ToIP Trust Registry Query Protocol (TRQP) v2.0

- **Specification Version:** 2.0
- **Document Status:** ToIP Approved Deliverable
- **ToIP Permalink:** https://github.com/trustoverip/tswg-trust-registry-protocol
- DOI: 10.5281/zenodo.18789010

> NOTE
>
> A Working Draft of the next version of this specification is also available.

Participate:

GitHub repo
File an issue
Start a discussion

- **Editors:** Darrell O'Donnell, Andor Kesselman, Drummond Reed
- **Contributors:** Alex Tweeddale, Antti Kettunen, Christine Martin, Dave Poltorak, Eric Drury, Fabrice Rochette, Jacques Latour, Jesse Carter, Jeff Braswell, Jon Bauer, Makki Elfatih, Marcus Ubani, Markus Sabadello, Scott Perry, Sankarshan Mukhopadhyay, Subhasis Ojha, Tim Bouma

For a complete overview of the motivations and core concepts behind TRQP, please see the TRQP Overview page.

§ 0.1 Intellectual Property Rights

§ 0.1 Change History

Public Review 02 — December 2025

- Restructured specification into modular files (overview, definitions, high-level architecture, identifiers, API schemas, HTTPS binding, conformance, considerations, references).
- Adopted the PARC (Principal, Action, Resource, Context) information model for identifiers and query structure.
- Replaced `ecosystem_id` and `assertion_id` with `authority_id`, `entity_id`, `action`, and `resource` aligned to the PARC model.
- Added formal JSON schemas for authorization and recognition request/response messages.
- Added HTTPS binding section with detailed request/response examples and error handling using RFC 7807 Problem Details.
- Added `locator` as optional parameter to the `context` object for authorization queries.
- Added conformance criteria section defining TRQP Endpoint, TRQP Consumer, and HTTPS Binding conformance targets.
- Added formal Normative and Informative References sections.
- Added Versioning, Extensibility, and Backwards Compatibility section.
- Deferred delegation and description (metadata) query types to a future version.
- Extensive editorial revision throughout.

Public Review 01 — April 2025

- Initial public review of TRQP v2.0 specification.
- Defined core protocol concepts: authorities, authority statements, trust registries, governance frameworks.
- Defined authorization and recognition query types.

- Established identifier requirements based on RFC 3986.
- Included Security, Privacy, and Implementation Considerations sections.

§ 1. Introduction

This section is informative.

The ToIP Trust Registry Query Protocol (TRQP) is a lightweight, read-only protocol for making fast, efficient queries for authoritative data from *trust registries*, also known as *trust lists*. To use an analogy, TRQP is to trust registries what DNS is to name servers.

The same way DNS name servers serve name domains, TRQP trust registries serve *trust domains*, also known as *digital trust ecosystems*. Four primary actors participate in the flow of verifiable data (including *verifiable credentials*) within the ecosystem: 1) data producers (issuers), 2) data subjects (holders), 3) data consumers (verifiers or relying parties), and 4) governing bodies (authorities).

Authorities determine the policies governing which actors can perform what actions on what data within the ecosystem. These policies are typically published in a human-readable form called a *governance framework*, also known as a *trust framework*. To make these policies accessible to software agents, they are published in a machine-readable form known as *authority statements*. Authority statements can be published in a file, issued to individual actors as verifiable credentials, or published in a trust registry.

Digitally-verifiable authority statements can be expressed using various standards, including X.509 certificate hierarchies, OpenID Federations, EBSI Trust Chains, or TRAIN trust lists. Although these standards can work well for *intra-ecosystem* authority verification, they are not optimized for *inter-ecosystem* authority verification.

The purpose of TRQP is to bridge this gap by providing a standard protocol for querying authority statements from any TRQP-compliant trust registry. It specifies a standard data model, query vocabulary, and transport protocol binding that can be implemented by any ecosystem regardless of its internal trust architecture.

TRQP focuses on two query types:

1. **Authorization Queries:** “Has Authority A authorized Entity B to take Action X on Resource Y?”
2. **Recognition Queries**: "Does Authority X recognize Entity B as an authority to authorize taking Action X on Resource Y?”

§ 2. Conventions and Definitions

§ 2.1 Keywords

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in IETF RFC 2119.

§ 2.2 Definitions

authority
: The entity responsible for making authority statements expressing the governance policies for its trust domain or digital trust ecosystem.

authority ID
: The globally unique identifier of an authority.

authority statement
: An assertion by an authority about another entity. Types of authority statements include authorization, recognition, delegation, and description (metadata).

authorization statement
: An authority statement in which an authority grants an authorization to an entity over which it has authority.

digital trust ecosystem
: A digital ecosystem in which participants are one or more interoperating trust communities. Governance of various roles within this ecosystem is typically managed by a governing body using a governance framework. Many digital trust ecosystems maintain one or more trust registries.

ecosystem
: See digital trust ecosystem.

ecosystem governing authority
: The authority responsible for governance of a digital trust ecosystem and for publishing its authority statements. An ecosystem governing authority may take any legal form or may not be a formal legal entity at all.

ecosystem governance framework
: A governance framework for a digital trust ecosystem. This may incorporate other types of frameworks such as credential governance frameworks.

entity ID
: The unique identifier of an entity within a trust domain or digital trust ecosystem.

governance framework
: A collection of one or more governance documents published by the governing body of an ecosystem or any kind of trust community.

inter-ecosystem
: An adjective describing relationships and data exchanges between participants in two or more separate ecosystems operating under separate governance frameworks.

intra-ecosystem
: An adjective describing relationships and data exchanges between participants operating within the same ecosystem and the same governance framework.

recognition statement
: An authority statement in which one authority recognizes the authority of another authority as a peer. Note that this recognition relationship may be unilateral or bilateral and is non-exclusive.

