O-RAN.WG2.A1AP-v03.01

*Technical Specification*

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

**A1 interface: Application Protocol**

Copyright © 2021 by O-RAN ALLIANCE e.V.

By using, accessing or downloading any part of this O-RAN specification document, including by copying, saving, distributing, displaying or preparing derivatives of, you agree to be and are bound to the terms of the O-RAN Adopter License Agreement contained in the Annex ZZZ of this specification. All other rights reserved.

O-RAN ALLIANCE e.V.

Buschkauler Weg 27, 53347 Alfter, Germany Register of Associations, Bonn VR 11238 VAT ID DE321720189

# 1 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 |

2

3

# 1 Contents

##### Revision History 2

##### Chapter 1 Introductory Material 5

1. 1.1 Scope 5
2. 1.1.1 Compatibility of A1 versions 5
3. 1.2 References 5
4. 1.3 Definitions and Abbreviations 6
5. 1.3.1 Definitions 6
6. 1.3.2 Abbreviations 7

##### Chapter 2 A1 Application Protocol 7

##### Chapter 3 A1 Services 8

1. 3.1 Introduction 8
2. 3.2 Policy Management Service 8
3. 3.2.1 Service Description 8
4. 3.2.2 Service Operations 10
5. 3.3 Enrichment Information Service 16
6. 3.3.1 Service Description 16
7. 3.3.2 EI Discovery Service Operations 18
8. 3.3.3 EI Job Control Service Operations 20
9. 3.3.4 EI Delivery Service Operations 24
10. 3.4 ML Model Management Service 25

##### Chapter 4 API Definitions 25

1. 4.1 Introduction 25
2. 4.1.2 Compatibility of API versions for A1 services 25
3. 4.2 A1-P (policy management) 26
4. 4.2.1 Introduction 26
5. 4.2.2 Usage of HTTP 26
6. 4.2.3 Resources 26
7. 4.2.4 Custom Operations without associated resources 32
8. 4.2.5 Notifications 32
9. 4.2.6 Data Model 33
10. 4.2.7 Error Handling 34
11. 4.3 A1-EI (enrichment information) 35
12. 4.3.1 Introduction 35
13. 4.3.2 Usage of HTTP 35
14. 4.3.3 Resources 35
15. 4.3.4 Custom Operations without associated resources 40
16. 4.3.5 Notifications 40
17. 4.3.6 Delivery 41
18. 4.3.7 Data model 41
19. 4.3.8 Error handling 42

##### Annex A (normative): OpenAPI specification 43

1. A.1 General 43
2. A.1.1 Versioning of A1 OpenAPI documents 43
3. A.1.2 Current API versions 43
4. A.2 Policy Management API 44
5. A.3 Enrichment Information API 48

##### Annex ZZZ : O-RAN Adopter License Agreement 54

1. Section 1: DEFINITIONS 54
2. Section 2: COPYRIGHT LICENSE 54
3. Section 3: FRAND LICENSE 54
4. Section 4: TERM AND TERMINATION 55
5. Section 5: CONFIDENTIALITY 55
6. Section 6: INDEMNIFICATION 55
7. Section 7: LIMITATIONS ON LIABILITY; NO WARRANTY 56
8. Section 8: ASSIGNMENT 56
9. Section 9: THIRD-PARTY BENEFICIARY RIGHTS 56
10. Section 10: BINDING ON AFFILIATES 56
11. Section 11: GENERAL 56

5

6

7

# 1 Chapter 1 Introductory Material

## 1.1 Scope

1. This Technical Specification has been produced by the O-RAN Alliance.
2. The contents of the present document are subject to continuing work within O-RAN and may change following formal
3. O-RAN approval. Should the O-RAN Alliance modify the contents of the present document, it will be re-released by O-
4. RAN with an identifying change of release date and an increase in version number as follows:
5. Release xx.yy.zz
6. where:
7. xx the first two-digit value is incremented for all changes of substance, i.e. technical enhancements, corrections,
8. updates, etc. (the initial approved document shall have xx=01).
9. yy the second two-digit value is incremented when editorial only changes have been incorporated in the
10. document.
11. zz the third two-digit value is included only in working versions of the document indicating incremental
12. changes during the editing process; externally published documents never have this third two-digit value
13. included.
14. The present document specifies the application protocol of the A1 interface. It is part of a TS-family covering the O-
15. RAN WG2: A1 interface as identified below: “**General Aspects and Principles**”. “**Transport Protocol**”.

