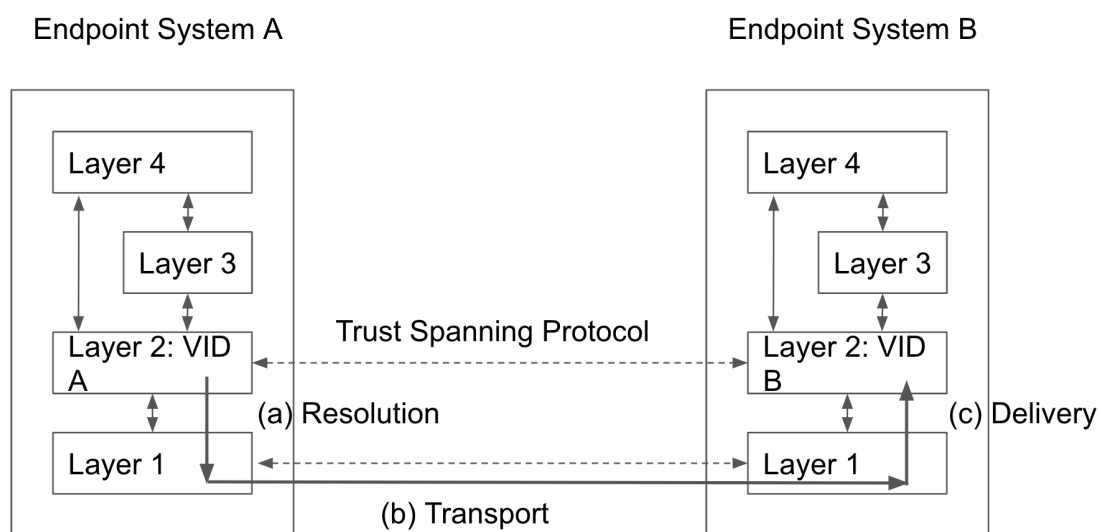


Figure 11: Overview of the ToIP Trust Spanning Protocol

The main function of this protocol is to enable universal end-to-end communication among all [endpoint systems](#) using trusted messages. This architectural choice is based on the following considerations:

- Existing Internet, local network, and mesh network infrastructure already supports universal end-to-end communication through various types of transport mechanisms.
- This form of communication should be able to be implemented on all common types of [endpoint systems](#) with minimum overhead and constraints.
- Messaging-based communication is a least common denominator that provides sufficient foundation for building more complex trust functions at higher layers.

This protocol is designed to be universal in the sense that all [endpoint systems](#), regardless of their form factors or implementation methods, can communicate with each other using messages incorporating standard trust guarantees.

To achieve ubiquity, this protocol should be kept as simple as possible to ease implementation challenges and allow maximum flexibility on all variants of [endpoint systems](#). Thus the requirements in the following sections are not only necessary but sufficient. Strong preference must be given not to add additional functions to this protocol unless they are universally beneficial. Strong preference must also be given to a single common protocol specification for maximum any-to-any interoperability.

A view of the ToIP protocol stack on an [endpoint system](#) is shown in Figure 12. The component specification for the [ToIP Trust Spanning Protocol](#) therefore needs to specify:

1. How to generate and maintain identifiers with the properties described in [Section 6.4](#).
2. The common message format that meets the design goals described in [Section 6.1](#).
3. How lower layer transport protocol(s) can be used to deliver messages between [endpoint systems](#).
4. Any required support from ToIP Layer 1.

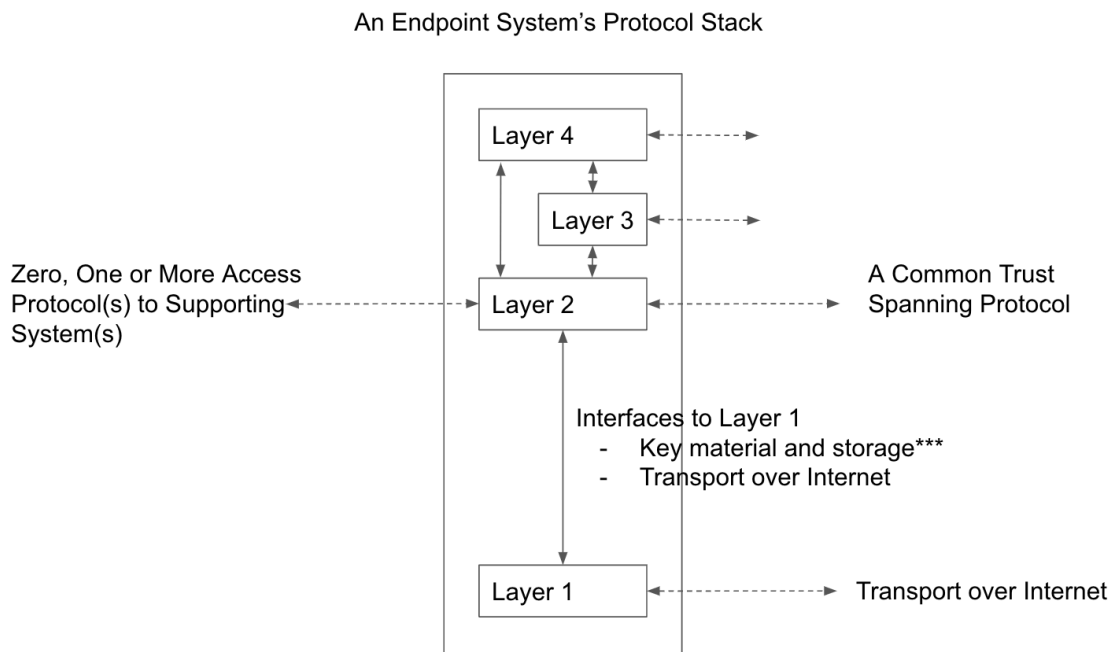


Figure 12: A view of the ToIP protocol stack on an [endpoint system](#)

The following sections enumerate the requirements in each of these four areas.

Identifiers

A key difference between Internet architecture and ToIP architecture is that the former only needed to identify the network endpoints of devices for data communications. The solution was Internet Protocol (IP) addresses: a global addressing scheme for network endpoints independent of any specific local area network.