TRQP binding
: A technical specification defining how to implement the TRQP protocol over a specific transport protocol.

TRQP bridge
: A system that connects a TRQP endpoint to a system of record. The bridge transforms a TRQP query into the query format supported by the system of record and performs the reverse mapping for the response.

TRQP consumer
: A network device (client or server) that sends TRQP queries to a TRQP endpoint.

TRQP endpoint
: The network service endpoint for a trust registry that implements TRQP.

trust registry
: A repository that serves as a source for authority statements or other governed information describing one or more trust communities. A trust registry is typically authorized by an ecosystem governance framework.

trust registry operator
: The legal entity responsible for operating a trust registry. A trust registry may be operated directly by an ecosystem governing authority, or operation may be delegated to an independent trust registry operator.

system of record
: An authoritative source for the authority statements available from a trust registry.

§ 3. Scope

This section is informative.

Figure 1 illustrates the four primary components involved with TRQP architecture.

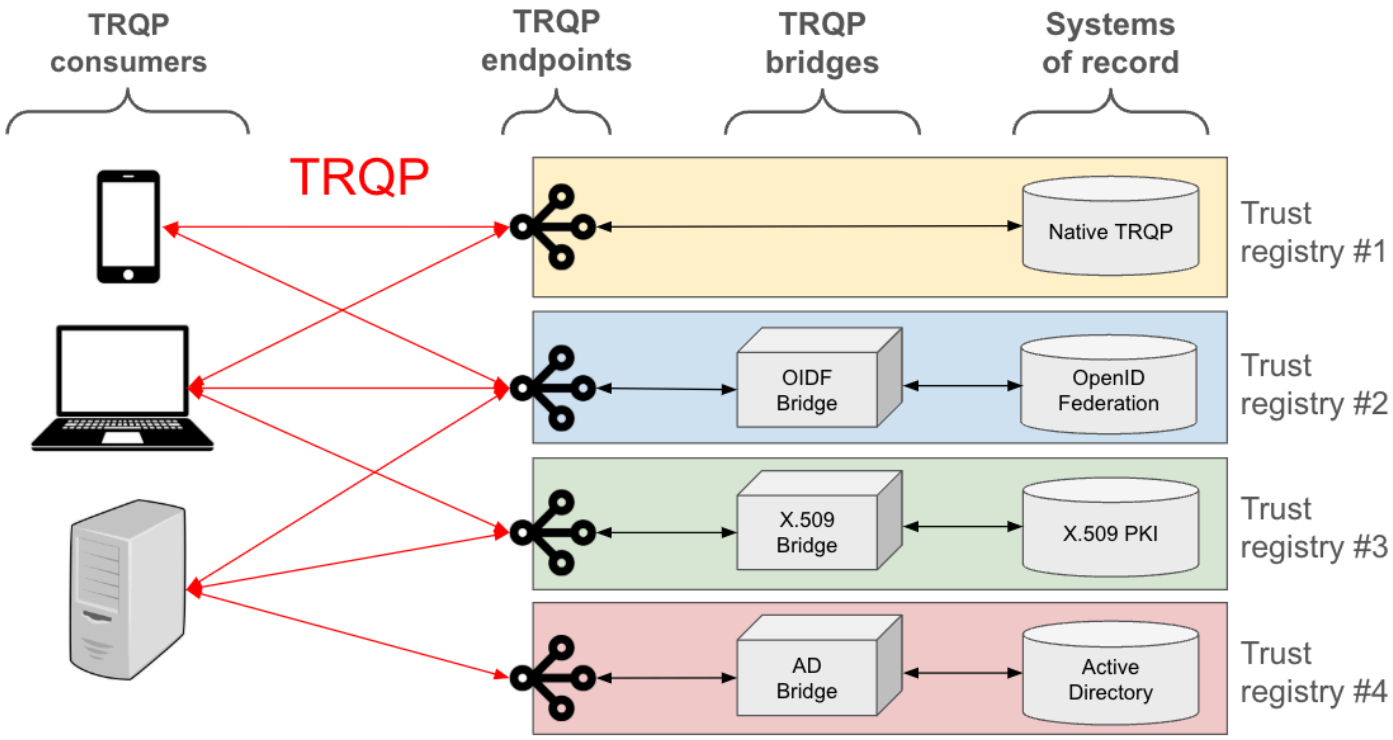

Figure 1: The primary components involved in TRQP architecture.