###### “Application Protocol”. “Type Definitions”.

1. 1.1.1 Compatibility of A1 versions
2. The version number of the present document indicates that there may be implications for the compatibility between A1
3. implementations in Non/Near-RT RICs that are based on different versions of this specification.
4. An incremented first digit of this specification could indicate that a new major feature (e.g. new A1 service) has been
5. added or that an incompatible change has been made to an A1 service. An incremented second digit could indicate that
6. an optional feature has been added, or that clarifications or corrections have been made.
7. The compatibility of A1 implementations in Non/Near-RT RICs depends on which A1 services that are implemented
8. and which version(s) of each A1 service that are implemented. The version of an A1 service is indicated by the API
9. version in the URI (see chapter 4) and compatibility is governed by the version of the OpenAPI document for the A1
10. service (see Annex A). The present document handles the service compatibility aspects while A1 interface: Type
11. Definitions [5] handles the compatibility for data types used by the A1 services.

## 1.2 References

1. The following documents contain provisions which, through reference in this text, constitute provisions of the present
2. document.
3. - References are either specific (identified by date of publication, edition number, version number, etc.) or
4. non-specific.
5. - For a specific reference, subsequent revisions do not apply.
6. - For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including
7. a GSM document), a non-specific reference implicitly refers to the latest version of that document.
8. [1] 3GPP TR 21.905: “3rd Generation Partnership Project; Technical Specification Group Services and System
9. Aspects; Vocabulary for 3GPP Specifications”
10. [2] O-RAN WG2: “Non-RT RIC and A1 interface (Use Case Requirements)”
11. [3] O-RAN WG2: “A1 interface: General Aspects and Principles”
12. [4] O-RAN WG2: “A1 interface: Transport Protocol”

|  |  |  |
| --- | --- | --- |
| 1 | [5] | O-RAN WG2: “A1 interface: Type Definitions” |
| 2 | [6] | 3GPP TS 23.501: “3rd Generation Partnership Project; Technical Specification Group Services and System |
| 3 |  | Aspects; System Architecture for the 5G System; Stage 2” |
| 4 | [7] | 3GPP TS 29.501: “3rd Generation Partnership Project; Technical Specification Group Core Network and |
| 5 |  | Terminals; 5G System; Principles and Guidelines for Services Definition; Stage 3” |
| 6 | [8] | 3GPP 29.xxx-SBI-Stage3-Template, https[://www.](http://www.3gpp.org/ftp/information/All_Templates/29.xxx-SBI-)3gp[p.org/ftp/information/All\_Templates/29.xxx-SBI-](http://www.3gpp.org/ftp/information/All_Templates/29.xxx-SBI-) |
| 7 |  | Stage3-Template.zip |
| 8 | [9] | 3GPP TS 32.158: “3rd Generation Partnership Project; Technical Specification Group Management and |
| 9 |  | orchestration; Design rules for REpresentational State Transfer (REST) Solution Sets (SS) |
| 10 | [10] | 3GPP TS 32.866: “3rd Generation Partnership Project; Technical Specification Group Services and System |
| 11 |  | Aspects; Telecommunication management; Study on a REST(REpresentational State Transfer)-ful HTTP- |
| 12 |  | based Solution Set (SS) |
| 13 | [11] | IETF RFC8259: “The JavaScript Object Notation (JSON) Data Interchange Format” |
| 14 | [12] | Semantic Versioning 2.0.0, [https://semver.org](https://semver.org/) |
| 15 | [13] | IETF RFC 3986: “Uniform Resource Identifier (URI): Generic Syntax” |
| 16 | [14] | IETF RFC7807: “Problem Details for HTTP APIs” |
| 17 | [15] | 3GPP TS 29.500: “3rd Generation Partnership Project; Technical Specification Group Core Network and |
| 18 |  | Terminals; 5G System; Technical Realization of Service Based Architecture; Stage 3” |
| 19 | [16] | OPENAPI initiative, OpenAPI 3.0.1 Specification, [http://spec.openapis.org](http://spec.openapis.org/)[/oas/v3.0.1.html](https://spec.openapis.org/oas/v3.0.0.html) |

