O-RAN.WG2.A1AP-v03.01

Technical Specification

O-RAN Working Group 2 (Non-RT RIC and A1 interface WG)

A1 interface: Application Protocol

O-RAN ALLIANCE e.V.
Buschkauler Weg 27, 53347 Alfter, Germany
Register of Associations, Bonn VR 11238
VAT ID DE321720189

Revision History

Date	Revision	Author	Description
2019.09.30	01.00	Patric Lind (Ericsson)	First version with A1-P (Policy Management service)
2020.03.13	01.01	Patric Lind (Ericsson)	Removal of multi-object operations and PATCH based procedures. Included Open API Specification and aligned text with it.
2020.07.20	02.00	John Power (Ericsson)	Defining A1-P/V2 based on policy types
2020.11.09	03.00	Patric Lind (Ericsson)	Defining A1-EI/V1 (A1 Enrichment Information service)
2021.03.13	03.01	Patric Lind (Ericsson)	Separation of application protocol from type definitions. Data models and type definitions moved to A1 interface: Type Definitions v01.00

Contents

Chapter 1 Introductory Material

1.1 Scope

This Technical Specification has been produced by the O-RAN Alliance.

The contents of the present document are subject to continuing work within O-RAN and may change following formal O-RAN approval. Should the O-RAN Alliance modify the contents of the present document, it will be re-released by O-RAN with an identifying change of release date and an increase in version number as follows:

Release xx.yy.zz

where:

xx the first two-digit value is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc. (the initial approved document shall have xx=01).

yy the second two-digit value is incremented when editorial only changes have been incorporated in the document.

zz the third two-digit value is included only in working versions of the document indicating incremental changes during the editing process; externally published documents never have this third two-digit value included.

The present document specifies the application protocol of the A1 interface. It is part of a TS-family covering the O-RAN WG2: A1 interface as identified below: “**General Aspects and Principles**”. “**Transport Protocol**”. “**Application Protocol**”. “**Type Definitions**”.

1.1.1 Compatibility of A1 versions

The version number of the present document indicates that there may be implications for the compatibility between A1 implementations in Non/Near-RT RICs that are based on different versions of this specification.

An incremented first digit of this specification could indicate that a new major feature (e.g. new A1 service) has been added or that an incompatible change has been made to an A1 service. An incremented second digit could indicate that an optional feature has been added, or that clarifications or corrections have been made.

The compatibility of A1 implementations in Non/Near-RT RICs depends on which A1 services that are implemented and which version(s) of each A1 service that are implemented. The version of an A1 service is indicated by the API version in the URI (see chapter 4) and compatibility is governed by the version of the OpenAPI document for the A1 service (see Annex A). The present document handles the service compatibility aspects while A1 interface: Type Definitions [5] handles the compatibility for data types used by the A1 services.

1.2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document.

[1] 3GPP TR 21.905: “3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Vocabulary for 3GPP Specifications”

[2] O-RAN WG2: “Non-RT RIC and A1 interface (Use Case Requirements)”

[3] O-RAN WG2: “A1 interface: General Aspects and Principles”

[4] O-RAN WG2: “A1 interface: Transport Protocol”

[5] O-RAN WG2: “A1 interface: Type Definitions”

[6] 3GPP TS 23.501: “3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; System Architecture for the 5G System; Stage 2”

[7] 3GPP TS 29.501: “3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; 5G System; Principles and Guidelines for Services Definition; Stage 3”

[8] 3GPP 29.xxx-SBI-Stage3-Template, https://www.3gpp.org/ftp/information/All_Templates/29.xxx-SBI-Stage3-Template.zip

[9] 3GPP TS 32.158: “3rd Generation Partnership Project; Technical Specification Group Management and orchestration; Design rules for REpresentational State Transfer (REST) Solution Sets (SS)

[10] 3GPP TS 32.866: “3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Telecommunication management; Study on a REST(REpresentational State Transfer)-ful HTTP-based Solution Set (SS)

[11] IETF RFC8259: “The JavaScript Object Notation (JSON) Data Interchange Format”

[12] Semantic Versioning 2.0.0, https://semver.org

[13] IETF RFC 3986: “Uniform Resource Identifier (URI): Generic Syntax”

[14] IETF RFC7807: “Problem Details for HTTP APIs”

[15] 3GPP TS 29.500: “3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; 5G System; Technical Realization of Service Based Architecture; Stage 3”

[16] OPENAPI initiative, OpenAPI 3.0.1 Specification, http://spec.openapis.org/oas/v3.0.1.html

1.3 Definitions and Abbreviations

1.3.1 Definitions

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

A1 policy	Declarative policy that is based on a policy type, identified by its PolicyId and contains a scope identifier and one or more policy statements.
EI job	Description of enrichment information to be produced and delivered that is identified by its EiJobId and contains a scope identifier and one or more parameters and conditions.
EiJobId	Simple Data Type representing the EI job identifier.
EI job identifier	Identifier of an EI job that is used for requesting and delivering A1 Enrichment Information.
EI job result	The resulting enrichment information delivered based on an EI job.
EI type	The model on which an EI job and its EI job result is based.
EiTypeId	Simple Data Type representing the EI type identifier.
EI type identifier	Identifier of an EI type.
PolicyId	Simple Data Type representing the policy identifier.
policy identifer	Identifier of an A1 policy that is used in policy operations.
PolicyObject	Representation of an A1 policy in JSON format used as payload in HTTP based policy procedures.
policy statement	Expression of a goal in an A1 policy that is related to policy objectives and/or policy resources and is to be applied to/for the entities identified by the scope identifier.

PolicyStatusObject	Representation of the status of an A1 policy in JSON format used as payload in HTTP based policy procedures.
PolicyTypeId	Simple Data Type representing the policy type identifier.
policy type	The model on which a PolicyObject and a PolicyStatusObject is based.
policy type identifier	Identifier of a policy type.
scope identifier	Identifier of what the statements in the policy or the EI job applies to (UE, group of UEs, slice, QoS flow, network resource or combinations thereof).

1.3.2 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1], O-RAN [2,3] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

Id	Identifier
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
KQI	Key Quality Indicator
ML	Machine Learning
QoS	Quality of Service
QoE	Quality of Experience
REST	REpresentational State Transfer
RAN	Radio Access Netework
RT	Real Time
RIC	RAN Intelligent Controller
RRM	Radio Resouirce Management
S-NSSAI	Single Network Slice Selection Assistance Information
SMO	Service Management and Orchestration
SPID	Subscriber Profile IDentity
UE	User Equipment
UEId	UE Identity
URI	Uniform Resource Identifier

Chapter 2 A1 Application Protocol

This document contains a REST method realization of the interface architecture, and policy and EI procedures identified in A1 interface: Generic Aspects and Principles [3]. It is based on HTTP transport as defined in A1 interface: Transport Protocol [4] and an application data model defined in A1 interface: Type Definitions [5].

This definition of the A1 Application Protocol (A1AP) is based on the 3GPP service framework for network functions specified in 3GPP TS 23.501 [6]. It corresponds to a REST-based Solution Set and is based on the structure in the 3GPP specification TS 29.501 [7] and the related TS template [8]. The design patterns for HTTP procedures are based on 3GPP TS 32.158 [9] and the design patterns for JSON objects are based on 3GPP TS 32.866 [10].

Chapter 3 A1 Services

3.1 Introduction

The A1AP contains APIs for the services defined in A1 interface: Generic Aspects and Principles [3]:

A1-P – Policy Management Service;

A1-EI – Enrichment Information Service,

and will in the future contain an API for:

A1-ML – ML Model Management Service.

Note: Service definition and API for A1-ML are FFS.

The A1AP is based on signaling between an A1 service consumer and an A1 service producer residing in the Non-RT RIC or in the Near-RT RIC.

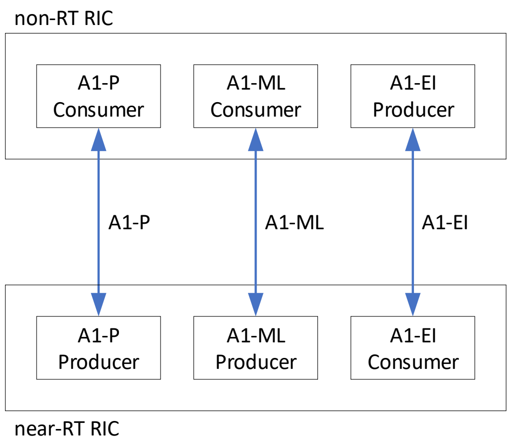

Figure 3.1-1 Service framework for the A1 services.

The interactions between Service Consumer and Service Producer is based on the service framework used for 3GPP Network Functions specified in 3GPP TS 23.501 [6] section 7.1.2 where requests are sent from the Consumer and responses and notifications are sent from the from the Producer. It is the Producer that handles the resources on which the Consumer performs operations. The terms consumer and producer does, thus, not refer to the direction of the data transfer over the A1 interface.

3.2 Policy Management Service

Description of A1 policy, policy statements, policy procedures and A1 policy life cycle aspects are found in A1 interface: Generic Aspects and Priniples [3]. The present document defines a REST based solution set for how to realize A1 policies and perform operations on them over the A1 interface based on the A1 interface: Transport Protocol specification [4] using data types and objects defined in A1 interface: Type Definitions [5].

3.2.1 Service Description

3.2.1.1 Functional elements

The A1AP is based on signaling between the A1-P Consumer residing in the Non-RT RIC and the A1-P Producer residing in the Near-RT RIC. Both the A1-P Consumer and the A1-P Producer contain a HTTP Client and a HTTP Server.

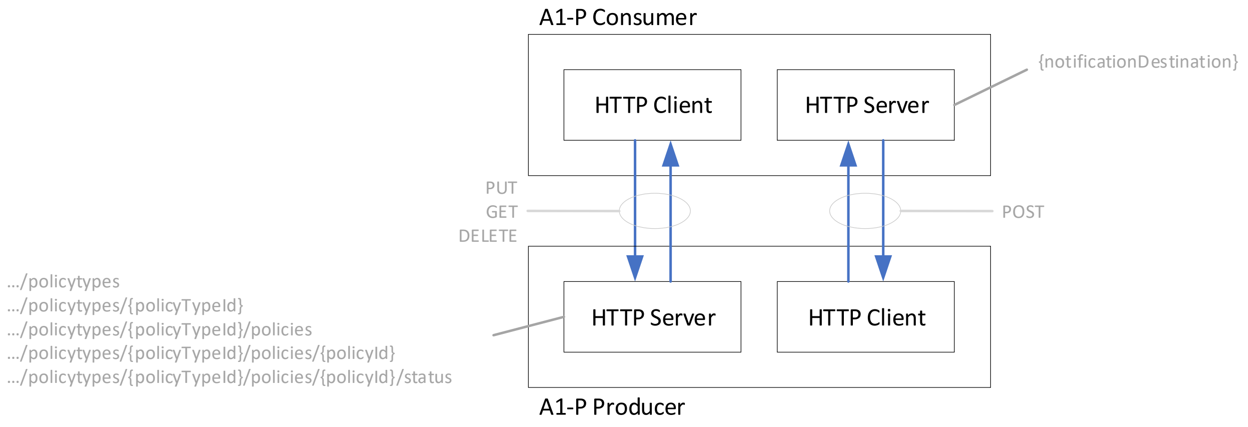
A1-P Consumer
HTTP Client
HTTP Server
{notificationDestination}
PUT
GET
DELETE
POST
…/policytypes
…/policytypes/{policyTypeId}
…/policytypes/{policyTypeId}/policies
…/policytypes/{policyTypeId}/policies/{policyId}
…/policytypes/{policyTypeId}/policies/{policyId}/status
HTTP Server
HTTP Client
A1-P Producer

Figure 3.2-1 HTTP roles in service framework. Arrows indicate direction of HTTP requests sent from HTTP Client to HTTP Server and HTTP responses sent from HTTP Server to HTTP Client.

The A1AP realizes the A1 policy procedures defined in A1 interface: Generic Aspects and Principles [3] using HTTP operations in accordance with A1 interface: Transport Protocol [4] where a policy is represented as a JSON object in accordance with IETF RFC8259 [11] as defined in A1 interface: Type Definitions [5].

3.2.1.2 Policy representation

The following principles are used for A1 policies when JSON is used as resource representation format:

- A policy corresponds to a resource (in the REST sense);
- A policy is represented as a JSON object referred to as a PolicyObject;
- A PolicyObject contains a scope identifier and at least one policy statement (e.g. one or more policy objective statements and/or one or more policy resource statements);
- A policy is identified by a policyId that is included in the URI when operation is for a single policy;
- The policyId is assigned by the A1-P Consumer when the policy is created;
- The A1-P Producer cannot modify or delete a policy;
- Policy feedback for a specifc policy is subscribed to when the policy is created by providing a callback URI in the Create policy operation;
- A PolicyObject does not contain any information related to which internal function in the Near-RT RIC that is to evaluate the policy;
- The A1-P Producer indicates for which policy types policy creation is supported, and the JSON schemas for those policy types can be retrieved by the A1-P Consumer;
- The A1-P Consumer cannot create, modify or delete policy types.

3.2.1.3 Representation objects

The following JSON objects are used within the service operations of the A1-P service:

PolicyTypeObject

The PolicyTypeObject contains the JSON schemas used to validate a PolicyObject and a PolicyStatusObject.

PolicyObject

The PolicyObject is the JSON representation of an A1 policy.

PolicyStatusObject

The PolicyStatusObject is the JSON representation of the enforcement status of an A1 policy.

ProblemDetails

The ProblemDetails object is the JSON representation of the content in a response message with other HTTP error response codes (4xx/5xx).

3.2.1.4 Resource identifiers

The main URI for A1 policy types is:

…/policytypes

A single policy type can be operated upon by adding the value of the policy type identifier to the URI:

…/policytypes/{policyTypeId}

The main URI for A1 policies is:

…/policytypes/{policyTypeId}/policies

A single policy can be operated upon by adding the value of the policy identifier to the URI:

…/policytypes/{policyTypeId}/policies/{policyId}

The main URI for status of a single policy is:

…/policytypes/{policyTypeId}/policies/{policyId}/status

The URI for policy notification is referred to as the notificationDestination and is based on a callback URI provided when creating a policy.

3.2.2 Service Operations

The following table describes the mapping between the A1 policy procedures and the HTTP methods used to realise them.

A1 policy procedure	**HTTP method**
Query all policy type identifiers	GET
Query policy type	GET
Create policy	PUT
Query policy	GET
Query all policy identifiers	GET
Query policy status	GET
Update policy	PUT
Delete policy	DELETE
Feedback policy	POST

Table 3.2-1 A1 policy procedures to HTTP methods mapping.

3.2.2.1 Introduction

The following sections describe the policy operations. For details on the PolicyObjects (in JSON format) transferred in the HTTP message bodies, see A1 Interface: Type Definitions [5].

The policy scope in a PolicyObject contains a scope identifier that can be e.g. a ueId, a groupId or a cellId. The A1-P Consumer needs to map policyIds to scope identifiers in order to manage e.g. all policies applicable to a specific individual ueId. If there are several policies related to the same scope identifier, then several policy operations need to be made to manage that specific scope.

The A1-P Producer allows the A1-P Consumer to create policies of specific types and the A1-P Consumer can discover the supported policy types using the Query policy type procedures. The A1-P Consumer then indicates the policy type identifier when creating or updating a policy and when querying for a specific policy.