The scope of this specification is limited to the TRQP protocol operating between TRQP consumers and TRQP endpoints representing addressable TRQP trust registries. The following are out-of-scope:

- **Systems of record**. This specification casts no requirements on how the system of record is designed or deployed. Also, because TRQP is read-only, this specification does not address create, update, or delete operations for the system of record.
- **TRQP bridges**. If the system of record is not a native TRQP trust registry, a TRQP bridge is needed to transform a TRQP query into the query format supported by the system of record. Separate specifications may be published for popular TRQP bridges, however they are out-of-scope for this specification.
- **Implementation Code**. TRQP defines the protocol; it does not provide the code for implementation.

§ 4. High-Level Architecture

This section is informative.

This section presents the major conceptual components in TRQP architecture.

§ 4.1 Authorities

Authorities are the parties responsible for establishing the policies governing their ecosystem. An authority may take any legal form or may not be a formal legal entity at all. The only requirement is that the authority be recognized by the stakeholders in their ecosystem as being authoritative for the authority statements in the trust registry or registries serving that ecosystem.

Nation states, companies, NGOs, universities, churches, associations, social networks, online communities, and open source projects are all examples of entities who could serve as authorities.

§ 4.2 Authority Statements

An authority statement is a machine-readable representation of a policy governing an entity's authorization within the authority's scope of authority. Trust registries serve as a mechanism for making authority statements accessible to parties who need to make trust decisions regarding those entities.

TRQP supports two basic types of authority statements:

§ 4.2.1 Authorization Authority Statements

An authorization statement expresses a hierarchical relationship between the `authority_id` and the `entity_id`. It represents a declaration by an authority that an entity under its jurisdiction or sphere-of-control is authorized to take a specific action on a specific resource.

Example Authorization Statement (Pseudocode)

```
authority_id: American Association of Motor Vehicle Administrators (AAMVA)
entity_id : Department Of Motor Vehicles (DMV)
action: issue
resource: DriversLicense
```

In English, this corresponds to the statement: "AAMVA, acting as an administering body delegated by its member states (the governing bodies), authorizes a member's DMV to issue Driver's Licenses according to the governance framework maintained by AAMVA on behalf of those states."

§ 4.2.2 Recognition Authority Statements

A recognition statement expresses that one authority recognizes the authority of another as a **peer**. Such a recognition relationship may be unilateral or bilateral and is non-exclusive.

> NOTE
>
> Unlike an authorization statement, the authority making a recognition statement is **not** asserting authority over the target authority. Rather it is a referral from one peer to another.

Example Recognition Statement (Pseudocode)

```
authority_id: France
entity_id : Germany
action: issue
resource: Passport
```

In English, this corresponds to the statement: “France recognizes the German Government for official German Government sources and issued credentials (e.g. passport issuance).”

§ 4.3 Governance Frameworks

Just as an authority may take any form, so may the policies governing its ecosystem. For the purposes of this specification, the collection of these policies (whether human-readable and/or machine-readable) is called the governance framework.

To facilitate trust decisions by its stakeholders—or by any other relying party—the authority is responsible for publishing its governance framework. Although they are not normative requirements of this specification, the following recommendations apply:

1. The governance framework should be published using a verifiable identifier so its authenticity can be verified.
2. The governance framework ID should be discoverable via the authority ID.
3. The governance framework should follow the recommendations of the ToIP Governance Architecture Specification and ToIP Governance Metamodel Specification.

§ 4.4 Trust Registries

In the context of this specification, a trust registry is a system accessible via a TRQP endpoint that can be queried for the authority statements published by one or more authorities. A trust registry is operated by a trust registry operator. The role of a trust registry operator may be performed directly by an authority or may be delegated to an independent trust registry operator

who specializes in this function. In the latter case, from a ToIP governance architecture perspective, the trust registry operator is serving as an administering authority.

The TRQP service endpoint for a trust registry may be published in the governance framework or discoverable from the authority_id as described in the identifiers section.

§ 5. Identifiers

This section is normative.

Interoperability of TRQP across decentralized trust domains, communities, and ecosystems depends on globally unique identifiers in the same way interoperability of the Internet depends on IP addresses and DNS names.

The following requirements apply to all identifiers defined in this section:

1. The identifier MUST be represented as a single string conforming to IETF RFC 3986.
2. For globally unique identifiers, it is RECOMMENDED to use a verifiable identifier such as a W3C Decentralized Identifier (DID), a KERI autonomic identifier (AID), or an HTTPS URL so their authenticity can be verified by any relying party.

For additional assurance, it is RECOMMENDED to use multi-anchoring of identifiers as defined by the IETF High Assurance DIDs using DNS specification or the work of the ToIP High Assurance Verifiable Identifiers Task Force.

> NOTE
>
> The TRQP information model is patterned after the PARC model. The identifiers in this section map well to PARC: Principal ~= `entity_id`; Action ~= `action` Resource ~= `resource`; Context ~= `authority_id` as the *mandatory* context. An optional `context` object is available for additional query conditions, such as time.