## 1.3 Definitions and Abbreviations

### 1.3.1 Definitions

1. For the purposes of the present document, the following terms and definitions apply.
2. **A1 policy** Declarative policy that is based on a policy type, identified by its PolicyId and contains a scope
3. identifier and one or more policy statements.
4. **EI job** Description of enrichment information to be produced and delivered that is identified by its
5. EiJobId and contains a scope identifier and one or more parameters and conditions.
6. **EiJobId** Simple Data Type representing the EI job identifier.
7. **EI job identifier** Identifier of an EI job that is used for requesting and delivering A1 Enrichment Information.
8. **EI job result** The resulting enrichment information delivered based on an EI job.
9. **EI type** The model on which an EI job and its EI job result is based.
10. **EiTypeId** Simple Data Type representing the EI type identifier.
11. **EI type identifier** Identifier of an EI type.
12. **PolicyId** Simple Data Type representing the policy identifier.
13. **policy identifer** Identifier of an A1 policy that is used in policy operations.
14. **PolicyObject** Representation of an A1 policy in JSON format used as payload in HTTP based policy
15. procedures.
16. **policy statement** Expression of a goal in an A1 policy that is related to policy objectives and/or policy resources
17. and is to be applied to/for the entities identified by the scope identifier.
    1. **PolicyStatusObject** Representation of the status of an A1 policy in JSON format used as payload in HTTP based
    2. policy procedures.
    3. **PolicyTypeId** Simple Data Type representing the policy type identifier.
    4. **policy type** The model on which a PolicyObject and a PolicyStatusObject is based.
    5. **policy type identifier** Identifier of a policy type.
    6. **scope identifier** Identifier of what the statements in the policy or the EI job applies to (UE, group of UEs, slice,
    7. QoS flow, network resource or combinations thereof).

### 1.3.2 Abbreviations

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

|  |  |  |
| --- | --- | --- |
| 12 | Id | Identifier |
| 13 | JSON | JavaScript Object Notation |
| 14 | KPI | Key Performance Indicator |
| 15 | KQI | Key Quality Indicator |
| 16 | ML | Machine Learning |
| 17 | QoS | Quality of Service |
| 18 | QoE | Quality of Experience |
| 19 | REST | REpresentational State Transfer |
| 20 | RAN | Radio Access Netework |
| 21 | RT | Real Time |
| 22 | RIC | RAN Intelligent Controller |
| 23 | RRM | Radio Resouirce Management |
| 24 | S-NSSAI | Single Network Slice Selection Assistance Information |
| 25 | SMO | Service Management and Orchestration |
| 26 | SPID | Subscriber Profile IDentity |
| 27 | UE | User Equipment |
| 28 | UEId | UE Identity |
| 29 | URI | Uniform Resource Identifier |

# 30 Chapter 2 A1 Application Protocol

1. This document contains a REST method realization of the interface architecture, and policy and EI procedures
2. identified in A1 interface: Generic Aspects and Principles [3]. It is based on HTTP transport as defined in A1 interface:
3. Transport Protocol [4] and an application data model defined in A1 interface: Type Definitions [5].
4. This definition of the A1 Application Protocol (A1AP) is based on the 3GPP service framework for network functions
5. specified in 3GPP TS 23.501 [6]. It corresponds to a REST-based Solution Set and is based on the structure in the 3GPP
6. specification TS 29.501 [7] and the related TS template [8]. The design patterns for HTTP procedures are based on
7. 3GPP TS 32.158 [9] and the design patterns for JSON objects are based on 3GPP TS 32.866 [10].

# 1 Chapter 3 A1 Services

## 3.1 Introduction

1. The A1AP contains APIs for the services defined in A1 interface: Generic Aspects and Principles [3]:
2. A1-P – Policy Management Service;
3. A1-EI – Enrichment Information Service,
4. and will in the future contain an API for:
5. A1-ML – ML Model Management Service.
6. Note: Service definition and API for A1-ML are FFS.
7. The A1AP is based on signaling between an A1 service consumer and an A1 service producer residing in the Non-RT
8. RIC or in the Near-RT RIC.

non-RT RIC

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A1-P  Consumer | | A1-ML  Consumer | | A1-EI  Producer | |
|  | A1-P | | A1-ML | | A1-EI |
| A1-P  Producer | | A1-ML  Producer | | A1-EI  Consumer | |

1. near-RT RIC
2. Figure 3.1-1 Service framework for the A1 services.
3. The interactions between Service Consumer and Service Producer is based on the service framework used for 3GPP
4. Network Functions specified in 3GPP TS 23.501 [6] section 7.1.2 where requests are sent from the Consumer and
5. responses and notifications are sent from the from the Producer. It is the Producer that handles the resources on which
6. the Consumer performs operations. The terms consumer and producer does, thus, not refer to the direction of the data
7. transfer over the A1 interface.