By contrast, ToIP Layer 2 architecture needs to identify and route messages between the entities participating in [trust relationships](#). While this set of entities may include the devices serving as [endpoint systems](#), it extends beyond the network to include identifiers for [parties](#) — people and organizations using the network to interact and transact.

In order to establish trust in the identifiers used in ToIP architecture — VIDs ([Section 6.4](#)) — regardless of the type of entity to which they are bound, they must meet the following requirements:

- A VID MUST be unique within the context in which it is used for identification. [REQ L2.2] This means VIDS intended to be globally resolvable need to be globally unique; VIDS intended to be locally resolvable need to be locally unique.
- A VID MUST be [cryptographically verifiable](#), i.e., verifiably bound to at least one set of [cryptographic keys](#) discoverable via an associated discovery protocol. [REQ L2.3]
- A VID SHOULD be decentralized, i.e., not require registration with a centralized authority. [REQ L2.4]
- A VID SHOULD be a [self-certifying identifier](#) (SCID), i.e., a fully portable identifier that can be verified using cryptography alone without requiring reference to any external system or [party](#). [REQ L2.5]
- A VID SHOULD support rotation of the associated [cryptographic keys](#) for the lifetime of the identifier. [REQ L2.6]
- A VID MAY also support rotation to an entirely different VID that can be [cryptographically verified](#) to be a synonym of the original VID. [REQ 2.7]
- A VID SHOULD support the ability to: a) associate the identifier with the network address of one or more [ToIP systems](#) that can deliver to one or more [endpoint systems](#) under the locus of control of the [VID controller](#), and b) if desired by that controller, enable that association to be discoverable. [REC L2.8]

Special considerations apply when a VID needs to be provably bound to a specific [party](#), i.e., a person or an organization. Proof of such a binding can be a critical factor in establishing a desired [level of assurance](#) in the identity of that [party](#). Such proof can be accomplished using multiple mechanisms such as:

1. Proof of control of the [cryptographic keys](#) bound to the VID.
2. Proof of control of one or more [verifiable credentials](#) describing the identified [party](#).
3. Proof of one or more [biometric](#) primitives describing the identified [party](#).

Such proofs may require support from one or more Layer 1 trust support functions within the [endpoint system](#), and/or support of one or more [supporting systems](#) outside of the [endpoint system](#), and/or the additional invocation of one or more Layer 3 [trust task protocols](#). These steps are out-of-scope for the Layer 2 [ToIP Trust Spanning Protocol](#).

Different considerations apply when a VID needs to be provably bound to a digital resource, such as a file, photo, or video. This can be accomplished using VIDs that serve as [content-addressable identifiers](#) or [self-addressing identifiers](#) (SAIDs) that are derived from a [cryptographic hash](#) of the subject resource.

Messages

Messages are the lingua franca of the [ToIP Trust Spanning Protocol](#). To achieve the design goals in [Section 6.1](#), the following requirements must be met:

The [ToIP Trust Spanning Protocol](#) specification MUST define how to construct and format messages that are [cryptographically verifiable](#) to have the following three properties:

- Authenticity: the message was sent from a sender who has control over the source VID and the contents of the message transmitted by the sender are received by the recipient who has control over the destination VID without modification.
- Confidentiality: the contents of the message are only accessible by authorized [parties](#).
- Metadata Privacy: the metadata related to the message and its transport and delivery is not exposed to unauthorized parties which may use it for tracking or unwanted correlation with other identifying data. [REC L2.9]

In a ToIP [endpoint system](#), an implementation of the [ToIP Trust Spanning Protocol](#) MUST support [authenticity](#). [REQ L2.10]

In a ToIP [endpoint system](#), an implementation of the [ToIP Trust Spanning Protocol](#) MAY support [confidentiality](#) and metadata privacy. [REQ L2.11]

The [ToIP Trust Spanning Protocol](#) MUST enable the composition of higher-level [trust tasks](#). [REQ 2.12] Examples of such features include discovery, threading, timeouts, ACKs, and attachments. However this requirement must be balanced with the requirement to only add additional functions to this protocol if they are universally beneficial.

The [ToIP Trust Spanning Protocol](#) MUST support extensible message schema. [REQ 2.13] This enables different [trust task protocols](#) to be constructed without changing the base format.

Routing

Routing of a message from a sender to a receiver proceeds in three steps as shown in Figure 8:

1. **Address resolution** takes the VID of the receiver and resolves it to: a) the network address of an [endpoint system](#) for the receiver that supports the desired Layer 1 transport mechanism, and b) the associated [cryptographic keys](#). For example, if the VID is a DID and the desired transport is HTTP, then a DID resolver resolves the DID following the associated DID method to retrieve the DID document. It then selects: a) the service type associated with the [ToIP Trust Spanning Protocol](#) and extracts an HTTP URL to which a connection can be made to deliver the message to the other [endpoint system](#), and b) the required [cryptographic keys](#).
2. **Transport** is the Layer 1 mechanism to send the message to the [endpoint system](#) of the receiver or to an [intermediary system](#) which can eventually deliver the message. In the above example, HTTP is the transport. Over the Internet, any transport layer protocol may be a suitable transport. Other contexts may use other transports, e.g. Bluetooth, QR code, or a publish-subscribe messaging system.
3. **Delivery** is the final step of delivering the message to Layer 2 of the receiver's [endpoint system](#). This step may include a sub-step for an [intermediary system](#) (Section 9) to deliver the message to the [endpoint system](#), and a second sub-step for the [endpoint system](#)'s Layer 1 transport to deliver the message to the Layer 2 interface.

These steps lead to the following requirements:

The [ToIP Trust Spanning Protocol](#) MUST support resolution of VIDs to: a) the network addresses of receiving [endpoint systems](#), and b) any required [cryptographic keys](#). [REC 2.14]

The [ToIP Trust Spanning Protocol](#) MUST support transport of messages via ToIP Layer 1 interfaces. [REC 2.15]

The [ToIP Trust Spanning Protocol](#) MUST support delivery of messages to the Layer 2 interface of the [endpoint system](#) of the ultimate receiver of the message. [REC 2.16]

The [ToIP Trust Spanning Protocol](#) MUST support the option to deliver messages via [intermediary systems](#). [REC 2.17]

The [ToIP Trust Spanning Protocol](#) MUST support [confidentiality](#) with regard to the metadata required for message routing. [REC 2.18]

Interface to Layer 1

Given these requirements for the [ToIP Trust Spanning Protocol](#) at Layer 2, the trust support function interfaces at Layer 1 should only need to include the following. Note that Layer 3 [trust tasks](#) or Layer 4 [trust applications](#) may also need to call these interfaces directly.