§ 5.1 `authority_id`

1. The `authority_id` MUST be the globally unique identifier of the authority in the context of an authority statement.
2. It is RECOMMENDED that the `authority_id` be published in the governance framework for that ecosystem.
3. It is RECOMMENDED that the TRQP service endpoint(s) for any authoritative trust registry be machine-discoverable via the `authority_id`. An example would be to publish either of the following in the DID document for the `authority_id`:

1. The authoritative TRQP service endpoint URL(s).
2. The DID(s) identifying authoritative trust registries. (In that case, the authoritative TRQP service endpoint URL(s) would be specified in the associated DID documents.)

4. The `ecosystem governance framework` MUST be discoverable via the `authority_id`. This can be established in the DID Document (in the case the `authority_id` is a DID, or another mechanism.)

§ 5.2 `entity_id`

1. The `entity_id` MUST be the identifier of the principal in an authority statement.
2. For a recognition or delegation statement, the `entity_id` MUST also be an `authority_id`.
3. For an authorization statement about a governed party, the `entity_id` MUST be unique in the scope of the authority. It is NOT REQUIRED for the `entity_id` to be globally unique.

§ 5.3 `action`

1. The identifier for an `action` in an authority statement MUST be a non-empty string conformant to the requirements in this section.
2. It is NOT REQUIRED for the `action` identifier string to be globally unique.
3. The `action` identifier string SHOULD:
   1. Be specified in the authority's governance framework.
   2. Be machine-discoverable via a query to the authoritative trust registries.

§ 5.4 `resource`

1. The identifier for a `resource` in an authority statement MUST be a non-empty string conformant to the requirements in this section.
2. It is NOT REQUIRED for the `resource` identifier string to be globally unique.
3. In addition, the requirements for `resource` identifiers SHOULD:
   1. Be defined in the authority's governance framework.
   2. Be machine-discoverable via a query to the authoritative trust registries.

§ 5.5 `context`

Because `authority_id` is the required context for all TRQP authority statements, a `context` object is OPTIONAL in a query. If a `context` object is included, it MUST conform to the requirements in this section.

1. A `context` object MUST be a JSON object whose members convey other query conditions.

2. If a `context` object needs to express a time-based condition:
    1. It MUST include a `time` parameter.
    2. The value of this parameter MUST be a time value expressed in RFC 3339 format.
    3. The value of the `time` parameter MUST be interpreted as the datetime as of which the target authority statement is valid.
3. Additional JSON object members specifying other conditions MAY be defined by TRQP profiles or bindings.

---

§ 6. Authorization Query and Response Schemas

This section is normative.

The purpose of a TRQP authorization query is to ask the question "Does `authority_id` authorize `entity_id` to take `action` on `resource` (with optional context conditions such as `time`)?"

TRQP authorization queries and responses MUST conform to the JSON schemas defined in this section.

§ 6.1 Authorization Query Schema

```
{
  "$id": "trqp-authorization-request",
  "title": "AuthorizationRequest",
  "type": "object",
  "required": [
    "entity_id",
    "action",
    "resource",
    "authority_id"
  ],
  "properties": {
    "entity_id": {
      "type": "string",
      "description": "The identifier for the entity that is being tested for authorizatio
    },
    "authority_id": {
      "type": "string",
      "description": "The identifier for the authority (e.g. ecosystem governing authorit
    },
    "action": {
```

```
        "type": "string",
        "description": "The action that the query is checking Authorization for."
      },
      "resource": {
        "type": "string",
        "description": "The resource that the query is checking Authorization for."
      },
      "context": {
        "type": "object",
        "properties": {
          "time": {
            "type": "string",
            "format": "date-time",
            "description": "Time for query to use. If blank, uses current server time. RFC3
          },
          "locator": {
              "type": "string",
              "description": "Provided for systems that require additional information to l
          }
        },
        "additionalProperties": {
          "type": "string"
        }
      }
    }
}
```

Example authorization query:

```
POST /authorization
Content-Type: application/json

{
  "entity_id":    "user-1234",
  "authority_id": "auth-service-A",
  "action":       "issue",
  "resource":     "country:state:driverlicense",
  "context": {
    "time": "2025-06-19T11:30:00Z"
  }
}
```

source: trqp_authorization_request.schema.json

§ 6.2 Authorization Response Schema


```
{
  "$id": "trqp-authorization-response",
  "title": "AuthorizationResponse",
  "type": "object",
  "required": [
    "entity_id",
    "action",
    "resource",
    "authority_id",
    "time_evaluated",
    "authorized"
  ],
  "properties": {
    "entity_id": {
      "type": "string",
      "description": "The identifier for the entity that was tested for authorization."
    },
    "authority_id": {
      "type": "string",
      "description": "The identifier for the authority that was queried."
    },
    "action": {
      "type": "string",
      "description": "The action that the query is checking Authorization for."
    },
    "resource": {
        "type": "string",
        "description": "The resource that the query is checking Authorization for."
    },
    "authorized": {
      "type": "boolean",
      "description": "True if the action+resource authorization has been confirmed, false
    },
    "time_requested": {
      "type": "string",
      "format": "date-time",
      "description": "the server time that was requested (may be blank), as provided in t
    },
    "time_evaluated": {
      "type": "string",
      "format": "date-time",
      "description": "the server time that the query was executed at on the endpoint serv
    },
```

```
    "message": {
      "type": "string",
      "description": "additional details attached to the response about the authorization
    },
    "context": {
      "type": "object",
      "description": "the context object that was supplied for the query that was evaluat
      "properties": {
        "time": {
          "type": "string",
          "description": "Time for query to use. If blank, uses current server time. RFC3
        },
        "locator": {
            "type": "string",
            "description": "Provided for systems that require additional information to l
        }
      },
      "additionalProperties": {
        "type": "string"
      }
    }
  }
}
```