3.2.2.2 Create policy

3.2.2.2.1 General

An A1 policy is created using a HTTP PUT request containing a PolicyObject in the payload. The format of the PolicyObject is checked, and the request is either accepted or rejected. If accepted, the policy is to be enforced.

3.2.2.2.2 Create single policy

The operation to create a single policy is based on HTTP PUT. The policy to be created is identified with a URI that includes the policyIdand the message body contains the policyObject.

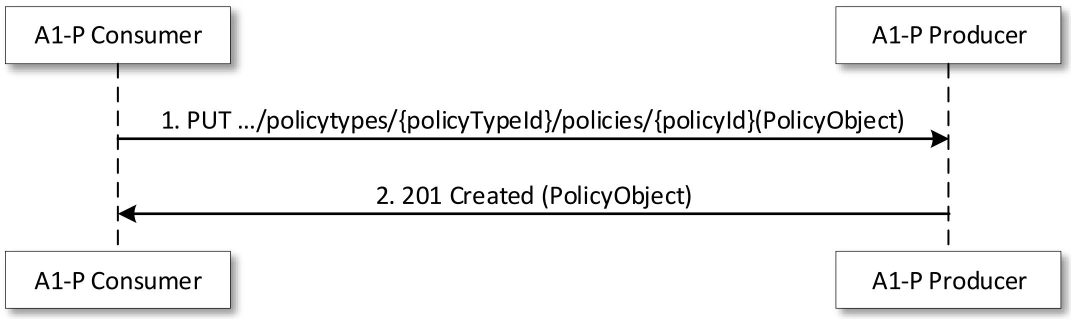
A1-P Consumer
A1-P Producer
1. PUT .../policytypes/{policyTypeId}/policies/{policyId}(PolicyObject)
2. 201 Created (PolicyObject)
A1-P Consumer
A1-P Producer

Figure 3.2.2.2-1 Create policy procedure.

The procedure is as follows:

1) The A1-P Consumer generates the policyId and sends a HTTP PUT request to the A1-P Producer. The target URI identifies the resource (policyId) under which the new policy shall be created. The message body carries a PolicyObject.
2) The A1-P Producer returns the HTTP PUT response. On success, “201 Created” is returned. The location header is present and carries the URI of the new policy and the message body the PolicyObject. On failure, the appropriate error code is returned, and the message body may contain additional error information.

As the A1-P Producer has indicated which policy types it supports, when creating a policy, the A1-P Consumer includes a policyTypeId in the URI for the PUT request. The policyTypeId is used by the A1-P Producer to select the appropriate schemas to use for validation of the PolicyObject and for PolicyStatus.

If the provided policyTypeId is not supported or validation of the PolicyObject fails, “400 Bad Request” is returned and the message body may contain additional error information.

If the A1-P Consumer likes to receive policy status updates related to the created policy, it includes the notificationDestination as a query paremeter in the PUT request.

3.2.2.2.3 Create multiple policies

The operation to create multiple policies is a sequence of operations to create a single policy.

3.2.2.3 Query policy

3.2.2.3.1 General

A1-P Consumer can use the Query policy procedure to read a single policy or to check which policies that exist.

3.2.2.3.2 Query single policy

The operation to query a single policy is based on HTTP GET. The policy to be read is identified with a URI that includes the policyId while the message body is empty, and the response returns the PolicyObject.

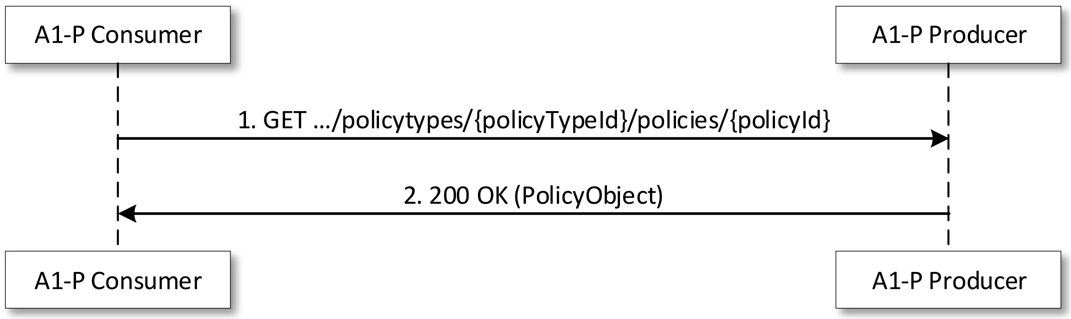

Figure 3.2.2.3-1 Query policy procedure.

The procedure is as follows:

1) The A1-P Consumer sends a HTTP GET request to the A1-P Producer. The target URI identifies the policy to be read based on the policyId under the parent resource "/policytypes/{policyTypeId}/policies". The message body is empty.
2) The A1-P Producer returns the HTTP GET response. On success, "200 OK" is returned. The message body carries a PolicyObject representing the read policy. On failure, the appropriate error code is returned, and the message response body may contain additional error information.

3.2.2.3.3 Query multiple policies

The operation to query multiple policies is a sequence of operations to query a single policy.

NOTE: to query all policies applicable to e.g. a dynamically defined group of UEs, a slice or a cell, the A1-P Consumer need to map the scope identifier to the applicable policyId(s) and make a sequence of requests.

3.2.2.3.4 Query all policies

The operation to query all policies is, for each policy type identifier retrieved as described in clause 3.2.2.7.2, a sequence of operations to query a single policy for each policy identifier retrieved as described in clause 3.2.2.3.5.

3.2.2.3.5 Query all policy identifiers

The operation to query all policy identifiers is based on HTTP GET. The resource to be read is identified within the URI while the message body is empty, and the response returns an array of identifiers representing all available policies of that policy type. The operation has to be performed for each policy type for which policies have been created.

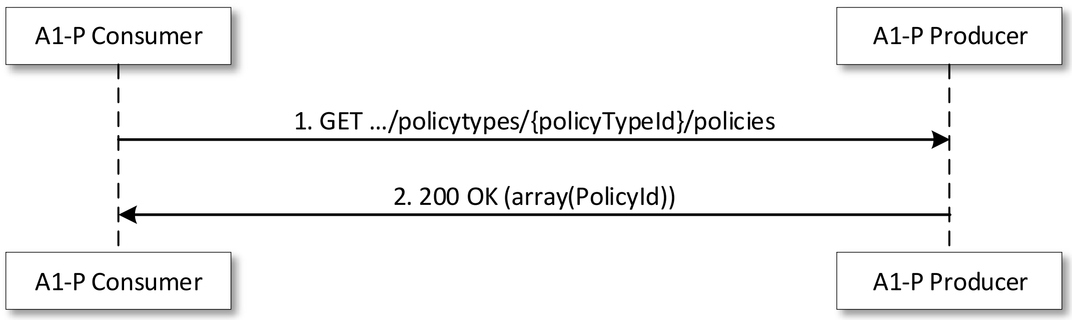
A1-P Consumer
A1-P Producer
1. GET .../policytypes/{policyTypeId}/policies
2. 200 OK (array(PolicyId))
A1-P Consumer
A1-P Producer

Figure 3.2.2.3-3 Query all policy identifiers procedure.

The procedure is as follows:

1) The A1-P Consumer sends a HTTP GET request to the A1-P Producer. The target URI identifies the parent resource "/policytypes/{policyTypeId}/policies". The message body is empty.

2) The A1-P Producer returns the HTTP GET response. On success, “200 OK” is returned. The message body carries an array of PolicyId representing all available policies of the given policy type. On failure, the appropriate error code is returned, and the message response body may contain additional error information.

3.2.2.3.6 Query policy status

The operation to query status for a single policy is based on HTTP GET. The policy for which status is to be read is identified with a URI that includes the policyId while the message body is empty, and the response returns a PolicyStatusObject.

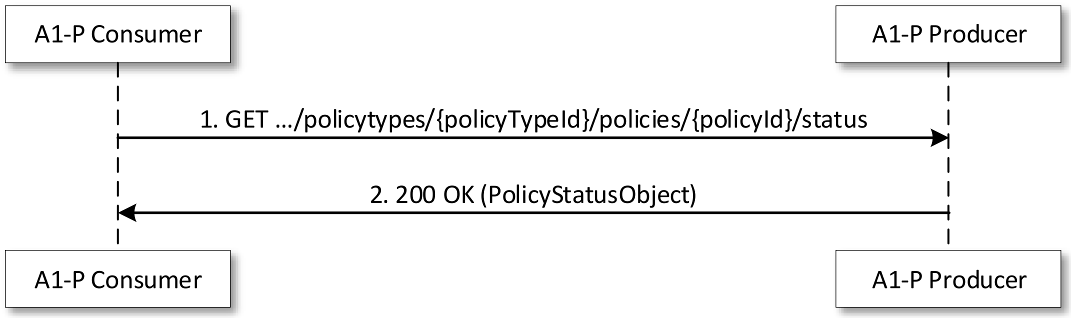
A1-P Consumer
A1-P Producer
1. GET .../policytypes/{policyTypeId}/policies/{policyId}/status
2. 200 OK (PolicyStatusObject)
A1-P Consumer
A1-P Producer

Figure 3.2.2.3-4 Query policy status procedure.

The procedure is as follows:

1) The A1-P Consumer sends a HTTP GET request to the A1-P Producer. The target URI identifies the policy for which status is to be read based on the policyId under the parent resource “/policytypes/{policyTypeId}/policies”. The message body is empty.

2) The A1-P Producer returns the HTTP GET response. On success, “200 OK” is returned. The message body carries a PolicyStatusObject representing the status of the policy. On failure, the appropriate error code is returned, and the message response body may contain additional error information.

3.2.2.4 Update policy

3.2.2.4.1 General

A1-P Consumer can use the Update policy procedure to replace one policy.

3.2.2.4.2 Update single policy

The operation to update a single policy is based on HTTP PUT. The policy to be updated is identified with a URI that includes the policyId and the message body contains the PolicyObject for the updated policy.

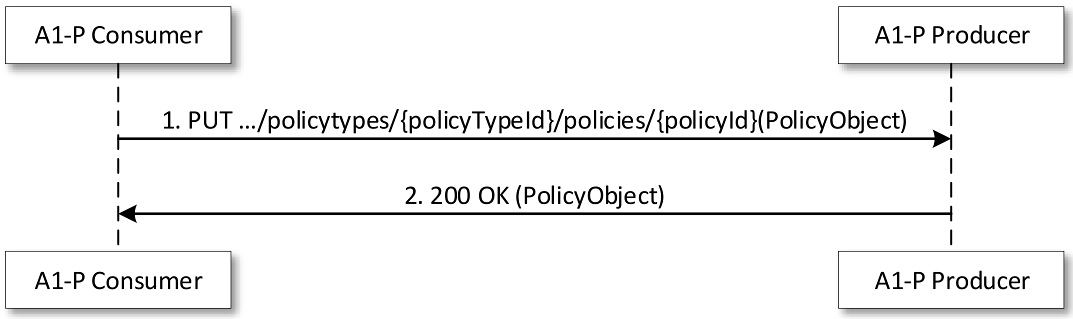

Figure 3.2.2.4-1 Update policy procedure.

The procedure is as follows:

1) The A1-P Consumer sends a HTTP PUT request to the A1-P Producer. The target URI identifies the policy to be updated based on the policyId under the parent resource "/policytypes/{policyTypeId}/policies". The message body contains a PolicyObject.
2) The A1-P Producer returns the HTTP PUT response. On success, "200 OK" is returned. The message body carries a PolicyObject representing the updated policy. On failure, the appropriate error code is returned, and the message response body may contain additional error information.
   NOTE: In case the policy does not exist, the PUT request is handled as a create policy request and "201 Created" is returned. The policyTypeId is used by the A1-P Producer to select the appropriate schemas to use for validation of the PolicyObject and thePolicyStatusObject in the same way as for the Create policy operation.

3.2.2.4.3 Update multiple policies

The operation to update multiple policies is a sequence of operations to update a single policy.

3.2.2.5 Delete policy

3.2.2.5.1 General

A1-P Consumer can use the delete policy procedure to delete a single policy.

3.2.2.5.2 Delete single policy

The operation to delete a single policy is based on HTTP DELETE. The policy to be deleted is identified with a URI that includes the PolicyId. Neither request nor response contain any PolicyObject in the message body.

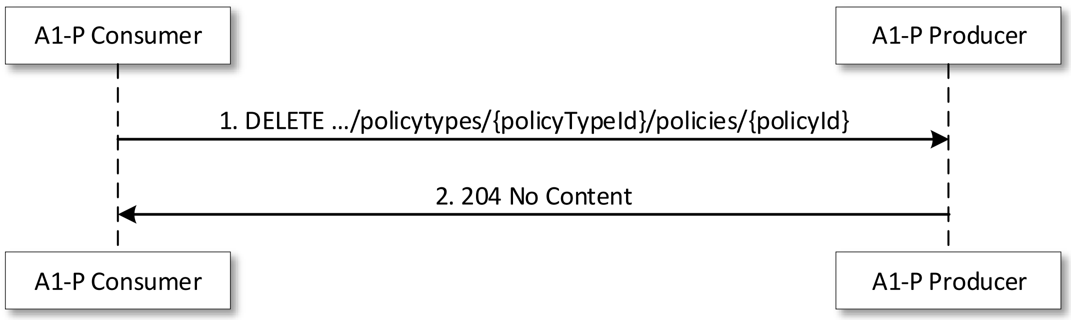

Figure 3.2.2.5-1 Delete policy procedure.

The procedure is as follows:

1) The A1-P Consumer sends a HTTP DELETE request to the A1-P Producer. The target URI identifies the policy to be deleted based on the policyId under the parent resource "/policytypes/{policyTypeId}/policies". The message body is empty.
2) The A1-P Producer returns the HTTP DELETE response. On success, "204 No Content" is returned. The message body is empty. On failure, the appropriate error code is returned, and the message response body may contain additional error information.

3.2.2.5.3 Delete multiple policies

The operation to delete multiple policies is a sequence of operations to delete a single policy.

3.2.2.6 Feedback policy

3.2.2.6.1 General

Feedback policy is an operation that requires the A1-P Producer to have a reduced feature HTTP Client for sending HTTP POST requests and receiving HTTP POST responses. Correspondingly, the A1-P Consumer is required to have a reduced feature HTTP Server for receiving HTTP POST requests and sending HTTP POST responses.

The A1-P Producer uses the Feedback policy operation to notify the A1-P Consumer about:

- Changes in the policy enforcement status for an A1 policy;

All notifications are sent to the URI for notification handling and the PolicyStatusObject contains the information about changes and causes.

3.2.2.6.2 Policy status update

The operation to provide policy feedback is based on HTTP POST. The URI contains the target resource for policy notification handling. The notification content is represented in a PolicyStatusObject that is included in the message body and can contain one notification.

The procedure is used to notify about an enforcement status change of a policy between 'enforced' and 'not enforced'.