1. **Key Management System (KMS)** is the interface for generating cryptographic quality keys, random numbers, or other values required by the cryptographic primitives used by the protocol.
2. **Secure storage** is the interface through which Layer 2 can create, read, write, and delete confidential or secret data.
3. **Transport** consists of one primitive via which the sender's Layer 2 implementation can submit a message for transmission and another primitive through which the receiver's Layer 1 implementation can deliver a message up to Layer 2.
4. **User binding** is the interface via which a Layer 2 implementation can request and verify a [biometric](#) or other [authentication](#) information from a user.

Intermediary Systems

This section is normative.

[Intermediary systems](#) are mediators for facilitating the [ToIP Trust Spanning Protocol](#). Since the Internet itself is routable as long as a VID can be resolved to a unique IP address, [intermediary systems](#) are not absolutely required. However they can be very beneficial in other aspects.

Examples of useful [intermediary systems](#) include:

- **Store and forward intermediaries.** Some [endpoint systems](#) are not always connected to the Internet (e.g. smartphones). As with email, effective communication between an [endpoint system](#) and a smartphone or similar device could use an [intermediary system](#) to receive and store the messages while the device is not connected. Messages are subsequently delivered to the eventual receiver when the device becomes connected again.
- **Multi-device intermediaries.** Individuals who use multiple computing devices (e.g., smartphone, laptop, smart watch, smart car, etc.) may choose to use an intermediary to simplify configuration and management of these devices as well as

routing of messages to the appropriate device.

- **Anonymizing intermediaries.** When confidentiality is desired in the routing of messages, intermediaries designed for this purpose can obfuscate the message routing path.
- **Auditing intermediaries.** When auditing of message traffic is required for regulatory or compliance reasons, intermediaries designed for this purpose can maintain the necessary audit logs.

[Intermediary systems](#) differ from [supporting systems](#) because they reside between [endpoint systems](#) and are visible to the [endpoint systems](#).

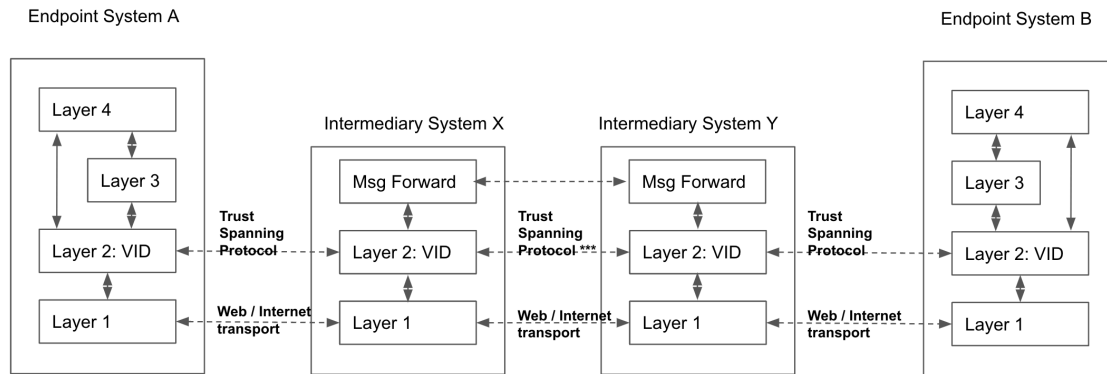


Figure 13: The role of intermediary systems

In Figure 13, end-to-end communication between [endpoint systems](#) A and B are routed through [intermediary systems](#) X and Y. In this case, all systems implement the Layer 2 protocol as described in [Section 8](#). Routing uses “nested envelopes” as follows:

1. [Endpoint system](#) A prepares a message for [endpoint system](#) B and puts it in an inner message envelope addressed to [endpoint system](#) B.
2. [Endpoint system](#) A places the inner message envelope inside an outer message envelope addressed to [intermediary system](#) X.
3. [Endpoint system](#) A delivers the outer message envelope to [intermediary system](#) X.
4. [Intermediary system](#) X removes the outer message envelope and replaces it with a new outer message envelope addressed to the next hop: [intermediary system](#) Y.
5. [Intermediary system](#) X delivers the new outer message envelope to [intermediary system](#) Y.
6. [Intermediary system](#) Y removes the outer message envelope.
7. [Intermediary system](#) Y delivers the inner message envelope to [endpoint system](#) B.

This pattern casts one requirement for the use of [intermediary systems](#):

A ToIP [intermediary system](#) SHOULD be able to perform the functions of a ToIP [endpoint system](#) for the purpose of routing enveloped messages using the [ToIP Trust Spanning Protocol](#). [REC A.3]

Supporting Systems

This section is normative.

Overview

An [endpoint system](#) may utilize services from any number of [supporting systems](#), either privileged or unprivileged, over the Internet or other networks.

- **Privileged supporting systems** are integral to the dependent [endpoint system](#)’s autonomy and authenticity. A privileged [supporting system](#) must implement strongly qualified trust mechanisms in order to play this [role](#). Such trust mechanisms can be a combination of technology (e.g. algorithmic) and governance policies (see [Section 12](#)). For example, a blockchain is a privileged supporting system for a [DID method](#) whose root of trust is the blockchain.
- **Unprivileged supporting systems** are [supporting systems](#) that are not required to support an [endpoint system](#)’s autonomy and authenticity. For example, a website that serves as a non-exclusive convenient discovery (e.g. advertising or search) mechanism for public VIDs is unprivileged.

Each type of [supporting system](#) may have a service access protocol standardized for the type of service it offers. There may be many such services with many different protocols. One [endpoint system](#) may utilize one set of [supporting systems](#) while another [endpoint system](#) may use a different set of [supporting systems](#). This difference in the types of [supporting systems](#) used does not impede the two [endpoint systems](#) in interoperating through the Layer 2 [ToIP Trust Spanning Protocol](#). Therefore, standardization across different services is not required.

An example of a common protocol stack for this purpose is a defined Web Service running on top of HTTPS. However, many types of protocols may be used for different [supporting systems](#).