Example authorization response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "entity_id":    "did:user-1234",
  "authority_id": "auth-service-A",
  "action":       "issue",
  "resource":     "country:state:driverlicense",
  "authorized":   true,
  "time_requested":         "2025-06-25T00:42:00Z",
  "time_evaluated":         "2025-06-19T11:30:00Z",
  "message":      "did:user-1234 is authorized for issue+country:state:driverlicense (act
}
```

source: trqp_authorization_response.schema.json

§ 7. Recognition Query and Response Schemas

This section is normative.

The purpose of a TRQP recognition query is to ask the question "Does `authority_id` recognize `entity_id` (another authority) to be authoritative for `action` on `resource`?"

TRQP recognition queries and responses MUST conform to the JSON schemas defined in this section.

§ 7.1 Recognition Query Schema

```
{
  "$id": "trqp-recognition-request",
  "title": "RecognitionRequest",
  "type": "object",
  "required": [
    "entity_id",
    "action",
    "resource",
    "authority_id"
  ],
  "properties": {
    "entity_id": {
      "type": "string",
      "description": "The identifier for the entity that is being tested for recognition.
    },
    "authority_id": {
        "type": "string",
        "description": "The identifier for the authority (e.g. ecosystem governing author
    },
    "action": {
      "type": "string",
      "description": "The action that the query is checking Recognition for."
    },
    "resource": {
      "type": "string",
      "description": "The resource that the query is checking Recognition for."
    },
    "context": {
      "type": "object",
      "properties": {
        "time": {
          "type": "string",
          "format": "date-time",
          "description": "Time for query to use. If blank, uses current server time. RFC3
        }
```

```
      },
      "additionalProperties": {
        "type": "string"
      }
    }
  }
}
```

Example recognition query:

```
POST /recognition
Content-Type: application/json

{
  "entity_id":    "service-42",
  "authority_id": "did:example",
  "action":       "recognize",
  "resource":     "listed-registry",
  "context": {
    "time": "2025-06-19T10:00:00Z"
  }
}
```

source: trqp_recognition_request.schema.json

§ 7.2 Recognition Response

```
{
  "$id": "trqp-recognition-response",
  "title": "RecognitionResponse",
  "type": "object",
  "required": [
    "entity_id",
    "action",
    "resource",
    "authority_id",
    "time_evaluated",
    "recognized"
  ],
  "properties": {
    "entity_id": {
        "type": "string",
        "description": "The identifier for the entity that was queried for recognition by
```

```
      },
      "authority_id": {
        "type": "string",
        "description": "The identifier for the authority (e.g. ecosystem governance autho
      },
    "action": {
      "type": "string",
      "description": "The action that the query is checking Recognition for."
    },
    "resource": {
      "type": "string",
      "description": "The resource that the query is checking Recognition for."
    },
    "recognized": {
      "type": "boolean",
      "description": "True if the action+resource has been recognized, false otherwise."
    },
    "time_requested": {
      "type": "string",
      "format": "date-time",
      "description": "the server time that was requested (may be blank), as provided in t
    },
    "time_evaluated": {
      "type": "string",
      "format": "date-time",
      "description": "the server time that the query was executed at on the endpoint serv
    },
    "message": {
      "type": "string",
      "description": "additional details attached to the response about the recognition r
    },
    "context": {
      "type": "object",
      "description": "the context object that was supplied for the query that was evaluat
      "properties": {
        "time": {
          "type": "string",
          "description": "Time for query to use. If blank, uses current server time. RFC3
        }
      },
      "additionalProperties": {
        "type": "string"
      }
    }
```

```
  }
}
```

Example recognition response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "entity_id":    "service-42",
  "authority_id": "did:example",
  "action":       "recognize",
  "resource":     "listed-registry",
  "recognized":   true,
  "time_requested":         "2025-06-19T10:00:00Z",
  "time_evaluated":         "2025-06-19T10:00:00Z",
  "message":      "Service-42 is recognized by did:example."
}
```

source: trqp_recognition_response.schema.json

§ 8. HTTPS Binding

This section is normative.

This section defines the requirements for making TRQP authorization and recognition queries over HTTPS with JSON.

---

§ 8.1 Common Request Headers

All HTTPS calls **MUST** include:

- `Content-Type: application/json`
- Optional tracing header: `X-Request-ID: <uuid>`

> NOTE
>
> Authentication requirements for TRQP endpoints are deployment-specific and determined by the trust registry operator's policy. This specification does not mandate a specific authentication mechanism. See the Implementation Considerations section for guidance.