## 3.2 Policy Management Service

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

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

1

A1-P Consumer

notificationDestination}

{

PUT

GET DELETE

POST

HTTP Client

HTTP Server

HTTP Server

HTTP Client

policytypes

policytypes/{policyTypeId}

policytypes/{policyTypeId}/policies

policytypes/{policyTypeId}/policies/{policyId}

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

2

A1-P Producer

* 1. Figure 3.2-1 HTTP roles in service framework. Arrows indicate direction of HTTP requests sent from HTTP Client to
  2. HTTP Server and HTTP responses sent from HTTP Server to HTTP Client.
  3. The A1AP realizes the A1 policy procedures defined in A1 interface: Generic Aspects and Principles [3] using HTTP
  4. operations in accordance with A1 interface: Transport Protocol [4] where a policy is represented as a JSON object in
  5. accordance with IETF RFC8259 [11] as defined in A1 interface: Type Definitions [5].

#### 3.2.1.2 Policy representation

* 1. The following principles are used for A1 policies when JSON is used as resource representation format:
  2. - A policy corresponds to a resource (in the REST sense);
  3. - A policy is represented as a JSON object referred to as a PolicyObject;
  4. - A PolicyObject contains a scope identifier and at least one policy statement (e.g. one or more policy objective
  5. statements and/or one or more policy resource statements);
  6. - A policy is identified by a policyId that is included in the URI when operation is for a single policy;
  7. - The policyId is assigned by the A1-P Consumer when the policy is created;
  8. - The A1-P Producer cannot modify or delete a policy;
  9. - Policy feedback for a specifc policy is subscribed to when the policy is created by providing a callback URI in
  10. the Create policy operation;
  11. - A PolicyObject does not contain any information related to which internal function in the Near-RT RIC that is
  12. to evaluate the policy;
  13. - The A1-P Producer indicates for which policy types policy creation is supported, and the JSON schemas for
  14. those policy types can be retrieved by the A1-P Consumer;
  15. - The A1-P Consumer cannot create, modify or delete policy types.

#### 3.2.1.3 Representation objects

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

###### PolicyTypeObject

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

###### PolicyObject

* 1. The PolicyObject is the JSON representation of an A1 policy.

###### PolicyStatusObject

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

###### ProblemDetails

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

#### 3.2.1.4 Resource identifiers

* + 1. The main URI for A1 policy types is:
    2. …/policytypes
    3. A single policy type can be operated upon by adding the value of the policy type identifier to the URI:
    4. …/policytypes/{policyTypeId}
    5. The main URI for A1 policies is:
    6. …/policytypes/{policyTypeId}/policies
    7. A single policy can be operated upon by adding the value of the policy identifier to the URI:
    8. …/policytypes/{policyTypeId}/policies/{policyId}
    9. The main URI for status of a single policy is:
    10. …/policytypes/{policyTypeId}/policies/{policyId}/status
    11. The URI for policy notification is referred to as the notificationDestination and is based on a callback URI provided
    12. when creating a policy.

### 3.2.2 Service Operations

* + 1. The following table describes the mapping between the A1 policy procedures and the HTTP methods used to realise
    2. 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 |

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

#### 3.2.2.1 Introduction

* + 1. The following sections describe the policy operations. For details on the PolicyObjects (in JSON format) transferred in
    2. the HTTP message bodies, see A1 Interface: Type Definitions [5].
    3. 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
    4. Consumer needs to map policyIds to scope identifiers in order to manage e.g. all policies applicable to a specific
    5. individual ueId. If there are several policies related to the same scope identifier, then several policy operations need to
    6. be made to manage that specific scope.

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

#### 3.2.2.2 Create policy

##### 5 3.2.2.2.1 General

1. An A1 policy is created using a HTTP PUT request containing a PolicyObject in the payload. The format of the
2. 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

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

11



1. PUT policytypes/{policyTypeId}/policies/{policyId}(PolicyObject)
2. 201 Created (PolicyObject)

A1-P Producer

A1-P Producer

A1-P Consumer

A1-P Consumer

12

1. The procedure is as follows:

Figure 3.2.2.2-1 Create policy procedure.