The ToIP protocol stack in an [endpoint system](#) MAY use the services of a [supporting system](#) at any layer. [REC A.4] Such design decisions can be made layer by layer to optimize the functions performed in each layer.

The following sections illustrated the layered interaction between [endpoint systems](#) and [supporting systems](#) using examples of known implementations.

Example 1 - A DID Method

A [DID method](#) may be implemented based on a distributed ledger, e.g. Hyperledger Indy. An [endpoint system](#), in this example, may be implemented using a Hyperledger Aries agent software module running on either a mobile device or a cloud platform. The Indy ledger is a privileged [supporting system](#) and the Aries agent implements layer 2 and layer 3 of the [endpoint system](#) stack. Such a design pattern is illustrated in Figure 14.

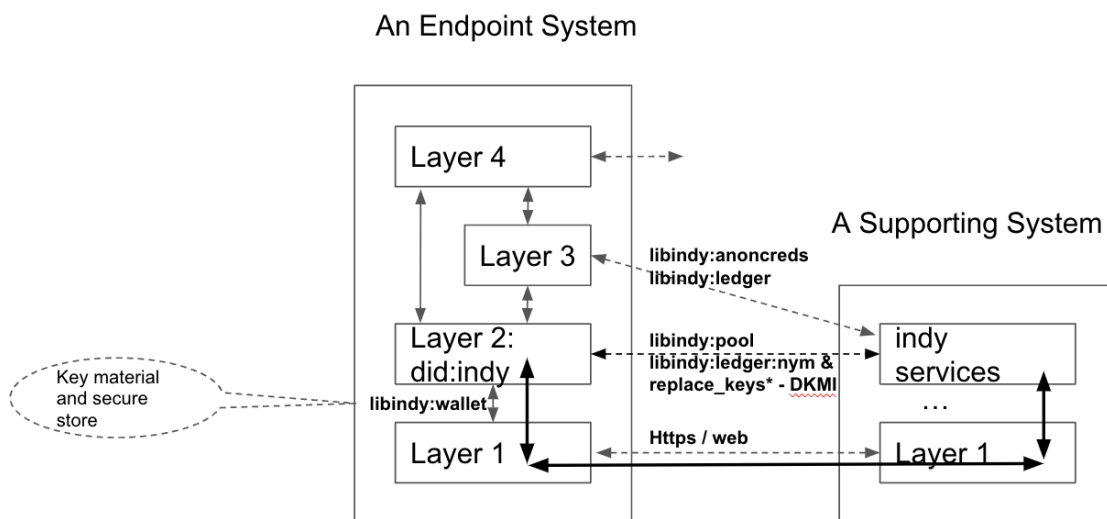


Figure 14: Example of Hyperledger Indy as a Supporting System

A Layer 2 implementation must implement both [DID resolution](#) and the [ToIP Trust Spanning Protocol](#). To implement [DID resolution](#) in this example, the Aries agent uses a local service (i.e. within its locus of control), i.e. a [digital wallet](#), which relies on, eventually, a KMS function and a secure storage function within the [endpoint system](#). It also uses a remote service (i.e. outside of its locus of control) — the Indy blockchain — via web service APIs built on top of HTTPS and other web protocols. This remote service protocol consists of three components in the case of Aries-Indy: pool API, anoncred API, and payment API. The web service eventually relies on the Internet Protocol stack for routing, transport and delivery. Collectively, it is a complete [endpoint system](#)-to-[supporting system](#) protocol that in this case runs over the web.

Example 2 - A KERI Witness

[KERI](#) [\[7\]](#) offers another example in this design pattern. In KERI, the [endpoint system](#) identifier is either an AID or a did:keri method. A layer 2 implementation will need certain key material and secure storage from the lower layer as well. In addition, it requires additional services that are outside of the [endpoint system](#)'s locus of control boundary. The [KERI Witness Pool](#) [\[7\]](#) is an example of such a supporting service as shown in Figure 15. Another example is [KERI Watcher Pool](#) [\[7\]](#).

These supporting services differ from local dependencies (e.g. secure storage) because they are outside of an [endpoint system](#)'s locus of control. The access protocol to such supporting services is also different from the [ToIP Trust Spanning Protocol](#) as it is a protocol between different types of parties and has a different protocol stack.

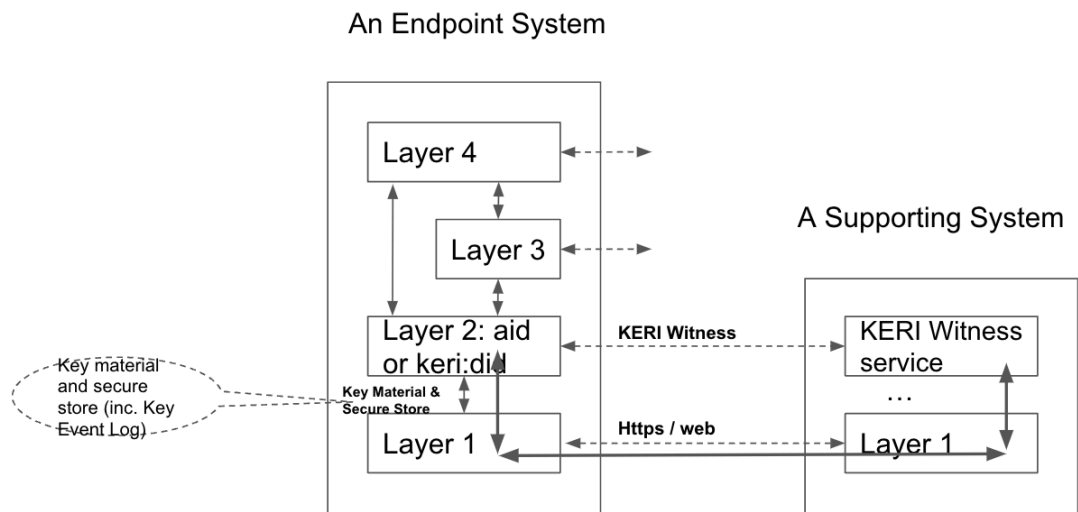


Figure 15: Example of a KERI witness as a supporting system

Generalization

Figure 16 illustrates a generalization of the pattern in which [endpoint systems](#) and their respective [supporting systems](#) interact. This figure makes it clear that the interoperability between [endpoint systems](#) in each layer is orthogonal to the methods of interaction with respective [supporting systems](#).