---

§ 8.2 Authorization over HTTPS

§ 8.2.1 HTTPS Authorization Request

```
POST /authorization HTTP/1.1
Host: registry.example.com
Content-Type: application/json
Authorization: Bearer eyJ...
X-Request-ID: d4f34c12-9b7a-4e3a-a5d1-7e4f8c2c9f10

{
  "entity_id":    "user-1234",
  "authority_id": "auth-service-A",
  "action":       "issue",
  "resource":     "engineer-license",
  "context": {
    "time": "2025-06-19T11:30:00Z"
  }
}
```

§ 8.2.2 HTTPS Authorization Response

A successful authorization call returns HTTP 200 with the JSON body below:

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-ID: d4f34c12-9b7a-4e3a-a5d1-7e4f8c2c9f10

{
  "entity_id":          "user-1234",
  "authority_id":       "auth-service-A",
  "action":             "issue",
  "resource":           "engineer-license",
  "authorized":         true,
  "time_requested":     "2025-06-19T11:30:00Z",
  "time_evaluated":     "2025-06-19T11:30:00Z",
  "message":            "User-1234 holds the admin role.",
  "context": {
    "time": "2025-06-19T11:30:00Z"
  }
}
```

---

§ 8.3 Recognition over HTTPS

§ 8.3.1 HTTPS Recognition Request

```
POST /recognition HTTP/1.1
Host: registry.example.com
Content-Type: application/json
Authorization: Bearer eyJ...
X-Request-ID: bfe9eb29-ab87-4ca3-be83-a1d5d8305716

{
  "entity_id":    "service-42",
  "authority_id": "did:example",
  "action":       "govern",
  "resource":     "professional-engineers",
  "context": {
    "time": "2025-06-19T10:00:00Z"
  }
}
```

§ 8.3.2 HTTPS Recognition Response

A successful recognition returns HTTP 200 with the JSON body below:

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-ID: bfe9eb29-ab87-4ca3-be83-a1d5d8305716

{
  "entity_id":      "service-42",
  "authority_id":   "did:example",
  "action":         "govern",
  "resource":       "professional-engineers",
  "recognized":     true,
  "time_requested": "2025-06-19T10:00:00Z",
  "time_evaluated": "2025-06-19T10:00:00Z",
  "message":        "Service-42 is recognized by auth-master.",
  "context": {
    "time": "2025-06-19T10:00:00Z"
  }
}
```

Error conditions (e.g. malformed JSON, unauthorized, not found) are signaled via standard HTTP 4xx/5xx status codes and a Problem Details JSON body.

---

§ 8.4 Error Handling

- **400 Bad Request** — invalid JSON or missing required fields
- **401 Unauthorized** — missing/invalid bearer token
- **404 Not Found** — entity, authority, action, or resource not recognized
- **500 Internal Server Error** — unexpected server failure

Error responses use the Problem Details for HTTP APIs format:

```
{
  "type":   "https://example.com/problems/invalid-action",
  "title":  "action not found",
  "status": 404,
  "detail": "action \"issue\" is not defined for authority auth-service-A."
}
```

§ 9. Versioning, Extensibility, and Backwards Compatibility

This section is normative.

§ 9.1 Protocol Versioning

1. The TRQP protocol version follows a **major.minor** numbering scheme (e.g., 2.0, 2.1, 3.0).
2. A **major version** change (e.g., 2.x to 3.0) indicates changes that are not backwards-compatible with the prior major version.
3. A **minor version** change (e.g., 2.0 to 2.1) indicates additions or clarifications that remain backwards-compatible with the current major version.
4. Implementations SHOULD include the TRQP specification version they conform to in their documentation or service metadata.

§ 9.2 Extensibility

1. The `context` object in TRQP queries is the primary extensibility mechanism. Because the `context` object permits additional JSON members beyond `time`, TRQP bindings and profiles MAY define additional `context` members for domain-specific query conditions.

2. A TRQP endpoint that receives a `context` member it does not recognize MUST ignore that member and process the query using only the members it supports.
3. Future TRQP bindings MAY define additional query types beyond authorization and recognition, provided they conform to the identifier requirements in the Identifiers section.
4. Extensions MUST NOT redefine or alter the semantics of the fields defined in this specification.

§ 9.3 Backwards Compatibility

The following rules define what constitutes a backwards-compatible change to this specification:

1. **Backwards-compatible changes** (permitted in minor versions):
    1. Adding new OPTIONAL fields to query or response schemas.
    2. Adding new OPTIONAL `context` members.
    3. Adding new query types.
    4. Adding new informative sections or clarifications to normative text that do not change its meaning.
2. **Breaking changes** (requiring a new major version):
    1. Removing or renaming existing required fields in query or response schemas.
    2. Changing the semantics of existing fields.
    3. Adding new required fields to query schemas.
    4. Changing the identifier requirements in a way that invalidates previously conformant identifiers.
3. Implementations conforming to a given major version MUST accept and process queries from any minor version within that major version, ignoring any unrecognized OPTIONAL fields.

§ 10. Conformance

This section is normative.

As well as sections marked *normative*, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this specification are to be interpreted as described in RFC 2119.