1. 1) The A1-P Consumer generates the policyId and sends a HTTP PUT request to the A1-P Producer. The target
2. URI identifies the resource (policyId) under which the new policy shall be created. The message body carries a
3. PolicyObject.
4. 2) The A1-P Producer returns the HTTP PUT response. On success, “201 Created” is returned. The location
5. header is present and carries the URI of the new policy and the message body the PolicyObject. On failure, the
6. appropriate error code is returned, and the message body may contain additional error information.
7. As the A1-P Producer has indicated which policy types it supports, when creating a policy, the A1-P Consumer includes
8. a policyTypeId in the URI for the PUT request. The policyTypeId is used by the A1-P Producer to select the appropriate
9. schemas to use for validation of the PolicyObject and for PolicyStatus.
10. If the provided policyTypeId is not supported or validation of the PolicyObject fails, “400 Bad Request” is returned and
11. the message body may contain additional error information.
12. If the A1-P Consumer likes to receive policy status updates related to the created policy, it includes the
13. notificationDestination as a query paremeter in the PUT request.

##### 3.2.2.2.3 Create multiple policies

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

#### 3.2.2.3 Query policy

##### 30 3.2.2.3.1 General

31 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

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

4



1. GET policytypes/{policyTypeId}/policies/{policyId}
2. 200 OK (PolicyObject)

A1-P Producer

A1-P Producer

A1-P Consumer

A1-P Consumer

5

1. The procedure is as follows:

Figure 3.2.2.3-1 Query policy procedure.

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

##### 3.2.2.3.3 Query multiple policies

1. The operation to query multiple policies is a sequence of operations to query a single policy.
2. NOTE: to query all policies applicable to e.g. a dynamically defined group of UEs, a slice or a cell, the A1-P
3. 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

1. The operation to query all policies is, for each policy type identifier retrieved as described in clause 3.2.2.7.2, a
2. 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

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

24



1. GET policytypes/{policyTypeId}/policies
2. 200 OK (array(PolicyId))

A1-P Producer

A1-P Producer

A1-P Consumer

A1-P Consumer

25

1. The procedure is as follows:

Figure 3.2.2.3-3 Query all policy identifiers procedure.

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

##### 3.2.2.3.6 Query policy status

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

8



1. GET policytypes/{policyTypeId}/policies/{policyId}/status
2. 200 OK (PolicyStatusObject)

A1-P Producer

A1-P Producer

A1-P Consumer

A1-P Consumer

9

1. The procedure is as follows:

Figure 3.2.2.3-4 Query policy status procedure.

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

#### 3.2.2.4 Update policy

##### 18 3.2.2.4.1 General

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

##### 3.2.2.4.2 Update single policy

1. The operation to update a single policy is based on HTTP PUT. The policy to be updated is identified with a URI that
2. includes the policyId and the message body contains the PolicyObject for the updated policy. 23

24



1. PUT policytypes/{policyTypeId}/policies/{policyId}(PolicyObject)
2. 200 OK (PolicyObject)

A1-P Producer

A1-P Producer

A1-P Consumer

A1-P Consumer

25

26 The procedure is as follows:

Figure 3.2.2.4-1 Update policy procedure.

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

##### 3.2.2.4.3 Update multiple policies

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

#### 3.2.2.5 Delete policy

##### 14 3.2.2.5.1 General

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

##### 3.2.2.5.2 Delete single policy

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

19



1. DELETE policytypes/{policyTypeId}/policies/{policyId}
2. 204 No Content

A1-P Producer

A1-P Producer

A1-P Consumer

A1-P Consumer

20

1. The procedure is as follows:

Figure 3.2.2.5-1 Delete policy procedure.

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

##### 3.2.2.5.3 Delete multiple policies

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

#### 3.2.2.6 Feedback policy

##### 31 3.2.2.6.1 General

1. Feedback policy is an operation that requires the A1-P Producer to have a reduced feature HTTP Client for sending
2. HTTP POST requests and receiving HTTP POST responses. Correspondingly, the A1-P Consumer is required to have a
3. reduced feature HTTP Server for receiving HTTP POST requests and sending HTTP POST responses.
4. The A1-P Producer uses the Feedback policy operation to notify the A1-P Consumer about:
5. - Changes in the policy enforcement status for an A1 policy;
6. All notifications are sent to the URI for notification handling and the PolicyStatusObject contains the information about
7. changes and causes.

##### 3.2.2.6.2 Policy status update

1. The operation to provide policy feedback is based on HTTP POST. The URI contains the target resource for policy
2. notification handling. The notification content is represented in a PolicyStatusObject that is included in the message
3. body and can contain one notification.
4. The procedure is used to notify about an enforcement status change of a policy between ‘enforced’ and ‘not enforced’.