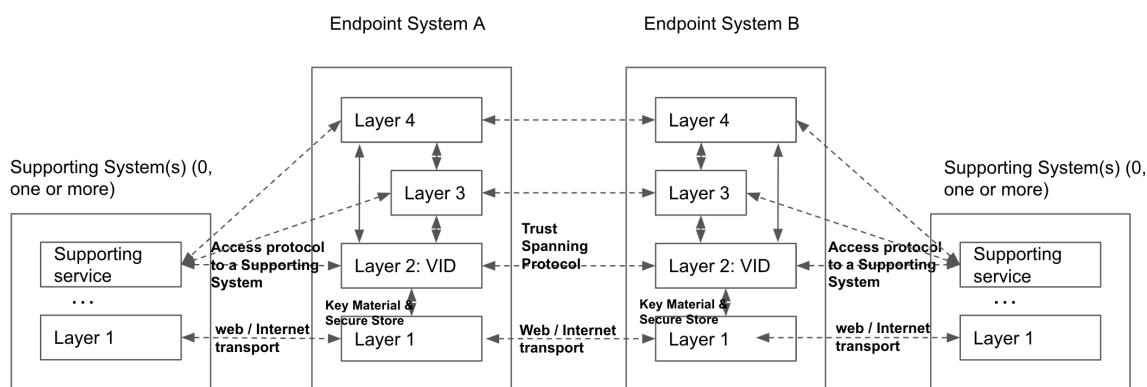


Figure 16: A generalization of how endpoint systems and supporting systems interact

Endpoint System Interoperability

Interoperability between Endpoint Systems Using Decentralized Identifiers

[Section 6.4](#) states that "[Endpoint systems](#) [need to] be able to associate, discover and verify the cryptographic keys associated with a VID." This capability is essential in order for two or more [endpoint systems](#) to be able to discover and connect with each other over the [ToIP Trust Spanning Protocol](#).

If an [endpoint system](#) is identified with a publicly resolvable [decentralized identifier](#) (DID) as defined in [section 6.4](#), this is straightforward because a [DID resolver](#) can:

1. Resolve the DID to the authoritative [DID document](#).
2. Extract the appropriate public key.
3. Extract the [service endpoint](#) URI for the [ToIP Trust Spanning Protocol](#).

If an [endpoint system](#) is identified with a private, pairwise DID — called a **peer DID** — the discovery and exchange of a [DID document](#) needs to use an [out-of-band interaction](#) (OOBI) protocol. Common examples include [QR codes](#) and custom-generated [deep links](#).

Interoperability between Endpoints Systems Using Other Verifiable Identifiers

If an [endpoint system](#) is not identified with a DID, but with some other kind of VID as defined in [section 6.4](#), then a different approach must be used to bootstrap communications using the [ToIP Trust Spanning Protocol](#). This requires enabling discovery and verification of:

1. The authoritative public key for the [endpoint system](#).
2. The authoritative [service endpoint](#) URI for communicating with the [endpoint system](#) over the [ToIP Trust Spanning Protocol](#).

If the VID is an HTTPS URL, there are at least two solutions:

1. Conversion of the HTTPS URL into a **did:web:** identifier as described in the [ToIP X.509 PKD Interop page](#) [↗](#).
2. Issuance by a trusted [issuer](#) (such as a [certification authority](#)) of a [verifiable credential](#) whose subject is the HTTPS URL and whose claims assert the authoritative public key and [ToIP Trust Spanning Protocol service endpoint](#) URI.

We anticipate that integration of decentralized PKI and X.509 PKI will be a topic of increasing interest and innovation.

Integration with the ToIP Governance Stack

As explained in the Introduction, this specification, maintained by the ToIP [Technology Stack Working Group](#) [↗](#), is focused entirely on requirements for the [ToIP Technology Stack](#). A separate set of specifications, maintained by the ToIP [Governance Stack Working Group](#) [↗](#), defines the requirements for the [ToIP Governance Stack](#). The first generation of the [ToIP Governance Architecture Specification](#) and related specifications were published in January 2022 and are summarized [here](#) [↗](#).

Although the [ToIP Governance Architecture Specification](#) consist largely of recommendations about the structure and content of [governance documents](#) for ToIP-based [digital trust ecosystems](#), there are a very small but vital set of technical requirements that are essential for “tying the two stacks together”.

In particular, section 3 of the [ToIP Governance Architecture Specification V1.0](#) [↗](#) specifies a set of **identification requirements** for ToIP-compatible governance frameworks. A high-level summary:

1. The [primary document](#) for the [governance framework](#) MUST be assigned a DID and be retrievable via a [DID URL](#). This DID identifies the [governance framework](#) itself as a digital object, and the [DID URL](#) allows it to be viewed and verified over the Web by any [party](#).
2. All other [controlled documents](#) in the [governance framework](#) MUST have [DID URLs](#).
3. The [DID URLs](#) for all [governance framework](#) documents MUST be **versioned** as the documents are versioned.
4. The [governing body](#), [administering body](#) (if separate from the [governing body](#)), and all [governed parties](#) in the [governance framework](#) MUST be identified with DIDs.

The use of persistent, discoverable, [cryptographically verifiable](#) identifiers for all parties and documents governing a [digital trust ecosystem](#) makes it much easier to bind technology to policy. For example:

- A [verifiable credential](#) issued within the ecosystem can include a [claim](#) asserting the DID of the authoritative [governance framework](#).
- The credential can also include the DID of one or more [trust registries](#) where a [holder](#) or [verifier](#) can verify that the DID of the credential [issuer](#) is authorized to issue that particular type of credential under that [governance framework](#).
- A credential asserting [certification](#) under the [governance framework](#) can include the DID of the [governance framework](#), the DID of the [certification authority](#), the DID of the [governed party](#) being certified, and the [DID URL](#) of the precise type of [certification](#) being awarded.



For additional recommendations about integration of the [ToIP Governance Stack](#) with the [ToIP Technology Stack](#), please see the [ToIP Governance Architecture Specification V1.0](#) [↗](#) and the [ToIP Governance Metamodel Specification V1.0](#) [↗](#).

References

NOTE: References in this second public review draft (PR2) are currently provided inline as hyperlinks. TODO-ADD separate lists of Normative and Informative References.