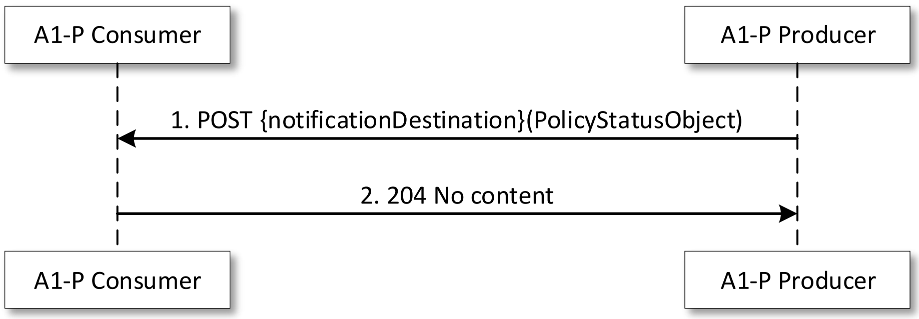

Figure 3.2.2.6-1 Feedback policy procedure.

The procedure is as follows:

1) The A1-P Producer sends a HTTP POST request to the A1-P Consumer. The target URI (notificationDestination) identifies the sink for policy notifications. The message body contains a PolicyStatusObject.
2) The A1-P Consumer returns the HTTP POST response with "204 No Content". The message body is empty.

3.2.2.7 Query policy type

3.2.2.7.1 General

A1-P Consumer can use the Query policy type procedures to check which policy types that are currently supported and to read the schemas for a single policy type.

3.2.2.7.2 Query all policy type identifiers

The operation to query all policy type identifiers is based on HTTP GET. The resource to be read is identified within the URI while the message body is empty, and the response returns an array of identifiers representing all available policy types.

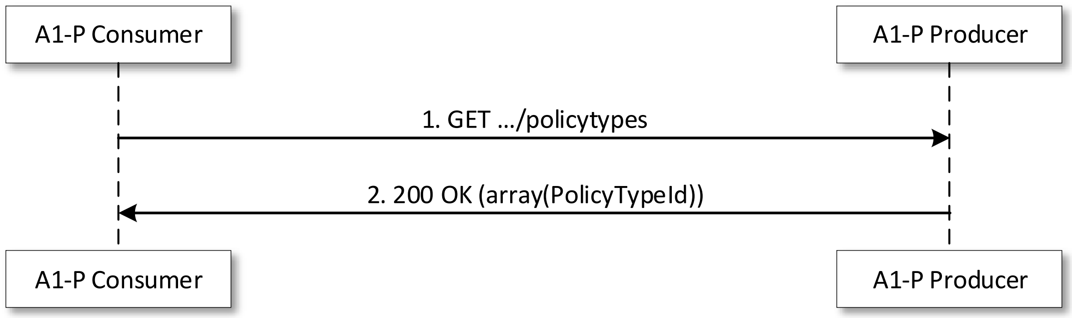

Figure 3.2.2.7-1 Query all policy type identifiers procedure.

The procedure is as follows:

1) The A1-P Consumer sends a HTTP GET request to the A1-P Producer. The target URI identifies the parent resource “/policytypes”. The message body is empty.
2) The A1-P Producer returns the HTTP GET response. On success, “200 OK” is returned. The message body carries an array of PolicyTypeId representing all available policy types. On failure, the appropriate error code is returned, and the message response body may contain additional error information.

3.2.2.7.3 Query single policy type

The operation to query a single policy type is based on HTTP GET. The policy type to be read is identified with a URI that includes the policyTypeId while the message body is empty, and the response returns the policy type object.

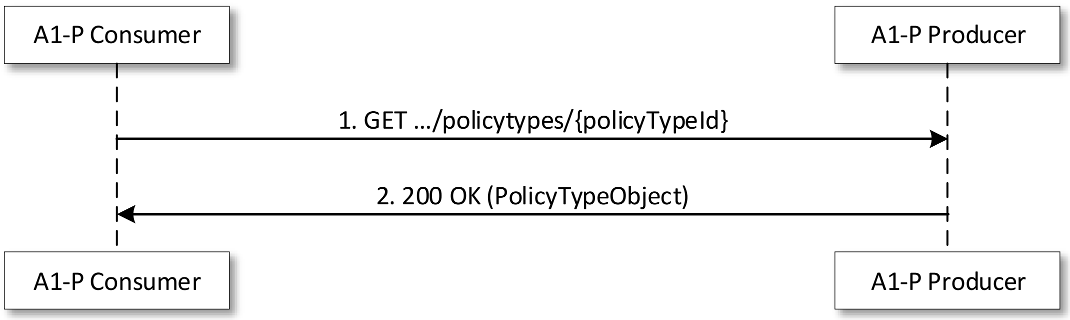
A1-P Consumer
A1-P Producer
1. GET …/policytypes/{policyTypeId}
2. 200 OK (PolicyTypeObject)
A1-P Consumer
A1-P Producer

Figure 3.2.2.7-2 Query policy type procedure.

The procedure is as follows:

1) The A1-P Consumer sends a HTTP GET request to the A1-P Producer. The target URI identifies the policy type to be read based on the policyTypeId under the parent resource “/policytypes”. The message body is empty.
2) The A1-P Producer returns the HTTP GET response. On success, “200 OK” is returned. The message body carries a PolicyTypeObject representing the read policy type. On failure, the appropriate error code is returned and the message response body may contain additional error information.

3.2.2.7.4 Query multiple policy types

The operation to query multiple policy types is a sequence of operations to query a single policy type.

3.2.2.7.5 Query all policy types

The operation to query all policy types is a sequence of operations to query a single policy type for each policy type identifier retrieved as described in clause 3.2.2.7.2.

3.3 Enrichment Information Service

Description of A1 Enrichment Information, EI transfer procedures and EI life cycle aspects are found in A1 interface: Generic Aspects and Principles [3]. This document defines a REST based solution set for how to realize discovery, request and delivery of A1 Enrichment Information over the A1 interface based on the A1 interface: Transport Protocol specification [4] using data types and objects defined in A1 interface: Type Definitions [5].

3.3.1 Service Description

3.3.1.1 Functional elements

The A1-EI service of A1AP is based on signaling between the A1-EI Consumer residing in the Near-RT RIC and the A1-EI Producer residing in the Non-RT RIC. Both the A1-EI Consumer and the A1-EI Producer contains a HTTP Client and a HTTP Server.

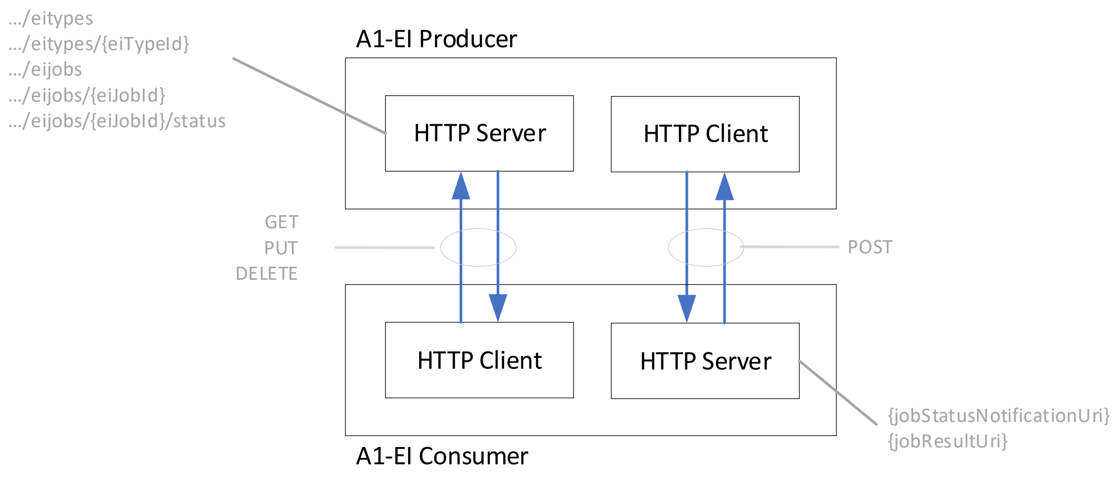

Figure 3.3.1.1-1 HTTP roles in service framework. Arrows indicate direction of HTTP requests sent from HTTP Client to HTTP Server and HTTP responses sent from HTTP Server to HTTP Client.

The A1AP realizes the A1 EI procedures defined in A1 interface: Generic Aspcts and Priniples [3] using HTTP operations in accordance with A1 interface: Transport Protocol [4] where EI types, jobs and job results are represented as JSON objects in accordance with RFC8259 [11] as defined in A1 interface: Type Definitions [5].

3.3.1.2 EI representation

The following principles are used for A1 Enrichment Information when JSON is used as resource representation format:

- The A1-EI Producer can indicate the EI types that are available;
- An EI type is identified by an EI type identifier and the schemas for available EI types can be retrieved by the A1-EI Consumer;
- An EI job can be created for delivery of information of a specific EI type;
- An EI job corresponds to a resource (in the REST sense);
- An EI job, when transferred over HTTP, is represented as a JSON object referred to as an EI job object;
- An EI job object contains a scope identifier and parameters and conditions related to the EI type the delivery is for;
- An EI job is identified by an EI job identifier that is included in the URI when operation is for an EI job;
- The EI job identifier is assigned by the A1-EI Consumer when the EI job is created;
- Status for a specific EI job can be queried and notifications can be subscribed to when the EI job is created by providing a callback URI in the create EI job operation;
- An EI job object does not contain any information related to which source that produces it nor which internal function in the near-RIC that is to consume it;
- EI job results are deliverd to a callback URI provided during create EI job operation;
- Delivered EI that is represented as a JSON object is referred to as an EI job result object.

3.3.1.3 Representation objects

The following JSON objects are used within the service operations of the A1-EI service:

EiTypeObject

The EI type object contains the JSON schemas used to formulate an EI job and interpret an EI job status object and an EI job result object.

EiJobObject

The EI job object is the JSON representation of an EI job.

EiJobStatusObject

The EI job status object is the JSON representation of the status for an EI job.

EiJobResultObject

The EI job result object is the JSON representation of the result delivered during an EI job.

ProblemDetails

The problem details object is the JSON representation of the content in a response message with other HTTP error response codes (4xx/5xx).

3.3.1.4 Resource identifiers

The main URI for A1 enrichment information is:

…/eitypes

A single EI type can be operated upon by adding the value of the EI type identifier to the URI:

…/eitypes/{eiTypeId}

The main URI for A1 EI jobs is:

…/eijobs

A single EI job can be operated upon by adding the value of the EI job identifier to the URI:

…/eijobs/{eiJobId}

The main URI for status of an EI job is:

…/eijobs/{eiJobId}/status

The URI for EI job status notification is referred to as the jobStatusNotificationUri and is based on a callback URI provided when creating an EI job.

The URI for EI delivery is referred to as the jobResultUri and is based on a callback URI provided when creating an EI job.

3.3.2 EI Discovery Service Operations

The following table describes the mapping between the A1 EI discovery procedures, and the HTTP methods used to realise them.

A1 EI procedure	HTTP method
Query EI type identifiers	GET
Query EI type	GET

Table 3.3.2-1 A1 EI procedures to HTTP methods mapping.

3.3.2.1 Introduction

The following sections describe the EI discovery operations. For further information on the EI objects transferred in the HTTP message bodies, see A1 Interface: Type Definitions [5].

The purpose of the EI discovery procedures is for the A1-EI Consumer to

- identify which EI types that are available from the A1-EI producer. Each specific type of enrichment information is identified by a unique EI type identifier (EiTypeId);
- request detailed information related to a specific EI type that can be used to create an EI job and to handle the delivery of results from the EI job.

3.3.2.2 Query EI types

3.3.2.2.1 General

A1-EI Consumer can use the Query EI type identifiers procedure to check which EI types that are available at the A1-EI producer and the Query EI type procedure to request details on a specific EI type.

3.3.2.2.2 Query EI type identifiers

The operation to query EI type identifiers is based on HTTP GET. The resource to be read is identified within the URI while the message body is empty, and the response returns an array of identifiers representing all available EI types.

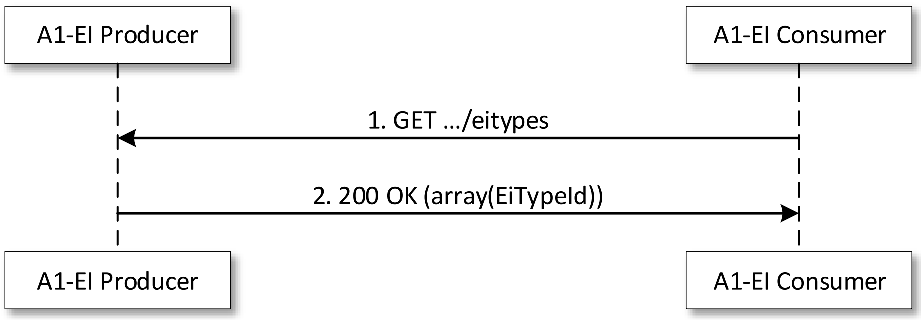

Figure 3.3.3.2.2-1 Query EI type identifiers procedure.

The procedure is as follows:

1) The A1-EI Consumer sends a HTTP GET request to the A1-EI Producer. The target URI identifies the parent resource "/eitypes". The message body is empty.
2) The A1-EI Producer returns the HTTP GET response. On success, "200 OK" is returned. The message body carries an array of EiTypeIds representing all available EI types. On failure, the appropriate error code is returned, and the message response body may contain additional error information.

3.3.2.2.3 Query EI type

The operation to query an EI type is based on HTTP GET. The EI type to be queried is identified with a URI that includes the eiTypeId while the message body is empty, and the response returns the EI type object.

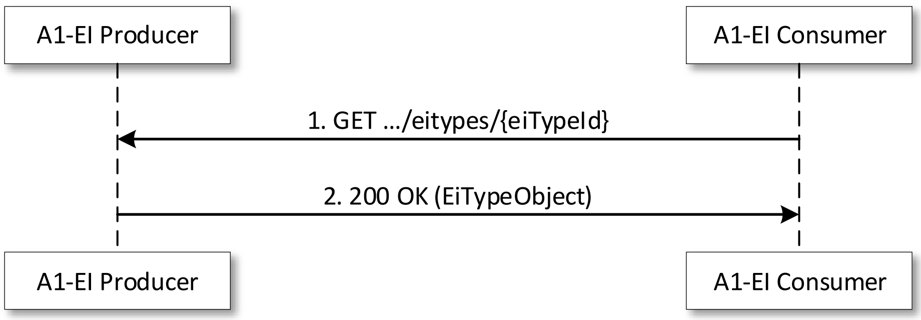

Figure 3.3.3.2.3-1 Query EI type procedure.

The procedure is as follows:

1) The A1-EI Consumer sends a HTTP GET request to the A1-EI Producer. The target URI identifies the EI type to be read based on the eiTypeId under the parent resource "/eitypes". The message body is empty.
2) The A1-EI Producer returns the HTTP GET response. On success, "200 OK" is returned. The message body carries an EITypeObject representing the read EI type. On failure, the appropriate error code is returned, and the message response body may contain additional error information.

The procedure can be used to query that a certain EI type is available for creation of EI jobs even if no detailed information is expected in the EI type object.

3.3.3 EI Job Control Service Operations

The following table describes the mapping between the A1 EI job control procedures and the HTTP methods used to realise them.

A1 EI procedure	HTTP method
Query EI job identifiers	GET
Create EI job	PUT
Query EI job	GET
Update EI job	PUT
Delete EI job	DELETE
Query EI job status	GET
Notify EI job status	POST

Table 3.3.3-1 A1 EI procedures to HTTP methods mapping.