§ 10.1 Conformance Targets

This specification defines two conformance targets:

1. **TRQP Endpoint** — a network service that receives TRQP queries and returns TRQP responses on behalf of a trust registry.
2. **TRQP Consumer** — a network client or server that sends TRQP queries to a TRQP endpoint.

A conformant implementation MUST satisfy the requirements for at least one of these targets using at least one TRQP binding.

§ 10.2 TRQP Endpoint Conformance

A conformant TRQP endpoint MUST:

1. Support at least one of the following query types:
    1. **Authorization queries** — accepting queries and returning responses that conform to the authorization query and response schemas defined in this specification.
    2. **Recognition queries** — accepting queries and returning responses that conform to the recognition query and response schemas defined in this specification.
2. Accept and validate identifiers conforming to the requirements in the Identifiers section, including:
    1. All identifiers MUST be represented as single strings conforming to RFC 3986.
    2. The `authority_id` MUST be a globally unique identifier.
    3. The `action` and `resource` MUST each be non-empty strings.
    4. If a `context` object is provided, it MUST be processed as a JSON object. If the `context` includes a `time` parameter, its value MUST be interpreted as an RFC 3339 datetime.
3. Return responses that conform to the response schemas defined in this specification for each supported query type.
4. Make the `ecosystem governance framework` discoverable via the `authority_id` as required in the Identifiers section.

§ 10.3 TRQP Consumer Conformance

A conformant TRQP consumer MUST:

1. Send queries that conform to the query schemas defined in this specification for at least one of the supported query types (authorization, recognition, or both).
2. Construct queries using identifiers that conform to the requirements in the Identifiers section, including:

1. All identifiers MUST be represented as single strings conforming to RFC 3986.
2. The `action` and `resource` MUST each be non-empty strings.
3. If a `context` object is included, it MUST be a JSON object conforming to the requirements in this specification.

3. Be capable of processing responses that conform to the response schemas defined in this specification for each query type it supports.

§ 10.4 TRQP HTTPS Binding Conformance

A conformant implementation of the TRQP HTTPS Binding MUST additionally satisfy the following requirements:

1. All HTTPS requests MUST include a `Content-Type: application/json` header.
2. Authorization queries MUST be sent as `POST /authorization` requests with a JSON body conforming to the authorization query schema.
3. Recognition queries MUST be sent as `POST /recognition` requests with a JSON body conforming to the recognition query schema.
4. TRQP endpoints MUST return error responses using the Problem Details for HTTP APIs format with appropriate HTTP status codes.
5. TRQP endpoints MUST return HTTP 200 for successful queries with a JSON body conforming to the appropriate response schema.

§ 10.5 Test Suite

A formal conformance test suite for TRQP v2.0 does not currently exist. The conformance targets and MUST-level requirements defined in this section are intended to serve as the normative basis for a future test suite.

The Trust Registry Task Force has formally acknowledged this gap and intends to publish a test suite as a companion deliverable to this specification. Contributions toward a test suite are welcome via the GitHub repository.

§ 11. Security Considerations

This section is informative.

All implementers ("TRQP binding" and "TRQP bridge") of TRQP should take the following threats into account and implement appropriate controls:

- **Trust Anchor Hijacking**: Use strong cryptography and rotate keys regularly.

- **Trust Registry Bugs**: Conduct code reviews, vulnerability scans, and robust QA.
- **Trust Anchor Spoofing**: Verify responses using known cryptographic anchors or certificate chains.
- **Domain Hijacking**: Protect DNS entries; if DNS-based discovery is used, consider DNSSEC or other verification.
- **Replay Attacks**: Use timestamps, nonces, and short-lived tokens.
- **Data Integrity**: Sign or hash data at rest, use TLS or equivalent in transit.
- **Denial of Service (DoS)**: Rate-limit queries, monitor usage, scale infrastructure appropriately.
- **Insufficient Data Validation**: Enforce strict schema checks and reject malformed data with clear error messages.
- **Trust Anchor Compromise**: Implement multi-tier trust anchors, and have a plan to revoke or replace compromised keys quickly.
- **Logging and Auditing**: Log all access, changes, and suspicious activities; adopt real-time monitoring.
- **Protocol Downgrade Attacks**: Default to the latest secure version, disallow fallback to insecure versions.
- **Sensitive Data Exposure**: Encrypt sensitive or personally identifiable information, and comply with relevant data protection laws.
- **Timing Attacks**: Where feasible, adopt constant-time operations for cryptographic and authorization checks.

§ 12. Privacy Considerations

This section is informative.

Implementers must design the system so that the handling of authorizations and identity information minimizes the risk of exposing sensitive details. In addition to data minimization and regulatory compliance, pay special attention to the following:

- **Careful Handling of Authorizations & Identities**:
  - Ensure that authorization tokens and identity data (such as DIDs) do not disclose more information than necessary.
  - Avoid exposing internal structures or hierarchies that could be exploited by correlating queries or monitoring network traffic.
- **Correlation Risk Mitigation**:
  - Recognize that even though only authority and entity identifiers are transmitted in recognition queries, combining this data with network-level information (such as IP

addresses) could lead to correlation attacks.