About the ToIP Foundation

Founded in May 2020, the ToIP Foundation has grown to over 200 participating organizations plus as many more individual participants. Our mission is to define an overall architecture for Internet-scale digital trust that combines cryptographic assurance at the machine layers (technology) with human accountability at the business, legal, and social layers (governance).

For more information about ToIP Foundation, please read our [Introduction to ToIP](#)  white paper or visit our website at <https://trustoverip.org/> .

Appendix A: Consolidated Requirements

For ease of reference, the following table consolidates all normative requirements in this specification. Each requirement is linked to the section in which it appears.

Req ##	Description
	General ToIP Architecture Requirements
A.1	Implementers SHOULD ensure autonomy for ToIP endpoint systems .
A.2	In a ToIP endpoint system , the higher layers of the ToIP protocol stack MUST communicate with
A.3	A ToIP intermediary system SHOULD be able to perform the functions of a ToIP endpoint system
A.4	The ToIP protocol stack in an endpoint system MAY use the services of a supporting system at a
	ToIP Layer 1 Requirements
L1.1	If a ToIP endpoint system includes trust support functions, then those functions MUST be includ
	ToIP Layer 2 Requirements
L2.1	A ToIP endpoint system MUST communicate with another ToIP endpoint system using the ToIP T
L2.2	A VID MUST be unique within the context in which it is used for identification.
L2.3	A VID MUST be cryptographically verifiable , i.e., verifiably bound to at least one set of cryptogra
L2.4	A VID SHOULD be decentralized, i.e., not require registration with a centralized authority.
L2.5	A VID SHOULD be a self-certifying identifier (SCID), i.e., a fully portable identifier that can be ve
L2.6	A VID SHOULD support rotation of the associated cryptographic keys for the lifetime of the iden
L2.7	A VID MAY also support rotation to an entirely different VID that can be cryptographically verifie
L2.8	A VID SHOULD support the ability to: a) associate the VID with the network address of one or m
L2.9	The ToIP Trust Spanning Protocol specification MUST define how to construct and format messa
L2.10	In a ToIP endpoint system , an implementation of the ToIP Trust Spanning Protocol MUST suppor
L2.11	In a ToIP endpoint system , an implementation of the ToIP Trust Spanning Protocol MAY support g
L2.12	The ToIP Trust Spanning Protocol MUST enable the composition of higher-level trust task protoc
L2.13	The ToIP Trust Spanning Protocol MUST support extensible message schema.
L2.14	The ToIP Trust Spanning Protocol MUST support resolution of VIDs to: a) the network addresses
L2.15	The ToIP Trust Spanning Protocol MUST support transport of messages via ToIP Layer 1 interfac
L2.16	The ToIP Trust Spanning Protocol MUST support delivery of messages to the Layer 2 interface o
L2.17	The ToIP Trust Spanning Protocol MUST support delivery of messages via intermediary systems
L2.18	The ToIP Trust Spanning Protocol MUST support confidentiality with regard to the metadata req

Req ##	Description
	ToIP Layer 3 Requirements
L3.1	A Layer 3 trust task protocol MUST communicate either over the Layer 2 ToIP Trust Spanning Protocol .
L3.2	A Layer 3 trust task MAY use other protocols, but only for other purposes (since short-circuiting
L3.3	A Layer 3 trust task protocol intended to communicate private data SHOULD support confidentiality .
	ToIP Layer 4 Requirements
L4.1	Layer 4 trust applications MAY use any number of Layer 3 trust task protocols .
L4.2	If a Layer 4 trust application does not use a Layer 3 trust task protocol , it MUST communicate with
L4.3	A Layer 4 trust application MUST support any ToIP-defined trust affordances relevant to that application.

Appendix B: Consolidated Views of the ToIP Technology Stack

The ToIP Technology Architecture Task Force has spent many hours discussing how to produce consolidated views of ToIP architecture that are both relatively easy to understand but still technically accurate. In the end, we agreed no single diagram is sufficient. Rather, different views of the architecture should be taken together to see the whole picture. In this appendix we present several of these views — and [we invite feedback](#) on others that might be helpful.

Functional Hourglass View

Figure B1 is a view of the types of functions that belong at each layer within a single [endpoint system](#) as defined in this specification. It illustrates how the [Hourglass Model](#) is implemented as a single [trust spanning protocol](#) at Layer 2, with multiple trust support functions below and multiple supported [trust task protocols](#) above. It also shows one example (at the far right) of a specific category of [supporting systems](#), in this case [verifiable data registries](#) (VDRs) upon which an [endpoint system](#) can rely as external sources of truth.