3.3.3.1 Introduction

The following sections describe the EI job control operations. For further information on the EI job objects transferred in the HTTP message bodies, see A1 Interface: Type Definitions [5].

The EI job contains a definition of the content and conditions for the delivery of the EI job result.

The A1-EI Producer allows the A1-EI Consumer to create EI jobs for specific EI types. The A1-EI Consumer can discover the supported EI types using the Query EI types procedures. The A1-EI Consumer then indicates the EI type identifier in all EI job related operations.

3.3.3.2 Query EI jobs

3.3.3.2.1 General

A1-EI Consumer can use the query EI job identifiers procedure to check which EI jobs that exist.

3.3.3.2.2 Query EI job identifiers

The operation to query EI job identifiers is based on HTTP GET. The resource to be read is identified within the URI while the message body is empty, and the response returns an array of identifiers representing all available EI jobs. The operation can be performed for each EI type for which EI jobs have been created, or for all created EI jobs.

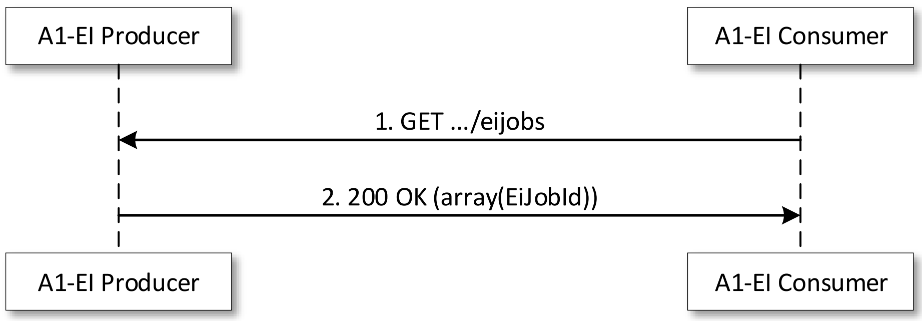

Figure 3.3.3.2.2-1 Query EI job identifiers procedure.

The procedure is as follows:

1) The A1-EI Consumer sends a HTTP GET request to the A1-EI Producer. The target URI identifies the parent resource "/eijobs". The message body is empty.
2) The A1-EI Producer returns the HTTP GET response. On success, "200 OK" is returned. The message body carries an array of EIJobIdentitiers representing all available EI jobs of the given EI type, or of all EI types. On failure, the appropriate error code is returned, and the message response body may contain additional error information.

If the A1-EI Consumer likes to receive EI job identifiers only related to a specific EI type, it includes the eiTypeId as a query parameter in the GET request.

3.3.3.3 Manage EI jobs

3.3.3.3.1 General

The operation to manage an EI job is based on an eiJobId created by the the A1-EI Consumer. The resource URI containing the eiJobId is used in operations to create, query, update and delete an EI job.

3.3.3.3.2 Create EI job

The operation to create an EI job is based on HTTP PUT with an EI job object in the payload. The format of the EI job object is checked, and the request is either accepted or rejected. If accepted, delivery of EI results will start based on the content and conditions defined in the EI job.

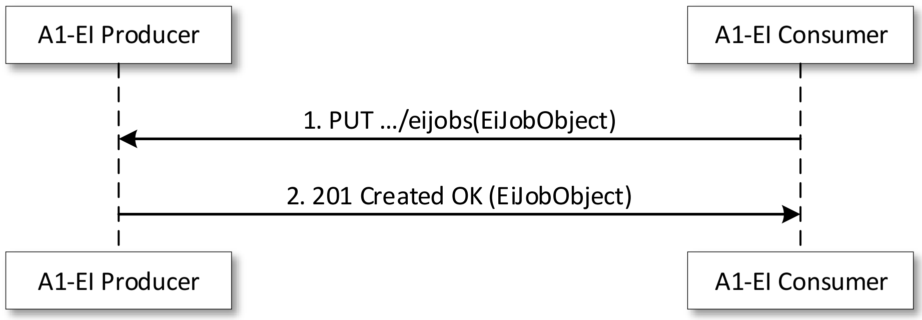
A1-EI Producer
A1-EI Consumer
1. PUT .../eijobs(EiJobObject)
2. 201 Created OK (EiJobObject)
A1-EI Producer
A1-EI Consumer

Figure 3.3.3.3.2-1 Create EI job procedure.

The procedure is as follows:

1) The A1-EI Consumer generates the eiJobId and sends a HTTP PUT request to the A1-EI Producer. The target URI identifies the resource ("/eijobs") under which the new EI job shall be created. The message body carries an EI job object.

2) The A1-EI Producer returns the HTTP PUT response. On success, "201 Created" is returned. The location header is present and carries the URI of the new EI job and the message body carries the EIJobObject. On failure, the appropriate error code is returned, and the message body may contain additional error information.

As the A1-EI Producer has indicated which EI types it supports, when creating an EI job the A1-EI Consumer includes an eiTypeId in the EiJobObject. The eiTypeId is used by the A1-EI Producer to select the appropriate schemas to use for validation of the EI job object and for EI job status.

If the provided eiTypeId is not supported or validation of the EI job object fails, "400 Bad Request" is returned and the message body may contain additional error information.

If the A1-EI Consumer likes to receive EI job status updates related to the created EI job, it includes the jobStatusNotificationUri in the EiJobObject.

3.3.3.3.3 Query EI job

The operation to query a single EI job is based on HTTP GET. The EI job to be read is identified with a URI that includes the eiJobId while the message body is empty, and the response returns the EI job object.

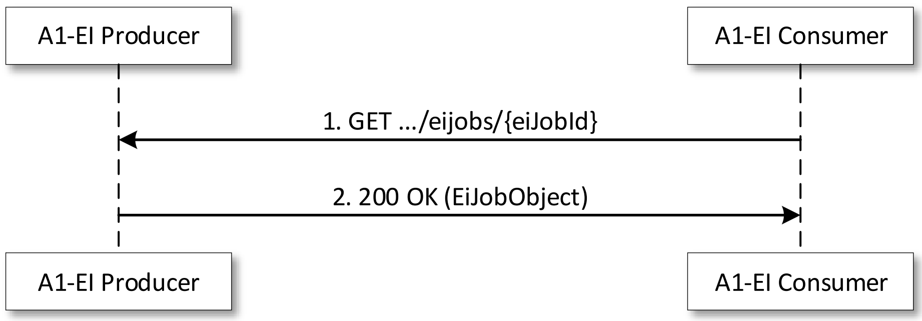

Figure 3.3.3.3.3-1 Query EI job procedure.

The procedure is as follows:

1) The A1-EI Consumer sends a HTTP GET request to the A1-EI Producer. The target URI identifies the EI job to be read based on the eiJobId under the parent resource "/eijobs". The message body is empty.
2) The A1-EI Producer returns the HTTP GET response. On success, "200 OK" is returned. The message body carries an EIJobObject representing the read EI job. On failure, the appropriate error code is returned, and the message response body may contain additional error information.

3.3.3.3.4 Update EI job

The operation to update a single EI job is based on HTTP PUT. The EI job to be updated is identified with a URI that includes the eiJobId and the message body contains the EI job object for the updated EI job.

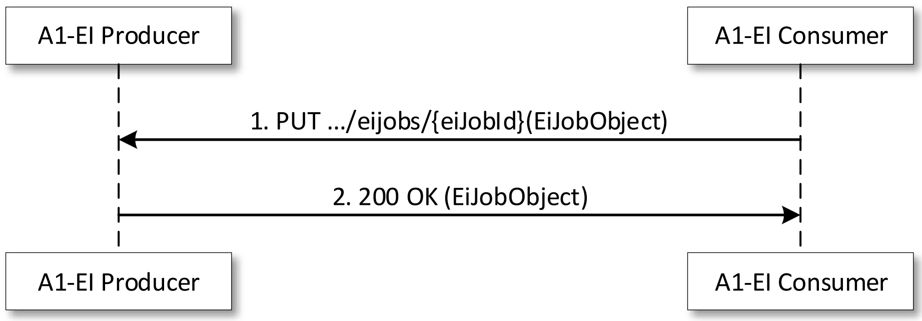

Figure 3.3.3.3.4-1 Update EI job procedure.

The procedure is as follows:

1) The A1-EI Consumer sends a HTTP PUT request to the A1-EI Producer. The target URI identifies the EI job to be updated based on the eiJobId under the parent resource "/eijobs". The message body contains an EI Job object.
2) The A1-EI Producer returns the HTTP PUT response. On success, "200 OK" is returned. The message body carries an EIJobObject representing the updated EI job. On failure, the appropriate error code is returned, and the message response body may contain additional error information.
   NOTE: In case the EI job does not exist, "404 Not Found" is returned.

3.3.3.3.5 Delete EI job

The operation to delete an EI job s based on HTTP DELETE. The EI job to be deleted is identified with a URI that includes the eiJobId. Neither request nor response contain any EI job object in the message body.

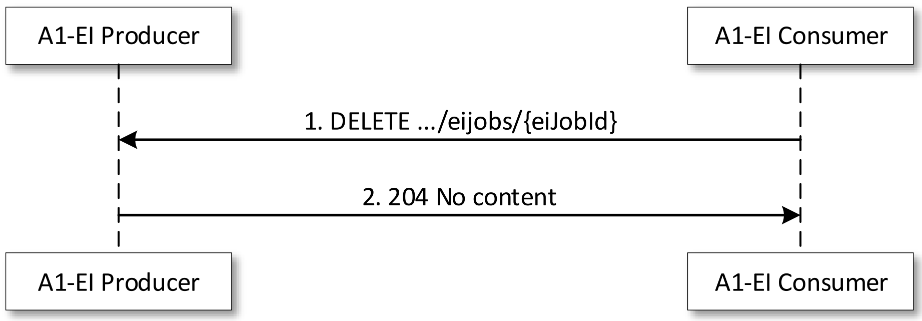

Figure 3.3.3.3.5-1 Delete EI job procedure.

The procedure is as follows:

1) The A1-EI Consumer sends a HTTP DELETE request to the A1-EI Producer. The target URI identifies the EI job to be deleted based on the eiJobId under the parent resource "/eijobs". The message body is empty.

2) The A1-EI Producer returns the HTTP DELETE response. On success, "204 No Content" is returned. The message body is empty. On failure, the appropriate error code is returned, and the message response body may contain additional error information.

3.3.3.4 Status of EI jobs

3.3.3.4.1 General

The A1-EI Consumer can query the A1-EI Producer for the status of an EI job. The query is made by adding "/status" to the URI of the EI job resource.

The A1-EI Producer uses the notify EI job status operation to notify the A1-EI Consumer about changes in status of an EI job. All notifications are sent to the URI for notification handling provided during EI job creation and the EiJobStatusObject contains the information about the status of the EI job.

3.3.3.4.2 Query EI job status

The operation to query status for an EI job is based on HTTP GET. The EI job for which status is to be read is identified with a URI that includes the eiJobId while the message body is empty, and the response returns an EI job status object.

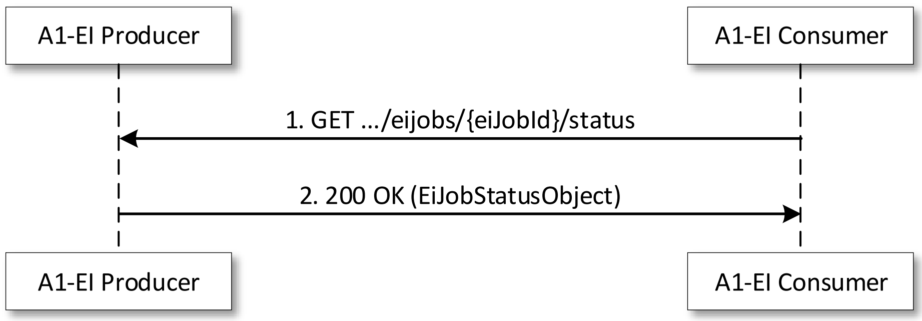
A1-EI Producer
A1-EI Consumer
1. GET .../eijobs/{eiJobId}/status
2. 200 OK (EiJobStatusObject)
A1-EI Producer
A1-EI Consumer

Figure 3.3.3.4.2-1 Query EI job status procedure.

The procedure is as follows:

1) The A1-EI Consumer sends a HTTP GET request to the A1-EI Producer. The target URI identifies the EI job for which status is to be read based on the eiJobId under the parent resource "/eijobs". The message body is empty.

2) The A1-EI Producer returns the HTTP GET response. On success, "200 OK" is returned. The message body carries an EI job status object representing the status of the EI job. On failure, the appropriate error code is returned, and the message response body may contain additional error information.

3.3.3.4.3 Notify EI job status

The operation to notify EI job status is based on HTTP POST. The URI contains the target resource for EI job notification handling. The notification content is represented in an EI job status object that is included in the message body and can contain one notification.

The procedure is used to notify about a status change of an EI job.

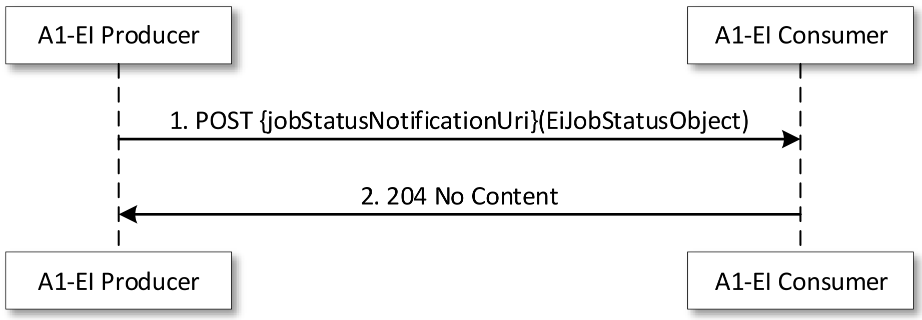

Figure 3.3.3.4.3-1 Notify EI job status procedure.

The procedure is as follows:

1) The A1-EI Producer sends a HTTP POST request to the A1-EI Consumer. The target URI (jobStatusNotificationUri) identifies the sink for EI job status notifications. The message body contains an EI job status object.

2) The A1-EI Consumer returns the HTTP POST response with “204 No Content”. The message body is empty.

3.3.4 EI Delivery Service Operations

The following table describes the mapping between the A1 EI delivery procedures, and the HTTP methods used to realise them.

A1 EI procedure	HTTP method
Deliver EI job result	POST

Table 3.3.4-1 A1 EI procedures to HTTP methods mapping.

3.3.4.1 Introduction

The following sections describe the EI delivery operations. For further information on the EI job result objects transferred in the HTTP message bodies, see A1 Interface: Type Definitions [5].

The purpose of the EI delivery procedures is for the A1-EI Producer to set up appropriate connections and deliver EI job results according to the service description agreed during job creation. The URL to which the EI job result is delivered is transferred from the A1-EI consumer in the EI job object.

3.3.4.2 Push based delivery

3.3.4.2.1 General

The push-based delivery method of EI is based on subscribe-notify paradigm where the EI job creation corresponds to the subscription and the EI delivery is made using HTTP POST in the same way as notifications.

During an EI job, the EI job results can be delivered in a single push or in several that are repeated with regular intervals or irregularly based on events.

3.3.4.2.2 Deliver EI job result

The operation to deliver EI job result is based on HTTP POST. The URI contains the target resource for EI job result handling. The delivered content is represented by an EI job result object.