- Consider implementing techniques such as query obfuscation, randomized response timings, or other privacy-preserving measures to reduce the risk of linking requests back to a particular requester.

§ 13. Implementation Considerations

This section is informative.

Implementing TRQP across multiple digital trust ecosystems involves two categories of considerations:

1. Establishing secure bridging mechanisms between heterogeneous governance frameworks.
2. Ensuring trust registries accurately represent the ecosystem model.

Key factors involved in each category:

- **Bridging Across Ecosystems**:
  - **Adapter Development**: As TRQP is designed to bridge various intra-trust frameworks (such as Open ID Federation, X509 Chains, or TRAIN), implementers should develop adapters tailored to their ecosystem. Each adapter translates local trust assertions into the common TRQP authority statements.
  - **Interoperability Focus**: The bridge should speak the “lowest common denominator” across frameworks. This requires abstracting complex internal authorization models into a standardized format that can be reliably queried by external verifiers.
  - **Flexible Query Handling**: Ensure your bridging solution can handle authorization and recognition queries as required. Some systems may only provide recognition (focused on connecting other trust registries) or authorization (focused on internal ecosystem governance). For example, when verifying whether an entity is authorized under a particular ecosystem authority, the adapter must translate internal rules into a compliant TRQP response.
- **Trust Registries & Ecosystem Representation**:
  - **Trust Registry as the Ecosystem State Holder**: A Trust Registry is not the ecosystem itself but a capability within the ecosystem that serves as the authoritative source for its authorization state. It must be capable of answering queries that reflect the current state of authority in the ecosystem.
  - **Ecosystem Model Mapping**:
    - Trust registries should maintain a clear mapping of internal ecosystem authorizations to the standardized TRQP model. This means maintaining up-to-date relationships

between entities, their roles, and the associated authorizations.

- The ecosystem authorization model should be abstracted in a way that hides the underlying complexity, ensuring that verifiers only need to interact with a uniform interface regardless of the diversity of the underlying systems of record and policy frameworks.

- **Bridging Governance & Registries**:
  - The registry should integrate with the broader ecosystem governance framework, adhering to the requirements of this specification for identifier creation (e.g., using compliant identifier strings, URIs, and/or DID methods) and service endpoint specifications.
  - Document how the registry bubbles up the state of authorizations, including how updates and revocations are handled to maintain an accurate and timely reflection of the ecosystem's trust landscape.

- **Authentication**: TRQP does not mandate a specific authentication mechanism for HTTPS endpoints. Authentication requirements are deployment-specific and should be defined by the trust registry operator in their governance framework. Common approaches include bearer tokens (OAuth 2.0), mutual TLS, or open unauthenticated access for public trust registries. Implementers should document their authentication requirements clearly so that TRQP consumers can obtain credentials through the appropriate channel.
- **Additional Bindings**: Additional bindings may be considered, including:
  - **DIDComm TRQP Binding**: establishing a DIDComm-based protocol, including RFCs that may be required for formalization.
  - **TSP TRQP Binding**: establishing a TSP-based protocol, including RFCs that may be required for formalization.

§ 14. References

§ 14.1 Normative References

[RFC 2119]
Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997. https://datatracker.ietf.org/doc/html/rfc2119

[RFC 3339]
Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002. https://datatracker.ietf.org/doc/html/rfc3339

[RFC 3986]

Berners-Lee, T., Fielding, R., and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax”, STD 66, RFC 3986, January 2005. https://datatracker.ietf.org/doc/html/rfc3986

[RFC 7807]

Nottingham, M. and E. Wilde, “Problem Details for HTTP APIs”, RFC 7807, March 2016. https://datatracker.ietf.org/doc/html/rfc7807

§ 14.2 Informative References

[Cedar-PARC]

“Cedar Policy Language — Authorization”, Amazon Web Services. https://docs.cedarpolicy.com/auth/authorization.html

[DID-Core]

Sporny, M., Guy, A., Sabadello, M., and D. Reed, “Decentralized Identifiers (DIDs) v1.0”, W3C Recommendation, July 2022. https://www.w3.org/TR/did-core/

[High-Assurance-DIDs-DNS]

Carter, J. et al., “High Assurance DIDs with DNS”, Internet-Draft draft-carter-high-assurance-dids-with-dns-03, IETF, 2024. Note: This document is an active Internet-Draft and has not yet been published as an RFC. Implementers should consult the IETF datatracker for the latest version. https://www.ietf.org/archive/id/draft-carter-high-assurance-dids-with-dns-03.html

[ToIP-Glossary]

Trust Over IP Foundation, “ToIP Main Glossary”. https://glossary.trustoverip.org/

[ToIP-Governance-Architecture]

Trust Over IP Foundation, “ToIP Governance Architecture Specification V1.0”, December 2021. https://trustoverip.org/wp-content/uploads/ToIP-Governance-Architecture-Specification-V1.0-2021-12-21.pdf

[ToIP-Governance-Metamodel]

Trust Over IP Foundation, “ToIP Governance Metamodel Specification V1.0”, December 2021. https://trustoverip.org/wp-content/uploads/ToIP-Governance-Metamodel-Specification-V1.0-2021-12-21.pdf