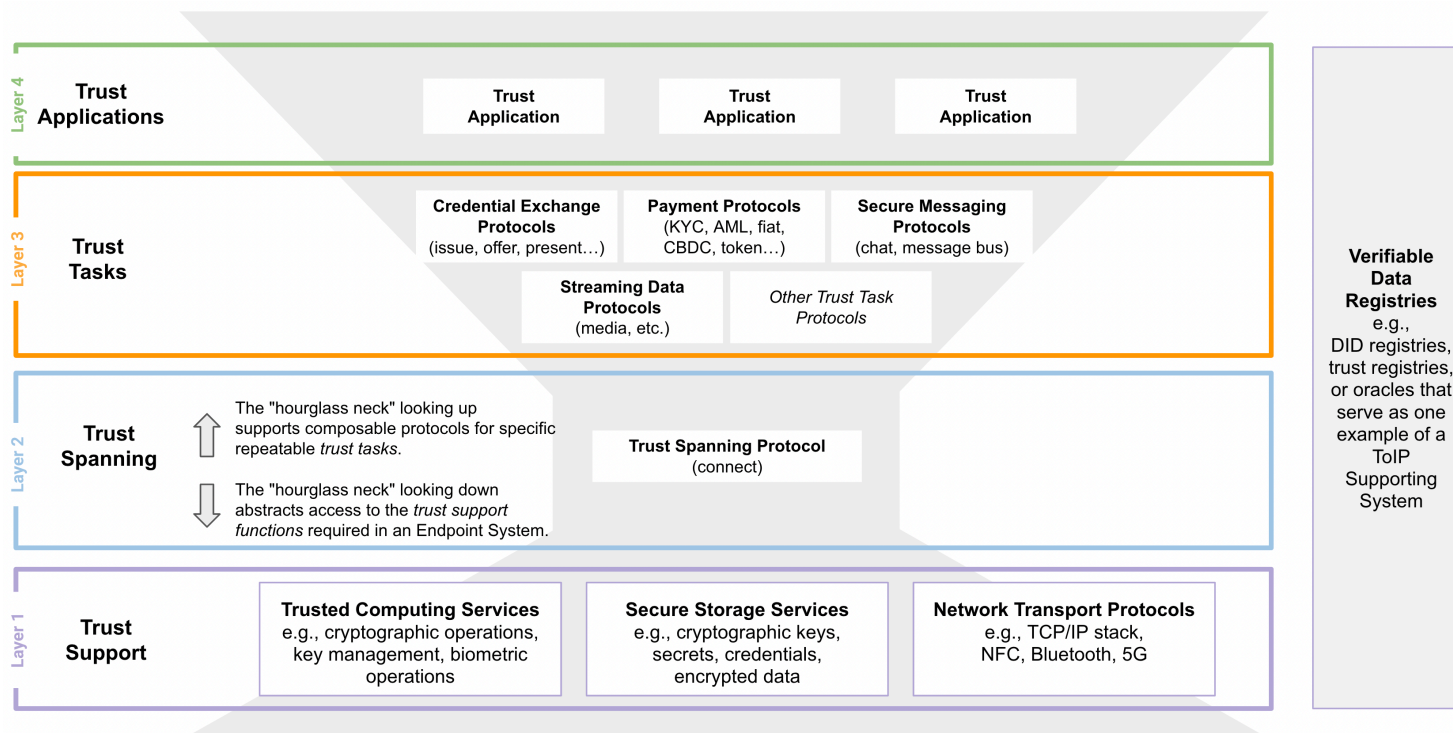


Figure B1: A layer-by-layer view of functions within an endpoint system (also showing verifiable data registries as one type of adjacent supporting system)

Sphere-of-Influence View

Figure B2 builds on Figure B1 by identifying those technical capabilities that fall within the purview of ToIP’s technical architecture and those that are outside that boundary and thus do not need to be governed by ToIP component specifications.

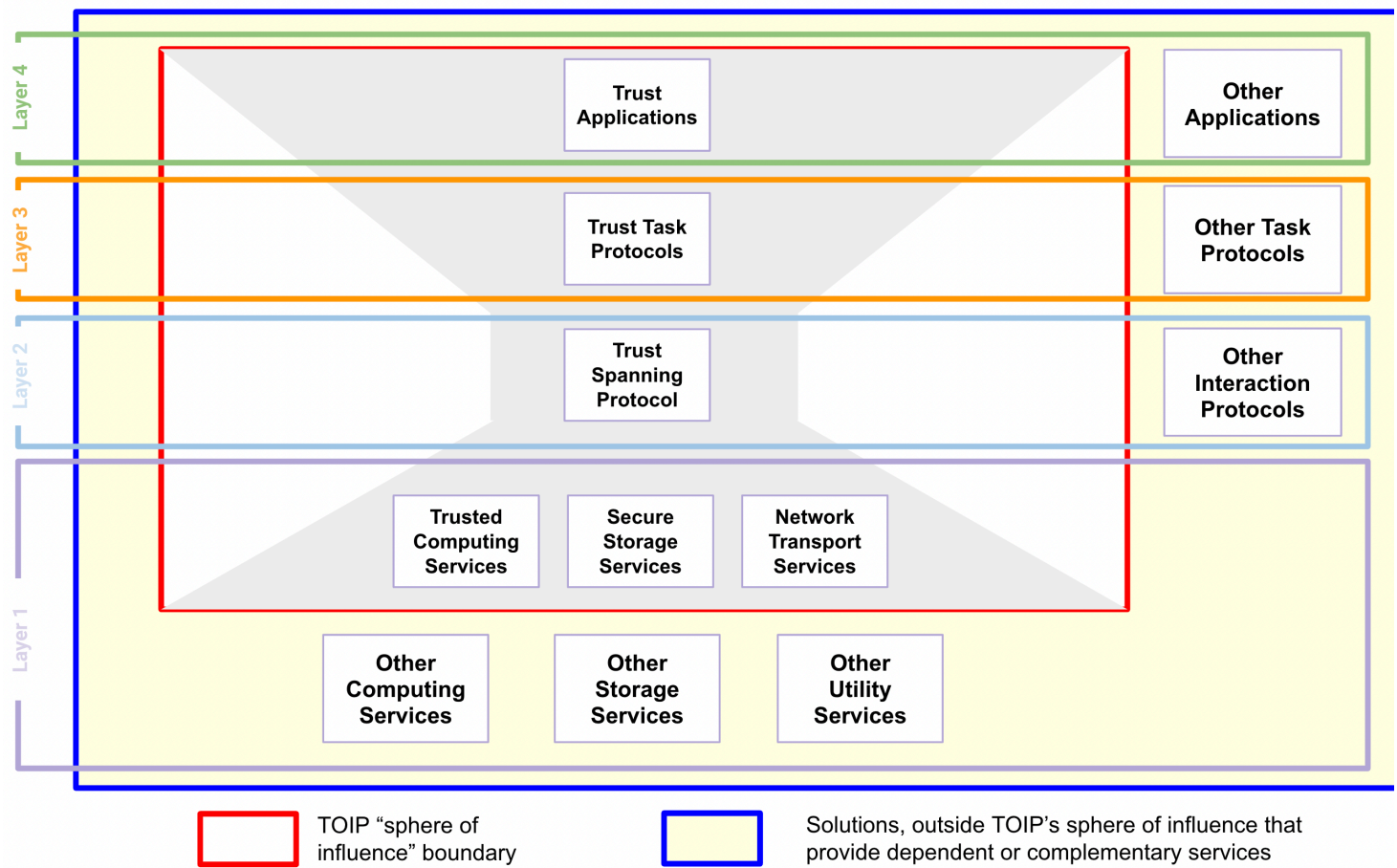


Figure B2: A view of the ToIP Technology Stack that shows what is inside and outside ToIP’s “sphere of influence”

This view shows how the logical capabilities and components identified in the functional Hourglass View can align with dependent solutions that are not governed by the requirements of the ToIP stack. For example, a [DID resolver](#) functioning at Layer 2 in an [endpoint system](#) may call a DID ledger functioning as a [verifiable data registry](#). While the [DID resolver](#) interface is a ToIP Layer 2 function, the DID ledger called by the associated [DID method](#) is a [supporting system](#) that has its own resolution protocol as defined by the [DID method](#).