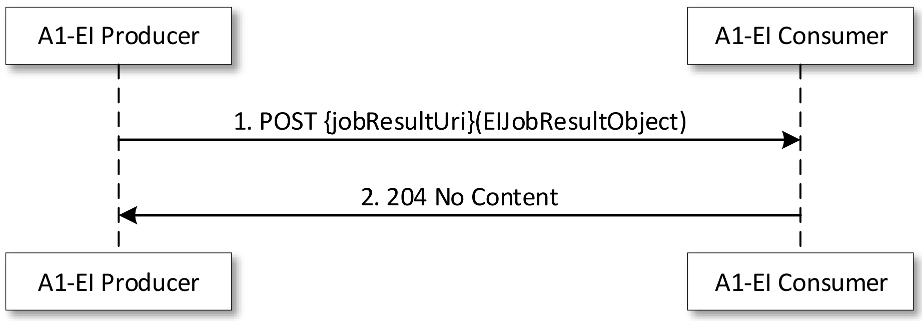

Figure 3.3.4.2.2-1 Deliver EI job result procedure.

The procedure is as follows:

1) The A1-EI Producer sends a HTTP POST request to the A1-EI Consumer. The target URI (jobResultUri) identifies the sink for EI job result delivieries. The message body contains an EI job result object.
2) The A1-EI Consumer returns the HTTP POST response with "204 No Content". The message body is empty.

3.4 ML Model Management Service

No explicit ML Model service operations are defined in this version of the specification.

Chapter 4 API Definitions

4.1 Introduction

4.1.1 Encoding of attributes in A1 data types

Identifiers and parameters that has been defined as integers are, when used over the A1 interface, encoded as JSON `"number"`.

Identifiers that have a hexadecimal or octet string representation are, when used over the A1 interface, encoded as JSON `"string"` with character ordering preserved and zeros filling rules followed.

4.1.2 Compatibility of API versions for A1 services

The API version and API name for each of the A1 services are defined in the following chapters. The API version is a single digit that corresponds to the major version of the corresponding OpenAPI document in Annex A. Based on the versioning rules for the OpenAPI documents, this implies that implementations of an A1 service in the Non/Near-RT RICs are

- compatible if the API version is the same and any difference between the sets of supported features is handled within the API version itself;
- not compatible in case the API versions are different.

The history of the introduction of an A1 service, and new API versions, is captured in the revision history of the present specification. The services and versions specified in the present version of the specification is summarized in clause A.1.2.

Note: Non/Near-RT RIC products that implement various API versions of an A1 service can still be made compatible as is it possible to support several API versions of an A1 service at the same time since each version of an A1 service is addressed by separate URIs.

4.2 A1-P (policy management)

This section contains the definition of the REST based API for the Policy Management Service referred to as A1-P.

4.2.1 Introduction

The A1-P service shall use the A1-P API.

The present version of the present specification defines API version 2 (v2) of the A1-P API.

Based on the URI structure defined in clause 4.4.1 of 3GPP TS 29.501 [7], the request URI used in HTTP request from the A1-P consumer towards the A1-P producer shall have the following structure:

{apiRoot}/A1-P/v2/<ResourceUriPart>

where the "ResourceUriPart" shall be as be defined in subclause 4.2.3.

4.2.2 Usage of HTTP

4.2.2.1 General

The A1 Transport, HTTP protocol and security requirements, is described in A1 interface: Transport Protocol [4].

4.2.2.2 HTTP standard headers

Note: the encodings and applicable MIME media type for the related Content-Type header are not specified in the current version.

4.2.2.3 HTTP custom headers

No HTTP custom headers are introduced in this version of the specification.

4.2.3 Resources

4.2.3.1 Overview

The resource URI structure for the A1-P API is illustrated in figure 4.2.3-1.

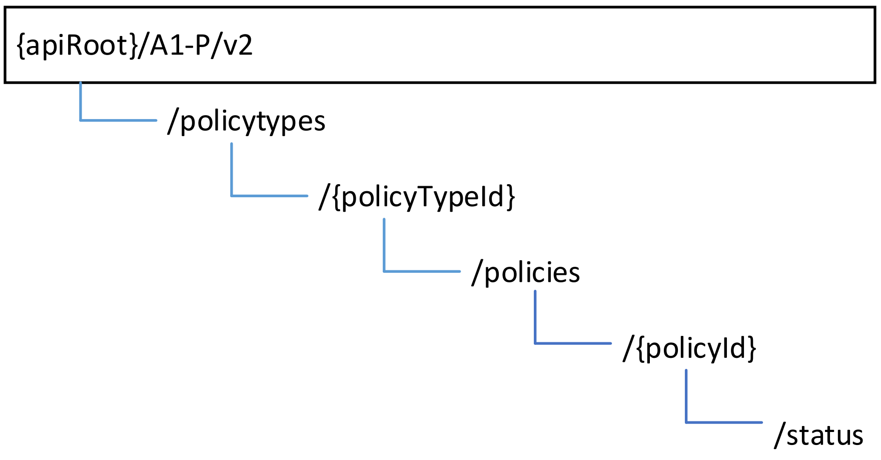

Figure 4.2.3-1: Resource URI structure of the A1-P API

Table 4.2.3-1 provides an overview of the resources and applicable HTTP methods.

Table 4.2.3-1: Resources and methods overview

Resource name	Resource URI	HTTP method or custom operation	Description
All Policy Type Identifiers	/policytypes	GET	Query all policy type identifiers
Individual Policy Type Object	/policytypes/{policyTypeId}	GET	Query policy type
Individual Policy Object	/policytypes/{policyTypeId}/policies/{policyId}	PUT	Create policy, Update policy
		GET	Query policy
		DELETE	Delete policy
Individual Policy Status Object	/policytypes/{policyTypeId}/policies/{policyId}/status	GET	Query policy status
All Policy Identifiers	/policytypes/{policyTypeId}/policies	GET	Query all policy identifiers of a given policy type
Notify Policy Status	{notificationDestination}	POST	Notify status

4.2.3.1.1 Policy type identifier

The PolicyTypeId is constructed based on two parts separated by "_" (underscore):

`typename_version`

where

`typename` is the unique label of the policy type;

`version` is the version of the policy type defined as major.minor.patch as described in SemVer [12].

The typename and version is assigned, and their uniqueness ensured, by the organizational entity that is responsible for the definition and maintenance of the policy type definition.

Note: the typename can be based on a prefix that indicates the organizational entity (e.g. ORAN or a company designator) and a text string that can be descriptive of the class, use case or variant of the policy type.

4.2.3.2 Individual Policy Object

The name of the resource is the PolicyId assigned by the A1-P Consumer when the policy is created.

4.2.3.2.1 Description

The resource represents an A1 policy.

4.2.3.2.2 Resource Definition

The Resource URI and the supported resource variables are as defined in previous sections.

4.2.3.2.3 Resource Standard Methods

The following subclauses specifies the standard methods supported by the resource.

4.2.3.2.3.1 HTTP PUT

This method shall support the request data structures specified in table 4.2.3.2.3.1-1 and the response data structures and response codes specified in table 4.2.3.2.3.1-2.

Table 4.2.3.2.3.1-1: Data structures supported by the HTTP PUT Request Body on this resource

Data type	P	Cardinality	Description
PolicyObject	M	1	Create policy

Table 4.2.3.2.3.1-2: Data structures supported by the HTTP PUT Response Body on this resource

Data type	P	Cardinality	Response codes	Description
PolicyObject	M	1	201 Created 200 OK	Confirmation of created or updated policy
ProblemDetails	O	0..1	4xx/5xx	Detailed problem description

This method shall support the URI query parameters specified in table 4.2.3.2.3.1-3.

Table 4.2.3.2.3.1-3: URI query parameters supported by the HTTP PUT method on this resource

Name	Data type	P	Cardinality	Description	Applicability
notificationDestination	string	O	0..1	Transfer of URL for notifications	Status notifications

4.2.3.2.3.2 HTTP GET

This method shall support the request data structures specified in table 4.2.3.2.3.2-1 and the response data structures and response codes specified in table 4.2.3.2.3.2-2.

Table 4.2.3.2.3.2-1: Data structures supported by the HTTP GET Request Body on this resource

Data type	P	Cardinality	Description
N/A		0	There is no object in the message body of a GET request

Table 4.2.3.2.3.2-2: Data structures supported by the HTTP GET Response Body on this resource

Data type	P	Cardinality	Response codes	Description
PolicyObject	M	1	200 OK	Requested policy object
ProblemDetails	O	0..1	4xx/5xx	Detailed problem description

4.2.3.2.3.3 HTTP DELETE

This method shall support the request data structures specified in table 4.2.3.2.3.3-1 and the response data structures and response codes specified in table 4.2.3.2.3.3-2.

Table 4.2.3.2.3.3-1: Data structures supported by the HTTP DELETE Request Body on this resource

Data type	P	Cardinality	Description
N/A			There is no object in the message body of a DELETE request

Table 4.2.3.2.3.3-2: Data structures supported by the HTTP DELETE Response Body on this resource

Data type	P	Cardinality	Response codes	Description
N/A			204 No content	Confirmation of successful deletion
ProblemDetails	O	0..1	4xx/5xx	Detailed problem description

4.2.3.2.3.4 HTTP POST

This method is not supported on the resource.

4.2.3.2.4 Resource Custom Operations

No custom operations are defined.

4.2.3.3 Individual Policy Status Object

4.2.3.3.1 Description

The resource represents the status of an A1 policy.

4.2.3.3.2 Resource Definition

The Resource URI and the supported resource variables are as defined in previous sections.

4.2.3.3.3 Resource Standard Methods

The following subclauses specifies the standard methods supported by the resource.

Note: URI query parameters are not specified in the current version.

4.2.3.3.3.1 HTTP PUT

Method is not supported on this resource.

4.2.3.3.3.2 HTTP GET

This method shall support the request data structures specified in table 4.2.3.3.3.2-1 and the response data structures and response codes specified in table 4.2.3.3.3.2-2.

Table 4.2.3.3.3.2-1: Data structures supported by the HTTP GET Request Body on this resource

Data type	P	Cardinality	Description
N/A			There is no object in the message body of a GET request

Table 4.2.3.3.3.2-2: Data structures supported by the HTTP GET Response Body on this resource

Data type	P	Cardinality	Response codes	Description
PolicyStatusObject	"M"	1	200 OK	Requested policy status object
ProblemDetails	"O"	0..1	4xx/5xx	Detailed problem description

4.2.3.3.3.3 HTTP DELETE

Method is not supported on this resource.

4.2.3.3.3.4 HTTP POST

Method is not supported on this resource.

4.2.3.3.4 Resource Custom Operations

No custom operations are defined.

4.2.3.4 All Policy Identifiers

4.2.3.4.1 Description

The resource represents A1 policy identifiers.

4.2.3.4.2 Resource Definition

The Resource URI and the supported resource variables are as defined in previous sections.

4.2.3.4.3 Resource Standard Methods

The following subclauses specifies the standard methods supported by the resource.

Note: URI query parameters are not specified in the current version.

4.2.3.4.3.1 HTTP PUT

Method is not supported on this resource.

4.2.3.4.3.2 HTTP GET

This method shall support the request data structures specified in table 4.2.3.6.3.2-1 and the response data structures and response codes specified in table 4.2.3.6.3.2-2.

Table 4.2.3.4.3.2-1: Data structures supported by the HTTP GET Request Body on this resource

Data type	P	Cardinality	Description
N/A			There is no object in the message body of a GET request

Table 4.2.3.4.3.2-2: Data structures supported by the HTTP GET Response Body on this resource

Data type	P	Cardinality	Response codes	Description
array(PolicyId)	M	0..N	200 OK	All policy identifiers
ProblemDetails	O	0..1	4xx/5xx	Detailed problem description

4.2.3.4.3.3 HTTP DELETE

Method is not supported on this resource.

4.2.3.4.3.4 HTTP POST

Method is not supported on this resource.

4.2.3.4.4 Resource Custom Operations

No custom operations are defined.

4.2.3.5 All Policy Type Identifiers

4.2.3.5.1 Description

The resource represents A1 policy type identifiers.

4.2.3.5.2 Resource Definition

The Resource URI and the supported resource variables are as defined in previous sections.

4.2.3.5.3 Resource Standard Methods

The following subclauses specifies the standard methods supported by the resource.

Note: URI query parameters are not specified in the current version.

4.2.3.5.3.1 HTTP PUT

Method is not supported on this resource.

4.2.3.5.3.2 HTTP GET

This method shall support the request data structures specified in table 4.2.3.5.3.2-1 and the response data structures and response codes specified in table 4.2.3.5.3.2-2.

Table 4.2.3.5.3.2-1: Data structures supported by the HTTP GET Request Body on this resource

Data type	P	Cardinality	Description
N/A			There is no object in the message body of a GET request

Table 4.2.3.5.3.2-2: Data structures supported by the HTTP GET Response Body on this resource

Data type	P	Cardinality	Response codes	Description
array(PolicyTypeId)	M	0..N	200 OK	All policy type identifiers
ProblemDetails	O	0..1	4xx/5xx	Detailed problem description

4.2.3.5.3.3 HTTP DELETE

Method is not supported on this resource.

4.2.3.5.3.4 HTTP POST

Method is not supported on this resource.

4.2.3.5.4 Resource Custom Operations

No custom operations are defined.

4.2.3.6 Individual Policy Type Object

4.2.3.6.1 Description

The resource represents an A1 policy type.

4.2.3.6.2 Resource Definition

The Resource URI and the supported resource variables are as defined in previous sections.

4.2.3.6.3 Resource Standard Methods

The following subclauses specifies the standard methods supported by the resource.

4.2.3.6.3.1 HTTP PUT

Method is not supported on this resource.

4.2.3.6.3.2 HTTP GET

This method shall support the request data structures specified in table 4.2.3.6.3.2-1 and the response data structures and response codes specified in table 4.2.3.6.3.2-2.

Table 4.2.3.6.3.2-1: Data structures supported by the HTTP GET Request Body on this resource

Data type	P	Cardinality	Description
N/A			There is no object in the message body of a GET request

Table 4.2.3.6.3.2-2: Data structures supported by the HTTP GET Response Body on this resource

Data type	P	Cardinality	Response codes	Description
PolicyTypeObject	M	1	200 OK	Requested policy type object
ProblemDetails	O	0..1	4xx/5xx	Detailed problem description

4.2.3.6.3.4 HTTP DELETE

This method is not supported on the resource.

4.2.3.6.3.5 HTTP POST

This method is not supported on the resource.

4.2.3.6.4 Resource Custom Operations

No custom operations are defined.

4.2.4 Custom Operations without associated resources

No custom operations are defined.

4.2.5 Notifications

4.2.5.1 Notify Policy Status

4.2.5.1.1 Description

The resource represents the destination for policy status notifications.

4.2.5.1.2 Resource Definition

The Resource URI is a callback URI provided as a query parameter in URL when creating a policy.

4.2.5.1.3 Resource Standard Methods

The following subclauses specifies the standard methods supported by the resource.

Note: URI query parameters are not specified in the current version.

4.2.5.1.3.1 HTTP PUT

Method is not supported on this resource.

4.2.5.1.3.2 HTTP GET

Method is not supported on this resource.

4.2.5.1.3.3 HTTP DELETE

Method is not supported on this resource.

4.2.5.1.3.4 HTTP POST

This method shall support the request data structures specified in table 4.2.5.1.3.4-1 and the response data structures and response codes specified in table 4.2.5.1.3.4-2.

Table 4.2.5.1.3.4-1: Data structures supported by the HTTP POST Request Body on this resource

Data type	P	Cardinality	Description
PolicyStatusObject	M	1	Notify policy

Table 4.2.5.1.3.4-2: Data structures supported by the HTTP POST Response Body on this resource

Data type	P	Cardinality	Response codes	Description
N/A			204 No content	Confirmation of received notification

4.2.6 Data Model

This subclause specifies the application protocol data model supported by the A1-P API.

The data model for the data types transported in the A1-P procedures is defined in A1 Interface: Type Definitions [5].

4.2.6.1 Simple data types and enumerations

This subclause defines simple data types and enumerations that can be referenced from procedures defined in the previous subclauses.

4.2.6.1.1 Simple data types

The URI for policy operations containing a policy object contains a PolicyId attribute.

Table 4.2.6.1.1-1: General definition of simple data types

Type Name	Type Definition	Description	Applicability
PolicyTypeId	string	policy type identifier assigned by the owner of a policy type definition (see A1 interface: Type Definitions [5]))	used in URI
PolicyId	string	policy identifier assigned by the A1-P Consumer when a policy is created	used in URI

4.2.6.2 Structured data types

4.2.6.2.1 Problem details

In case a policy request is not accepted, additional information can be provided in the response in addition to the normal HTTP status code.

The ProblemDetails statement contains the following attributes:

Table 4.2.6.2.1-1: Definition of statement type ProblemDetails

Attribute name	Data type	P	Cardinality	Description	Applicability
type	string	O	0..1	a URI reference according to IETF RFC 3986 [13] that identifies the problem type	
title	string	O	0..1	human-readable summary of the problem type	
status	number	O	0..1	the HTTP status code	
detail	string	O	0..1	human-readable explanation	
instance	string	O	0..1	URI reference that identifies the specific occurrence of the problem	

NOTE: attribute names are as defined in IETF RFC 7807 [14].

4.2.7 Error Handling

4.2.7.1 General

HTTP error handling shall be supported as specified in subclause 5.2.4 of 3GPP TS 29.500 [15] and according to the principles in 3GPP TS 29.501 [7].

4.2.7.2 Protocol Errors

No protocol errors are described in this version of the specification.

4.2.7.3 Application Errors

The application errors defined for the A1-P service are listed in Table 4.2.7.3-1.

Table 4.2.7.3-1: Application errors

Application Error	HTTP status code	Description
Bad Request	400	Object in payload not properly formulated or not related to the method
Not Found	404	No resource found at the URI
Method Not Allowed	405	Method not allowed for the URI
Conflict	409	Request could not be processed in the current state of the resource
Too many requests	429	Too many requests in a given amount of time
Service unavailable	503	Request cannot be handled (overloaded, maintenance).
Insufficient storage	507	Unable to store the representation.

4.3 A1-EI (enrichment information)

This section contains the definition of the REST based API for the Enrichment Information Service referred to as A1-EI.

4.3.1 Introduction

The A1-EI service shall use the A1-EI API.

The present version of the present specification defines API version 1 (v1) of the A1-EI API.

Based on the URI structure defined in clause 4.4.1 of 3GPP TS 29.501 [7], the request URI used in HTTP request from the A1-EI consumer towards the A1-EI producer shall have the following structure:

{apiRoot}/A1-EI/v1/<ResourceUriPart>

where the "ResourceUriPart" shall be as be defined in subclause 4.3.3.

4.3.2 Usage of HTTP

4.3.2.1 General

The A1 Transport, HTTP protocol and security requirements, is described in A1 interface: Transport Protocol [4].

4.3.2.2 HTTP standard headers

Note: the encodings and applicable MIME media type for the related Content-Type header are not specified in the current version.

4.3.2.3 HTTP custom headers

No HTTP custom headers are introduced in this version of the specification.

4.3.3 Resources

4.3.3.1 Overview

The resource URI structure for the A1-EI API is illustrated in figure 4.3.3.1-1.

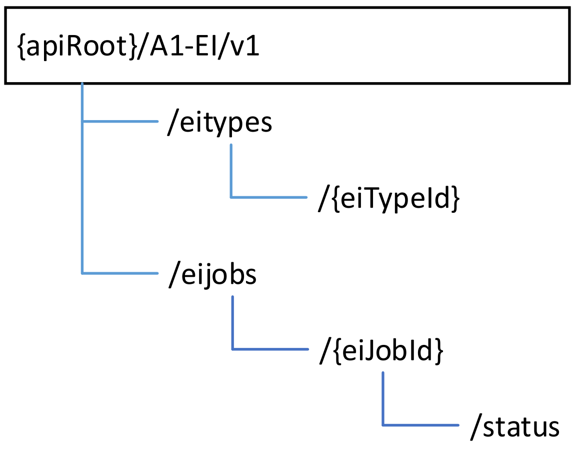

Figure 4.3.3.1-1: Resource URI structure of the A1-EI API

Table 4.3.3.1-1 provides an overview of the resources and applicable HTTP methods.

Table 4.3.3.1-1: Resources and methods overview

Resource name	Resource URI	HTTP method or custom operation	Description
All EI Type Identifiers	/eitypes	GET	Query all EI type identifiers
Individual EI Type	/ietypes/{eiTypeId}	GET	Query EI type
All EI Jobs	/eijobs	GET	Query all EI job identifiers
Individual EI Job	/eijobs/{eiJobId}	GET	Query EI job
		PUT	Create/Update EI job
		DELETE	Delete EI job
Individual EI Job Status	/eijobs/{eiJobId}/status	GET	Query EI job status
Notify EI Status	{jobStatusNotificationUri}	POST	Notify EI job status
Deliver EI	{jobResultUri}	POST	Deliver EI job result

4.3.3.1.1 EI type identifier

The EiTypeId is constructed based on two parts separated by "_" (underscore):

`typename_version`

where

`typename` is the unique label of the EI type;

`version` is the version of the EI type defined as major.minor.patch as described in SemVer [12].

The typename and version is assigned, and their uniqueness ensured, by the organizational entity that is responsible for the definition and maintenance of the EI type definition.

Note: the typename can be based on a prefix that indicates the organizational entity (e.g. ORAN or a company designator) and a text string that can be descriptive of the class, use case or variant of the EI type.

4.3.3.1.2 EI job identifier

An EiJobId is assigned by the Near-RT RIC and is unique within the domain of operation of the Non-RT RIC.

4.3.3.2 All EI Type Identifiers

4.3.3.2.1 Description

The resource represents EI type identifiers.

4.3.3.2.2 Resource Definition

The Resource URI and the supported resource variables are as defined in previous sections.

4.3.3.2.3 Resource Standard Methods

The following subclauses specifies the standard methods supported by the resource.

4.3.3.2.3.1 HTTP GET

This method shall support the request data structures specified in table 4.3.3.2.3.1-1 and the response data structures and response codes specified in table 4.3.3.2.3.1-2.

Table 4.3.3.2.3.1-1: Data structures supported by the HTTP GET Request Body on this resource

Data type	P	Cardinality	Description
N/A			There is no object in the message body of a GET request

Table 4.3.3.2.3.1-2: Data structures supported by the HTTP GET Response Body on this resource

Data type	P	Cardinality	Response codes	Description
Array(EiTypeId)	M	0..N	200 OK	All EI type identifiers
ProblemDetails	O	0..1	4xx/5xx	Detailed problem description

4.3.3.2.4 Resource Custom Operations

No custom operations are defined.

4.3.3.3 Individual EI Type

4.3.3.3.1 Description

The resource represents an EI type.

4.3.3.3.2 Resource Definition

The Resource URI and the supported resource variables are as defined in previous sections.

4.3.3.3.3 Resource Standard Methods

The following subclauses specifies the standard methods supported by the resource.

4.3.3.3.3.1 HTTP GET

This method shall support the request data structures specified in table 4.3.3.3.3.1-1 and the response data structures and response codes specified in table 4.3.3.3.3.1-2.

Table 4.3.3.3.3.1-1: Data structures supported by the HTTP GET Request Body on this resource

Data type	P	Cardinality	Description
N/A			There is no object in the message body of a GET request

Table 4.3.3.3.3.1-2: Data structures supported by the HTTP GET Response Body on this resource

Data type	P	Cardinality	Response codes	Description
EiTypeObject	M	1	200 OK	Requested EI type object
ProblemDetails	O	0..1	4xx/5xx	Detailed problem description

4.3.3.3.4 Resource Custom Operations

No custom operations are defined.

4.3.3.4 All EI Jobs

4.3.3.4.1 Description

The resource represents EI job identifiers.

4.3.3.4.2 Resource Definition

The Resource URI and the supported resource variables are as defined in previous sections.

4.3.3.4.3 Resource Standard Methods

The following subclauses specifies the standard methods supported by the resource.

Note: URI query parameters are not specified in the current version.

4.3.3.4.3.1 HTTP GET

This method shall support the request data structures specified in table 4.3.3.4.3.1-1 and the response data structures and response codes specified in table 4.3.3.4.3.1-2.

Table 4.3.3.4.3.1-1: Data structures supported by the HTTP GET Request Body on this resource

Data type	P	Cardinality	Description
N/A			There is no object in the message body of a GET request

Table 4.3.3.4.3.1-2: Data structures supported by the HTTP GET Response Body on this resource

Data type	P	Cardinality	Response codes	Description
array(EiJobId)	M	1	200 OK	All EI job identifiers
ProblemDetails	O	0..1	4xx/5xx	Detailed problem description

This method shall support the URI query parameters specified in table 4.3.3.4.3.1-3.

Table 4.3.3.4.3.1-3: URI query parameters supported by the HTTP GET method on this resource

Name	Data type	P	Cardinality	Description	Applicability
eiTypeId	string	"O"	0..1	eiTypeid for which EI Job identifiers are requested	Retrieve Ei Job identifiers for a certain EI Type

4.3.3.5 Individual EI Job

4.3.3.5.1 Description

The resource represents an EI job.

4.3.3.5.2 Resource Definition

The Resource URI and the supported resource variables are as defined in previous sections.

4.3.3.5.3 Resource Standard Methods

The following subclauses specifies the standard methods supported by the resource.

4.3.3.5.3.1 HTTP PUT

This method shall support the request data structures specified in table 4.3.3.5.3.1-1 and the response data structures and response codes specified in table 4.3.3.5.3.1-2.

Table 4.3.3.5.3.1-1: Data structures supported by the HTTP PUT Request Body on this resource

Data type	P	Cardinality	Description
EiJobObject	M	1	Create or Update EI job

Table 4.3.3.5.3.1-2: Data structures supported by the HTTP PUT Response Body on this resource

Data type	P	Cardinality	Response codes	Description
EiJobObject	M	1	201 Created 200 OK	Confirmation of created or updated EI job
ProblemDetails	O	0..1	4xx/5xx	Detailed problem description

4.3.3.5.3.2 HTTP GET

This method shall support the request data structures specified in table 4.3.3.5.3.2-1 and the response data structures and response codes specified in table 4.3.3.5.3.2-2.

Table 4.3.3.5.3.2-1: Data structures supported by the HTTP GET Request Body on this resource

Data type	P	Cardinality	Description
N/A			There is no object in the message body of a GET request

Table 4.3.3.5.3.2-2: Data structures supported by the HTTP GET Response Body on this resource

Data type	P	Cardinality	Response codes	Description
EiJobObject	M	1	200 OK	Requested EI job object
ProblemDetails	O	0..1	4xx/5xx	Detailed problem description

4.3.3.5.3.3 HTTP DELETE

This method shall support the request data structures specified in table 4.3.3.5.3.3-1 and the response data structures and response codes specified in table 4.3.3.5.3.3-2.

Table 4.3.3.5.3.3-1: Data structures supported by the HTTP DELETE Request Body on this resource

Data type	P	Cardinality	Description
N/A			There is no object in the message body of a DELETE request

Table 4.3.3.5.3.3-2: Data structures supported by the HTTP DELETE Response Body on this resource

Data type	P	Cardinality	Response codes	Description
N/A			204 No content	Confirmation of successful deletion
ProblemDetails	O	0..1	4xx/5xx	Detailed problem description

4.3.3.5.4 Resource Custom Operations

No custom operations are defined.

4.3.3.6 Individual EI Job Status

4.3.3.6.1 Description

The resource represents the status of an EI job.

4.3.3.6.2 Resource Definition

The Resource URI and the supported resource variables are as defined in previous sections.

4.3.3.6.3 Resource Standard Methods

The following subclauses specifies the standard methods supported by the resource.

Note: URI query parameters are not specified in the current version.

4.3.3.6.3.1 HTTP GET

This method shall support the request data structures specified in table 4.3.3.6.3.1-1 and the response data structures and response codes specified in table 4.3.3.6.3.1-2.

Table 4.3.3.6.3.1-1: Data structures supported by the HTTP GET Request Body on this resource

Data type	P	Cardinality	Description
N/A			There is no object in the message body of a GET request

Table 4.3.3.6.3.1-2: Data structures supported by the HTTP GET Response Body on this resource

Data type	P	Cardinality	Response codes	Description
EiJobStatusObject	M	1	200 OK	Requested EI job status object
ProblemDetails	O	0..1	4xx/5xx	Detailed problem description

4.2.3.3.4 Resource Custom Operations

No custom operations are defined.

4.3.4 Custom Operations without associated resources

No custom operations are defined.

4.3.5 Notifications

4.3.5.1 Notify EI Job Status

4.3.5.1.1 Description

The resource represents the destination for EI job status notifications.

4.3.5.1.2 Resource Definition

The Resource URI is a callback URI provided as a query parameter in URL when creating an EI job.

4.3.5.1.3 Resource Standard Methods

The following subclauses specifies the standard methods supported by the resource.

Note: URI query parameters are not specified in the current version.

4.3.5.1.3.1 HTTP POST

This method shall support the request data structures specified in table 4.3.5.1.3.1-1 and the response data structures and response codes specified in table 4.3.5.1.3.1-2.

Table 4.3.5.1.3.1-1: Data structures supported by the HTTP POST Request Body on this resource

Data type	P	Cardinality	Description
EiJobStatusObject	M	1	Notify EI job status

Table 4.3.5.1.3.1-2: Data structures supported by the HTTP POST Response Body on this resource

Data type	P	Cardinality	Response codes	Description
N/A			204 No content	Confirmation of received notification

4.3.6 Delivery

4.3.6.1 Deliver EI

4.3.6.1.1 Description

The resource represents the destination for EI delivery in the case of push-based delivery.

4.3.6.1.2 Resource Definition

The Resource URI is a target URI provided in the EI job object during EI job creation.

4.3.6.1.3 Resource Standard Methods

The following subclauses specifies the standard methods supported by the resource.

4.3.6.1.3.1 HTTP POST

This method shall support the request data structures specified in table 4.3.6.1.3.1-1 and the response data structures and response codes specified in table 4.3.6.1.3.1-2.

Table 4.3.6.1.3.1-1: Data structures supported by the HTTP POST Request Body on this resource

Data type	P	Cardinality	Description
EiJobResultObject	M	1	Carry EI payload, i.e. the result from an EI job

Table 4.3.6.1.3.1-2: Data structures supported by the HTTP POST Response Body on this resource

Data type	P	Cardinality	Response codes	Description
N/A			204 No content	Confirmation of received notification
ProblemDetails	O	0..1	4xx/5xx	Detailed problem description

4.3.7 Data model

This subclause specifies the application protocol data model supported by the A1-EI API.