Interaction Pattern View

Figure B3 builds on B1 and B2 by showing the interaction patterns between two different [endpoint systems](#) as well as between an [endpoint system](#) and a set of [supporting systems](#) (on the far right).

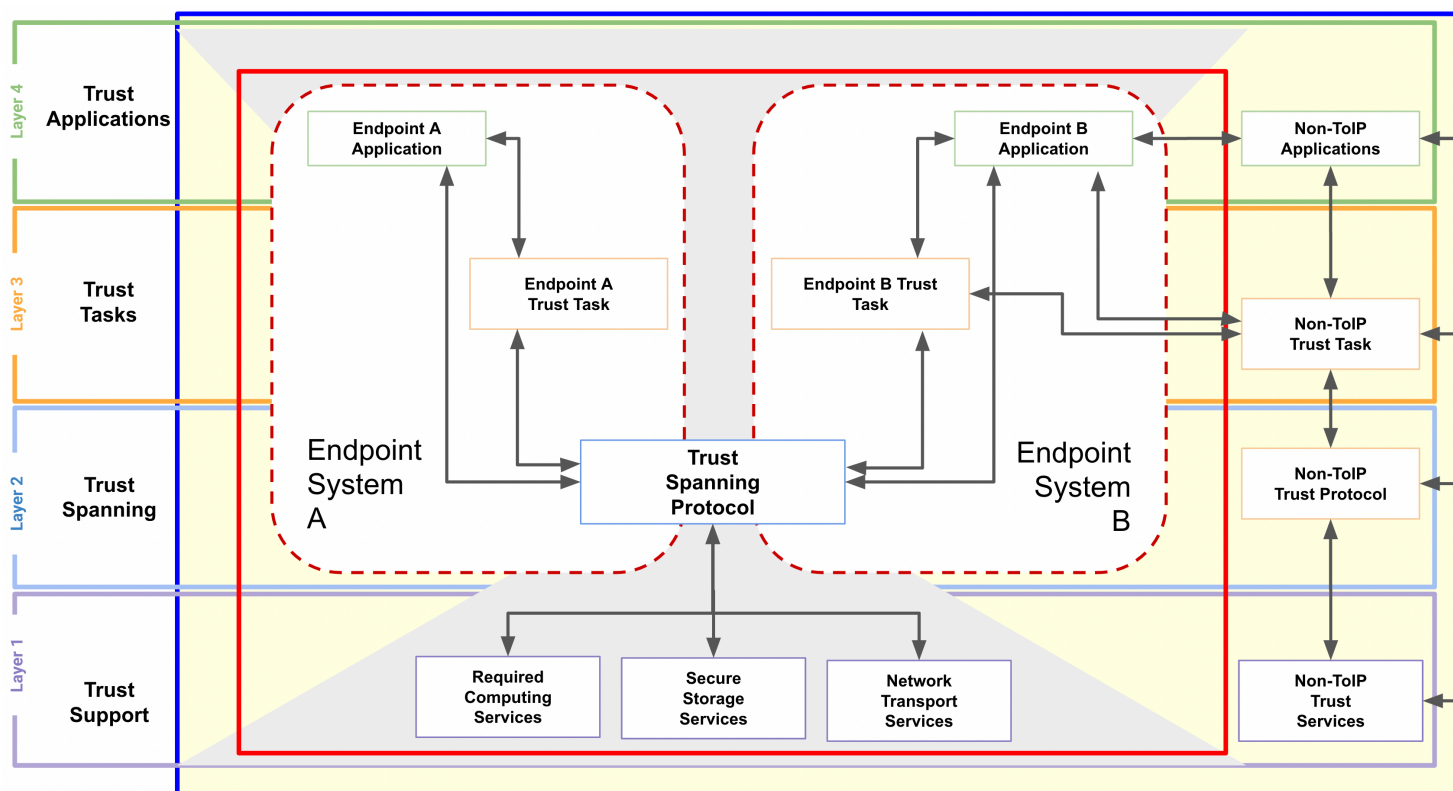


Figure B3: A view showing the interaction patterns both within and between two endpoint systems (as well as with supporting systems on the far right)

Appendix C: Mapping of Existing Technologies into the ToIP Technology Stack

Just as the TCP/IP stack did not need to reinvent or replace existing local area networks, but instead added a new inter-networking layer to connect them, the ToIP stack does not reinvent or replace existing centralized or federated identity systems and PKI trust infrastructures. The ToIP stack adds a new inter-*trust* networking layer to connect the existing trust domains.

As this new layer of decentralized digital identity and trust infrastructure has been evolving, many individual pieces of the puzzle have been developed in parallel. For example:

- Almost 200 different DID methods (see the [W3C DID Spec Registries](#) ^[7]).
- At least a dozen different digital credential formats and signature schemes.
- At least three digital credential exchange protocols (DIDComm/Aries, OIDC4VC, W3C VC API)
- Several new decentralized payment protocols (e.g., [TBDex](#) ^[7]).
- Multiple [QR code](#) formats and [out-of-band introduction](#) (OOBI) protocols (e.g., [OASIS Secure QR Codes](#) ^[7]).

From the perspective of the ToIP stack, all of these are potential component specifications.

- Some may fully meet the requirements of this [ToIP Technology Architecture Specification](#) without any modification.
- Others may require ToIP-conformant profiles to be written.
- Still others may require (hopefully small) revisions to meet ToIP requirements.
- A few (ideally very few) may need to be developed from scratch, either at the ToIP Foundation or another suitable industry organization. Two examples are the [ToIP Trust Spanning Protocol](#) and the [ToIP Trust Registry Query Protocol](#).

Given the speed at which this new evolutionary branch of the Internet is evolving, the ToIP Foundation is maintaining a "mapping" of existing technologies and open standards into the ToIP stack on a web page called [Evolution of the ToIP Stack](#) ^[7].

We recommend referring to this page to see the current mapping. The Foundation intends to publish an updated version of this document with each major development in the space.