The data model for the data types transported in the A1-EI procedures is defined in A1 Interface: Type Definitions [5].

4.3.7.1 Simple data types and enumerations

4.3.7.1.1 Simple data types

The URIs for policy operations contains an eiTypeId attribute and an eiJobId attribute.

Table 4.3.7.1.1-1: General definition of simple data types for URI identifiers

Type Name	Type Definition	Description	Applicability
EiTypeId	string	EI type identifier assigned by the owner of an EI type definition	used in URI
EiJobId	string	EI job identifier assigned by the A1-EI Consumer when an EI job is created	used in URI

Table 4.3.7.1.1-2: General definition of simple data types for callback URIs

Type Name	Type Definition	Description	Applicability
jobStatusNotificationUri	string	target URI for EI job status notifcations	provided in EI Job object and used in job status notification procedure
jobResultUri	string	target URI for EI job results	provided in EI Job object and used in job result deliver procedure

4.3.7.2 Structured data types

4.3.7.2.1 Problem details

The problem details statement is the same as defined for A1-P, see chapter 4.2.6.2.1.

4.3.8 Error handling

4.3.8.1 General

HTTP error handling shall be supported as specified in subclause 5.2.4 of 3GPP TS 29.500 [15] and according to the principles in 3GPP TS 29.501 [7].

4.3.8.2 Protocol Errors

No protocol errors are described in this version of the specification.

4.3.8.3 Application Errors

The application errors defined for the A1-EI service are listed in Table 4.3.8.3-1.

Table 4.3.8.3-1: Application errors

Application Error	HTTP status code	Description
Bad Request	400	Object in payload not properly formulated or not related to the method
Not Found	404	No resource found at the URI
Method Not Allowed	405	Method not allowed for the URI
Conflict	409	Request could not be processed in the current state of the resource
Too many requests	429	Too many requests in a given amount of time
Service unavailable	503	Request cannot be handled (overloaded, maintenance).
Insufficient storage	507	Unable to store the representation.

Annex A (normative): OpenAPI specification

A.1 General

This Annex specifies the formal definition of the A1 API(s). It consists of OpenAPI documents in YAML format that are based on the OpenAPI 3.0.0 Specification [16].

Informative copies of the OpenAPI documents contained in this O-RAN Technical Specification may be available at a later stage.

A.1.1 Versioning of A1 OpenAPI documents

The OpenAPI documents for the A1 services found in this chapter are versioned by Semantic Versioning 2.0.0 [12] as described in the OpenAPI Specification [16]. When included in the present specification, the OpenAPI documents are considered as released and are versioned using three digit major.minor.patch where the main compatibility expectations stated for Sematic Versioning [12] implies:

major version is stepped up when incompatible API changes are made to the OpenAPI document. This corresponds to saying that implementations of an A1 service in Non/Near-RT RICs are incompatible in case the API version is different. The major version in the OpenAPI document corresponds to the API version in the URI for of the A1 service defined in chapter 4.

minor version is stepped up when features are added to the OpenAPI document in way that keeps implementations compatible although all features are not supported by both the service producer and the service consumer of the A1 service.

patch version is stepped up when errors are corrected in a backward compatible way, when or editorial changes are made to the OpenAPI document, but no features are added.

Note: Non/Near-RT RIC products that implement various API versions of an A1 service can be compatible by supporting several API versions of an A1 service. The present specification specifies only one API version, and contains only one OpenAPI document, for each A1 service.

A.1.2 Current API versions

The present version of present specification defines the API versions indicated in table A.1-1.

Table A.1-1

API name	API version	Open API version
A1-P	v2	2.0.1
A1-EI	v1	1.0.1

Note: API names and API versions are defined in clause 4 and Open API versions are defined by the Open API documents in the present clause.

A.2 Policy Management API

```
openapi: 3.0.1
info:
  title: 'A1-P Policy Management Service'
  version: 2.0.1
  description: |
    API for Policy Management Service.
    © 2021, O-RAN Alliance.
    All rights reserved.
externalDocs:
  description: 'O-RAN.WG2.A1AP-v03.01 A1 interface: Application Protocol'
  url: 'https://www.o-ran.org/specifications'
servers:
  - url: '{apiRoot}/A1-P/v2'
    variables:
      apiRoot:
        default: 'https://example.com'
        description: 'apiRoot as defined in clause 4.2.1 in ORAN-WG2.A1.AP'
paths:
  '/policytypes':
    get:
      description: 'Get all policy type identifiers'
      tags:
      - All Policy Type Identifiers
      responses:
        200:
          description: 'Array of all policy type identifiers'
          content:
            application/json:
              schema:
                type: array
                items:
                  "$ref": "#/components/schemas/PolicyTypeId"
                minItems: 0
        429:
          "$ref": "#/components/responses/429-TooManyRequests"
        503:
          "$ref": "#/components/responses/503-ServiceUnavailable"

  '/policytypes/{policyTypeId}':
    parameters:
      - name: policyTypeId
        in: path
        required: true
        schema:
          "$ref": "#/components/schemas/PolicyTypeId"
    get:
      description: 'Get the schemas for a policy type'
      tags:
      - Individual Policy Type
      responses:
        200:
          description: 'The policy type schemas'
          content:
            application/json:
              schema:
                "$ref": "#/components/schemas/PolicyTypeObject"
        404:
          "$ref": "#/components/responses/404-NotFound"
        429:
          "$ref": "#/components/responses/429-TooManyRequests"
        503:
          "$ref": "#/components/responses/503-ServiceUnavailable"

  '/policytypes/{policyTypeId}/policies':
    get:
      description: 'Get all policy identifiers'
      tags:
      - All Policy Identifiers
      parameters:
        - name: policyTypeId
          in: path
          required: true
          schema:
            "$ref": "#/components/schemas/PolicyTypeId"
      responses:
        200:
          description: 'Array of all policy identifiers'
          content:
            application/json:
              schema:
                type: array
                items:
                  "$ref": "#/components/schemas/PolicyId"
                minItems: 0
        429:
          "$ref": "#/components/responses/429-TooManyRequests"
        503:
          "$ref": "#/components/responses/503-ServiceUnavailable"

  '/policytypes/{policyTypeId}/policies/{policyId}':
    parameters:
      - name: policyTypeId
        in: path
        required: true
        schema:
          "$ref": "#/components/schemas/PolicyTypeId"
      - name: policyId
        in: path
        required: true
        schema:
          "$ref": "#/components/schemas/PolicyId"
    put:
      description: 'Create, or update, a policy'
      tags:
      - Individual Policy Object
      parameters:
        - name: notificationDestination
          in: query
          required: false
          schema:
            "$ref": "#/components/schemas/NotificationDestination"
      requestBody:
        required: true
        content:
          application/json:
            schema:
              "$ref": "#/components/schemas/PolicyObject"
      responses:
        200:
          description: 'The policy was updated'
          content:
            application/json:
              schema:
                "$ref": "#/components/schemas/PolicyObject"
        201:
          description: 'The policy was created'
          content:
            application/json:
              schema:
                "$ref": "#/components/schemas/PolicyObject"
          headers:
            Location:
              description: 'Contains the URI of the created policy'
              required: true
              schema:
                type: string
        400:
          "$ref": "#/components/responses/400-BadRequest"
        409:
          "$ref": "#/components/responses/409-Conflict"
        429:
          "$ref": "#/components/responses/429-TooManyRequests"
        503:
          "$ref": "#/components/responses/503-ServiceUnavailable"
        507:
          "$ref": "#/components/responses/507-InsufficientStorage"
      callbacks:
        policyStatusNotification:
          '{$request.query.notificationDestination}':
            post:
              description: 'Notify about status changes for this policy'
              requestBody:
                required: true
                content:
                  application/json:
                    schema:
                      "$ref": "#/components/schemas/PolicyStatusObject"
              responses:
                204:
                  description: 'Notification received'
    get:
      description: 'Query a policy'
      tags:
      - Individual Policy Object
      responses:
        200:
          description: 'The requested policy'
          content:
            application/json:
              schema:
                "$ref": "#/components/schemas/PolicyObject"
        404:
          "$ref": "#/components/responses/404-NotFound"
        409:
          "$ref": "#/components/responses/409-Conflict"
        429:
          "$ref": "#/components/responses/429-TooManyRequests"
        503:
          "$ref": "#/components/responses/503-ServiceUnavailable"
    delete:
      description: 'Delete a policy'
      tags:
      - Individual Policy Object
      responses:
        204:
          description: 'The policy was deleted'
        404:
          "$ref": "#/components/responses/404-NotFound"
        429:
          "$ref": "#/components/responses/429-TooManyRequests"
        503:
          "$ref": "#/components/responses/503-ServiceUnavailable"

  '/policytypes/{policyTypeId}/policies/{policyId}/status':
    parameters:
      - name: policyTypeId
        in: path
        required: true
        schema:
          "$ref": "#/components/schemas/PolicyTypeId"
      - name: policyId
        in: path
        required: true
        schema:
          "$ref": "#/components/schemas/PolicyId"
    get:
      description: 'Query a policy status'
      tags:
      - Individual Policy Status Object
      responses:
        200:
          description: 'The requested policy status'
          content:
            application/json:
              schema:
                "$ref": "#/components/schemas/PolicyStatusObject"
        404:
          "$ref": "#/components/responses/404-NotFound"
        409:
          "$ref": "#/components/responses/409-Conflict"
        429:
          "$ref": "#/components/responses/429-TooManyRequests"
        503:
          "$ref": "#/components/responses/503-ServiceUnavailable"

components:
  schemas:
    #
    # Representation objects
    #
    PolicyObject:
      description: 'A generic policy object that can be used to transport any policy. Additionally,
a policy shall be valid according to the schema of its specific policy type.'
      type: object

    PolicyStatusObject:
      description: 'A generic policy status object that can be used to transport any policy status.
Additionally, a policy status shall be valid according to the schema of its specific policy type.'
      type: object

    PolicyTypeObject:
      description: 'A definition of a policy type, i.e. the schemas for a policy respectively its
status'
      type: object
      properties:
        policySchema:
          "$ref": "#/components/schemas/JsonSchema"
        statusSchema:
          "$ref": "#/components/schemas/JsonSchema"
      required:
        - policySchema

    ProblemDetails:
      description: 'A problem detail to carry details in a HTTP response according to RFC 7807'
      type: object
      properties:
        type:
          type: string
        title:
          type: string
        status:
          type: number
        detail:
          type: string
        instance:
          type: string

    #
    # Simple data types
    #
    JsonSchema:
      description: 'A JSON schema following http://json-schema.org/draft-07/schema'
      type: object

    NotificationDestination:
      description: 'A complete callback URI defined according to IETF RFC 3986 where to send
notifications'
      type: string

    PolicyId:
      description: 'Policy identifier assigned by the A1-P Consumer when a policy is created'
      type: string

    PolicyTypeId:
      description: 'Policy type identifier assigned by the A1-P Provider'
      type: string

  responses:
    400-BadRequest:
      description: 'Object in payload not properly formulated or not related to the method'
      content:
        application/problem+json:
          schema:
            "$ref": "#/components/schemas/ProblemDetails"

    404-NotFound:
      description: 'No resource found at the URI'
      content:
        application/problem+json:
          schema:
            "$ref": "#/components/schemas/ProblemDetails"

    405-MethodNotAllowed:
      description: 'Method not allowed for the URI'
      content:
        application/problem+json:
          schema:
            "$ref": "#/components/schemas/ProblemDetails"

      409-Conflict:
      description: 'Request could not be processed in the current state of the resource'
      content:
        application/problem+json:
          schema:
            "$ref": "#/components/schemas/ProblemDetails"

      429-TooManyRequests:
      description: 'Too many requests have been sent in a given amount of time'
      content:
        application/problem+json:
          schema:
            "$ref": "#/components/schemas/ProblemDetails"

      503-ServiceUnavailable:
      description: 'The provider is currently unable to handle the request due to a temporary
overload'
      content:
        application/problem+json:
          schema:
            "$ref": "#/components/schemas/ProblemDetails"

      507-InsufficientStorage:
      description: 'The method could not be performed on the resource because the provider is unable
to store the representation needed to successfully complete the request'
      content:
        application/problem+json:
          schema:
            "$ref": "#/components/schemas/ProblemDetails"
```

A.3 Enrichment Information API

```
openapi: 3.0.1
info:
  title: A1-EI Enrichment Information Service
  description: |
    API for Enrichment Information Service.
    © 2021, O-RAN Alliance.
    All rights reserved.
  version: 1.0.1
externalDocs:
  description: 'O-RAN.WG2.A1AP-v03.01 A1 interface: Application Protocol'
  url: https://www.o-ran.org/specifications
servers:
- url: //localhost:36353/
tags:
- name: A1-EI (enrichment information)
  description: ""
paths:
  /A1-EI/v1/eijobs/{eiJobId}:
    get:
      tags:
      - A1-EI (enrichment information)
      summary: Individual EI job
      operationId: getIndividualEiJobUsingGET
      parameters:
      - name: eiJobId
        in: path
        description: eiJobId
        required: true
        schema:
          type: string
      responses:
        200:
          description: EI job
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/EiJobObject'
        401:
          description: Unauthorized
          content: {}
        403:
          description: Forbidden
          content: {}
        404:
          description: Enrichment Information job is not found
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ProblemDetails'
      deprecated: false
    put:
      tags:
      - A1-EI (enrichment information)
      summary: Individual EI job
      operationId: putIndividualEiJobUsingPUT
      parameters:
      - name: eiJobId
        in: path
        description: eiJobId
        required: true
        schema:
          type: string
      requestBody:
        description: eiJobObject
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/EiJobObject'
        required: true
      responses:
        200:
          description: Job updated
          content: {}
        201:
          description: Job created
          content: {}
        401:
          description: Unauthorized
          content: {}
        403:
          description: Forbidden
          content: {}
        404:
          description: Enrichment Information type is not found
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ProblemDetails'
      deprecated: false
      x-codegen-request-body-name: eiJobObject
      callbacks:
       jobStatusNotification:
          '{$request.body.jobStatusNotificationUri}':
            post:
              description: 'Notify about status changes for this EI job'
              requestBody:
                required: true
                content:
                  application/json:
                    schema:
                      "$ref": "#/components/schemas/EiJobStatusObject"
              responses:
                204:
                  description: 'Notification received'
          jobResult:
          '{$request.body.jobResultUri}':
            post:
              description: 'Deliverance of EI'
              requestBody:
                required: true
                content:
                  application/json:
                    schema:
                      "$ref": "#/components/schemas/EiResultObject"
              responses:
                204:
                  description: 'Information received'
    delete:
      tags:
      - A1-EI (enrichment information)
      summary: Individual EI job
      operationId: deleteIndividualEiJobUsingDELETE
      parameters:
      - name: eiJobId
        in: path
        description: eiJobId
        required: true
        schema:
          type: string
      responses:
        200:
          description: Not used
          content: {}
        204:
          description: Job deleted
          content: {}
        401:
          description: Unauthorized
          content: {}
        403:
          description: Forbidden
          content: {}
        404:
          description: Enrichment Information job is not found
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ProblemDetails'
      deprecated: false
  /A1-EI/v1/eitypes/{eiTypeId}:
    get:
      tags:
      - A1-EI (enrichment information)
      summary: Individual EI type
      operationId: getEiTypeUsingGET
      parameters:
      - name: eiTypeId
        in: path
        description: eiTypeId
        required: true
        schema:
          type: string
      responses:
        200:
          description: EI type
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/EiTypeObject'
        401:
          description: Unauthorized
          content: {}
        403:
          description: Forbidden
          content: {}
        404:
          description: Enrichment Information type is not found
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ProblemDetails'
      deprecated: false
  /A1-EI/v1/eijobs:
    get:
      tags:
      - A1-EI (enrichment information)
      summary: EI job identifiers
      description: query for EI job identifiers
      operationId: getEiJobIdsUsingGET
      parameters:
      - name: eiTypeId
        in: query
        description: selects EI jobs of matching EI type
        allowEmptyValue: false
        schema:
          type: string
      - name: owner
        in: query
        description: selects EI jobs for one EI job owner
        allowEmptyValue: false
        schema:
          type: string
      responses:
        200:
          description: EI job identifiers
          content:
            application/json:
              schema:
                type: array
                items:
                  type: string
        401:
          description: Unauthorized
          content: {}
        403:
          description: Forbidden
          content: {}
        404:
          description: Enrichment Information type is not found
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ProblemDetails'
      deprecated: false
  /A1-EI/v1/eijobs/{eiJobId}/status:
    get:
      tags:
      - A1-EI (enrichment information)
      summary: EI job status
      operationId: getEiJobStatusUsingGET
      parameters:
      - name: eiJobId
        in: path
        description: eiJobId
        required: true
        schema:
          type: string
      responses:
        200:
          description: EI job status
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/EiJobStatusObject'
        401:
          description: Unauthorized
          content: {}
        403:
          description: Forbidden
          content: {}
        404:
          description: Enrichment Information job is not found
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ProblemDetails'
      deprecated: false
  /A1-EI/v1/eitypes:
    get:
      tags:
      - A1-EI (enrichment information)
      summary: EI type identifiers
      operationId: getEiTypeIdentifiersUsingGET
      responses:
        200:
          description: EI type identifiers
          content:
            application/json:
              schema:
                type: array
                items:
                  type: string
        401:
          description: Unauthorized
          content: {}
        403:
          description: Forbidden
          content: {}
        404:
          description: Not Found
          content: {}
      deprecated: false
components:
  schemas:
    EiTypeObject:
      title: EiTypeObject
      type: object
      description: Information for an EI type
    ProblemDetails:
      title: ProblemDetails
      type: object
      properties:
        detail:
          type: string
          description: A human-readable explanation specific to this occurrence of
            the problem.
          example: EI job type not found
        status:
          type: integer
          description: The HTTP status code generated by the origin server for this
            occurrence of the problem.
          format: int32
          example: 404
        type:
          type: string
        title:
          type: string
        instance:
          type: string
      description: A problem detail to carry details in a HTTP response according
        to RFC 7807
    EiJobStatusObject:
      title: EiJobStatusObject
      required:
      - eiJobStatus
      type: object
      properties:
        eiJobStatus:
          type: string
          description: |-
            values:
            ENABLED: the A1-EI producer is able to deliver EI result for the EI job
            DISABLED: the A1-EI producer is unable to deliver EI result for the EI job
          enum:
          - ENABLED
          - DISABLED
      description: Status for an EI job
    EiJobObject:
      title: EiJobObject
      required:
      - eiTypeId
      - jobDefinition
      - jobResultUri
      type: object
      properties:
        eiTypeId:
          type: string
          description: EI type Idenitifier of the EI job
        jobResultUri:
          type: string
          description: The target URI of the EI data
        jobStatusNotificationUri:
          type: string
          description: The target of EI job status notifications
        jobDefinition:
          type: object
          properties: {}
          description: EI type specific job data
      description: Information for an Enrichment Information Job
    EiResultObject:
     title: EiResultObject
```

Annex ZZZ : O-RAN Adopter License Agreement

BY DOWNLOADING, USING OR OTHERWISE ACCESSING ANY O-RAN SPECIFICATION, ADOPTER AGREES TO THE TERMS OF THIS AGREEMENT.

This O-RAN Adopter License Agreement (the “Agreement”) is made by and between the O-RAN Alliance and the entity that downloads, uses or otherwise accesses any O-RAN Specification, including its Affiliates (the “Adopter”).

This is a license agreement for entities who wish to adopt any O-RAN Specification.

Section 1: DEFINITIONS

1.1 “Affiliate” means an entity that directly or indirectly controls, is controlled by, or is under common control with another entity, so long as such control exists. For the purpose of this Section, “Control” means beneficial ownership of fifty (50%) percent or more of the voting stock or equity in an entity.

1.2 “Compliant Implementation” means any system, device, method or operation (whether implemented in hardware, software or combinations thereof) that fully conforms to a Final Specification.

1.3 “Adopter(s)” means all entities, who are not Members, Contributors or Academic Contributors, including their Affiliates, who wish to download, use or otherwise access O-RAN Specifications.

1.4 “Minor Update” means an update or revision to an O-RAN Specification published by O-RAN Alliance that does not add any significant new features or functionality and remains interoperable with the prior version of an O-RAN Specification. The term “O-RAN Specifications” includes Minor Updates.

1.5 “Necessary Claims” means those claims of all present and future patents and patent applications, other than design patents and design registrations, throughout the world, which (i) are owned or otherwise licensable by a Member, Contributor or Academic Contributor during the term of its Member, Contributor or Academic Contributorship; (ii) such Member, Contributor or Academic Contributor has the right to grant a license without the payment of consideration to a third party; and (iii) are necessarily infringed by a Compliant Implementation (without considering any Contributions not included in the Final Specification). A claim is necessarily infringed only when it is not possible on technical (but not commercial) grounds, taking into account normal technical practice and the state of the art generally available at the date any Final Specification was published by the O-RAN Alliance or the date the patent claim first came into existence, whichever last occurred, to make, sell, lease, otherwise dispose of, repair, use or operate a Compliant Implementation without infringing that claim. For the avoidance of doubt in exceptional cases where a Final Specification can only be implemented by technical solutions, all of which infringe patent claims, all such patent claims shall be considered Necessary Claims.

1.6 “Defensive Suspension” means for the purposes of any license grant pursuant to Section 3, Member, Contributor, Academic Contributor, Adopter, or any of their Affiliates, may have the discretion to include in their license a term allowing the licensor to suspend the license against a licensee who brings a patent infringement suit against the licensing Member, Contributor, Academic Contributor, Adopter, or any of their Affiliates.

Section 2: COPYRIGHT LICENSE

2.1 Subject to the terms and conditions of this Agreement, O-RAN Alliance hereby grants to Adopter a nonexclusive, nontransferable, irrevocable, non-sublicensable, worldwide copyright license to obtain, use and modify O-RAN Specifications, but not to further distribute such O-RAN Specification in any modified or unmodified way, solely in furtherance of implementations of an ORAN

Specification.

2.2 Adopter shall not use O-RAN Specifications except as expressly set forth in this Agreement or in a separate written agreement with O-RAN Alliance.

Section 3: FRAND LICENSE

3.1 Members, Contributors and Academic Contributors and their Affiliates are prepared to grant based on a separate Patent License Agreement to each Adopter under Fair Reasonable And Non- Discriminatory (FRAND) terms and conditions with or without compensation (royalties) a nonexclusive, non-transferable, irrevocable (but subject to Defensive Suspension), non-sublicensable, worldwide patent license under their Necessary Claims to make, have made, use, import, offer to sell, lease, sell and otherwise distribute Compliant Implementations; provided, however, that such license shall not extend: (a) to any part or function of a product in which a Compliant Implementation is incorporated that is not itself part of the Compliant Implementation; or (b) to any Adopter if that Adopter is not making a reciprocal grant to Members, Contributors and Academic Contributors, as set forth in Section 3.3. For the avoidance of doubt, the foregoing licensing commitment includes the distribution by the Adopter's distributors and the use by the Adopter's customers of such licensed Compliant Implementations.

3.2 Notwithstanding the above, if any Member, Contributor or Academic Contributor, Adopter or their Affiliates has reserved the right to charge a FRAND royalty or other fee for its license of Necessary Claims to Adopter, then Adopter is entitled to charge a FRAND royalty or other fee to such Member, Contributor or Academic Contributor, Adopter and its Affiliates for its license of Necessary Claims to its licensees.

3.3 Adopter, on behalf of itself and its Affiliates, shall be prepared to grant based on a separate Patent License Agreement to each Members, Contributors, Academic Contributors, Adopters and their Affiliates under Fair Reasonable And Non-Discriminatory (FRAND) terms and conditions with or without compensation (royalties) a nonexclusive, non-transferable, irrevocable (but subject to Defensive Suspension), non-sublicensable, worldwide patent license under their Necessary Claims to make, have made, use, import, offer to sell, lease, sell and otherwise distribute Compliant Implementations; provided, however, that such license will not extend: (a) to any part or function of a product in which a Compliant Implementation is incorporated that is not itself part of the Compliant Implementation; or (b) to any Members, Contributors, Academic Contributors, Adopters and their Affiliates that is not making a reciprocal grant to Adopter, as set forth in Section 3.1. For the avoidance of doubt, the foregoing licensing commitment includes the distribution by the Members', Contributors', Academic Contributors', Adopters' and their Affiliates' distributors and the use by the Members', Contributors', Academic Contributors', Adopters' and their Affiliates' customers of such licensed Compliant Implementations.

Section 4: TERM AND TERMINATION

4.1 This Agreement shall remain in force, unless early terminated according to this Section 4.

4.2 O-RAN Alliance on behalf of its Members, Contributors and Academic Contributors may terminate this Agreement if Adopter materially breaches this Agreement and does not cure or is not capable of curing such breach within thirty (30) days after being given notice specifying the breach.

4.3 Sections 1, 3, 5 - 11 of this Agreement shall survive any termination of this Agreement. Under surviving Section 3, after termination of this Agreement, Adopter will continue to grant licenses (a) to entities who become Adopters after the date of termination; and (b) for future versions of ORAN Specifications that are backwards compatible with the version that was current as of the date of termination.

Section 5: CONFIDENTIALITY

Adopter will use the same care and discretion to avoid disclosure, publication, and dissemination of O-RAN Specifications to third parties, as Adopter employs with its own confidential information, but no less than reasonable care. Any disclosure by Adopter to its Affiliates, contractors and consultants should be subject to an obligation of confidentiality at least as restrictive as those contained in this Section. The foregoing obligation shall not apply to any information which is: (1) rightfully known by Adopter without any limitation on use or disclosure prior to disclosure; (2) publicly available through no fault of Adopter; (3) rightfully received without a duty of confidentiality; (4) disclosed by O-RAN Alliance or a Member, Contributor or Academic Contributor to a third party without a duty of confidentiality on such third party; (5) independently developed by Adopter; (6) disclosed pursuant to the order of a court or other authorized governmental body, or as required by law, provided that Adopter provides reasonable prior written notice to O-RAN Alliance, and cooperates with O-RAN Alliance and/or the applicable Member, Contributor or Academic Contributor to have the opportunity to oppose any such order; or (7) disclosed by Adopter with O-RAN Alliance's prior written approval.

Section 6: INDEMNIFICATION

Adopter shall indemnify, defend, and hold harmless the O-RAN Alliance, its Members, Contributors or Academic Contributors, and their employees, and agents and their respective successors, heirs and assigns (the "Indemnitees"), against any liability, damage, loss, or expense (including reasonable attorneys' fees and expenses) incurred by or imposed upon any of the Indemnitees in connection with any claims, suits, investigations, actions, demands or judgments arising out of Adopter's use of the licensed O-RAN Specifications or Adopter's commercialization of products that comply with O-RAN Specifications.

Section 7: LIMITATIONS ON LIABILITY; NO WARRANTY

EXCEPT FOR BREACH OF CONFIDENTIALITY, ADOPTER'S BREACH OF SECTION 3, AND ADOPTER'S INDEMNIFICATION OBLIGATIONS, IN NO EVENT SHALL ANY PARTY BE LIABLE TO ANY OTHER PARTY OR THIRD PARTY FOR ANY INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES RESULTING FROM ITS PERFORMANCE OR NON-PERFORMANCE UNDER THIS AGREEMENT, IN EACH CASE WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, AND WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES. O-RAN SPECIFICATIONS ARE PROVIDED "AS IS" WITH NO WARRANTIES OR CONDITIONS WHATSOEVER, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. THE O-RAN ALLIANCE AND THE MEMBERS, CONTRIBUTORS OR ACADEMIC CONTRIBUTORS EXPRESSLY DISCLAIM ANY WARRANTY OR CONDITION OF MERCHANTABILITY, SECURITY, SATISFACTORY QUALITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, ERROR-FREE OPERATION, OR ANY WARRANTY OR CONDITION FOR O-RAN SPECIFICATIONS.

Section 8: ASSIGNMENT

Adopter may not assign the Agreement or any of its rights or obligations under this Agreement or make any grants or other sublicenses to this Agreement, except as expressly authorized hereunder, without having first received the prior, written consent of the O-RAN Alliance, which consent may be withheld in O-RAN Alliance's sole discretion. O-RAN Alliance may freely assign this Agreement.

Section 9: THIRD-PARTY BENEFICIARY RIGHTS

Adopter acknowledges and agrees that Members, Contributors and Academic Contributors (including future Members, Contributors and Academic Contributors) are entitled to rights as a third-party beneficiary under this Agreement, including as licensees under Section 3.

Section 10: BINDING ON AFFILIATES

Execution of this Agreement by Adopter in its capacity as a legal entity or association constitutes that legal entity's or association's agreement that its Affiliates are likewise bound to the obligations that are applicable to Adopter hereunder and are also entitled to the benefits of the rights of Adopter hereunder.

Section 11: GENERAL

This Agreement is governed by the laws of Germany without regard to its conflict or choice of law provisions.

This Agreement constitutes the entire agreement between the parties as to its express subject matter and expressly supersedes and replaces any prior or contemporaneous agreements between the parties, whether written or oral, relating to the subject matter of this Agreement.

Adopter, on behalf of itself and its Affiliates, agrees to comply at all times with all applicable laws, rules and regulations with respect to its and its Affiliates' performance under this Agreement, including without limitation, export control and antitrust laws. Without limiting the generality of the foregoing, Adopter acknowledges that this Agreement prohibits any communication that would violate the antitrust laws.

By execution hereof, no form of any partnership, joint venture or other special relationship is created between Adopter, or O-RAN Alliance or its Members, Contributors or Academic Contributors. Except as expressly set forth in this Agreement, no party is authorized to make any commitment on behalf of Adopter, or O-RAN Alliance or its Members, Contributors or Academic Contributors.

In the event that any provision of this Agreement conflicts with governing law or if any provision is held to be null, void or otherwise ineffective or invalid by a court of competent jurisdiction, (i) such provisions will be deemed stricken from the contract, and (ii) the remaining terms, provisions, covenants and restrictions of this Agreement will remain in full force and effect.

Any failure by a party or third party beneficiary to insist upon or enforce performance by another party of any of the provisions of this Agreement or to exercise any rights or remedies under this Agreement or otherwise by law shall not be construed as a waiver or relinquishment to any extent of the other parties' or third party beneficiary's right to assert or rely upon any such provision, right or remedy in that or any other instance; rather the same shall be and remain in full force and effect.