10



1. POST {notificationDestination}(PolicyStatusObject)
2. 204 No content

A1-P Producer

A1-P Producer

A1-P Consumer

A1-P Consumer

11

1. The procedure is as follows:

Figure 3.2.2.6-1 Feedback policy procedure.

1. 1) The A1-P Producer sends a HTTP POST request to the A1-P Consumer. The target URI
2. (notificationDestination) identifies the sink for policy notifications. The message body contains a
3. PolicyStatusObject.
4. 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

##### 18 3.2.2.7.1 General

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

##### 3.2.2.7.2 Query all policy type identifiers

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

25



1. GET policytypes
2. 200 OK (array(PolicyTypeId))

A1-P Producer

A1-P Producer

A1-P Consumer

A1-P Consumer

26 Figure 3.2.2.7-1 Query all policy type identifiers procedure.

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

##### 3.2.2.7.3 Query single policy type

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

10



1. GET policytypes/{policyTypeId}
2. 200 OK (PolicyTypeObject)

A1-P Producer

A1-P Producer

A1-P Consumer

A1-P Consumer

11

1. The procedure is as follows:

Figure 3.2.2.7-2 Query policy type procedure.

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

##### 3.2.2.7.4 Query multiple policy types

1. 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

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

## 3.3 Enrichment Information Service

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

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

eitypes

eitypes/{eiTypeId}

eijobs

eijobs/{eiJobId}

eijobs/{eiJobId}/status

A1-EI Producer

* 1. A1-EI Consumer

jobStatusNotificationUri}

{jobResultUri}

GET

PUT DELETE

POST

{

HTTP Server

HTTP Client

HTTP Client

HTTP Server

* 1. Figure 3.3.1.1-1 HTTP roles in service framework. Arrows indicate direction of HTTP requests sent from HTTP Client
  2. to HTTP Server and HTTP responses sent from HTTP Server to HTTP Client.
  3. The A1AP realizes the A1 EI procedures defined in A1 interface: Generic Aspcts and Priniples [3] using HTTP
  4. operations in accordance with A1 interface: Transport Protocol [4] where EI types, jobs and job results are represented
  5. as JSON objects in accordance with RFC8259 [11] as defined in A1 interface: Type Definitions [5].

#### 3.3.1.2 EI representation

* 1. The following principles are used for A1 Enrichment Information when JSON is used as resource representation format:
  2. - The A1-EI Producer can indicate the EI types that are available;
  3. - An EI type is identified by an EI type identifier and the schemas for available EI types can be retrieved by the
  4. A1-EI Consumer;
  5. - An EI job can be created for delivery of information of a specific EI type;
  6. - An EI job corresponds to a resource (in the REST sense);
  7. - An EI job, when transferred over HTTP, is represented as a JSON object referred to as an EI job object;
  8. - An EI job object contains a scope identifier and parameters and conditions related to the EI type the delivery is
  9. for;
  10. - An EI job is identified by an EI job identifier that is included in the URI when operation is for an EI job;
  11. - The EI job identifier is assigned by the A1-EI Consumer when the EI job is created;
  12. - Status for a specific EI job can be queried and notifications can be subscribed to when the EI job is created by
  13. providing a callback URI in the create EI job operation;
  14. - An EI job object does not contain any information related to which source that produces it nor which internal
  15. function in the near-RIC that is to consume it;
  16. - EI job results are deliverd to a callback URI provided during create EI job operation;
  17. - Delivered EI that is represented as a JSON object is referred to as an EI job result object.

#### 3.3.1.3 Representation objects

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

###### EiTypeObject

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

###### EiJobObject

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

###### EiJobStatusObject

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

###### EiJobResultObject

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

###### ProblemDetails

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

#### 3.3.1.4 Resource identifiers

1. The main URI for A1 enrichment information is:
2. …/eitypes
3. A single EI type can be operated upon by adding the value of the EI type identifier to the URI:
4. …/eitypes/{eiTypeId}
5. The main URI for A1 EI jobs is:
6. …/eijobs
7. A single EI job can be operated upon by adding the value of the EI job identifier to the URI:
8. …/eijobs/{eiJobId}
9. The main URI for status of an EI job is:
10. …/eijobs/{eiJobId}/status
11. The URI for EI job status notification is referred to as the jobStatusNotificationUri and is based on a callback URI
12. provided when creating an EI job.
13. The URI for EI delivery is referred to as the jobResultUri and is based on a callback URI provided when creating an EI
14. job.

### 3.3.2 EI Discovery Service Operations

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

|  |  |
| --- | --- |
| **A1 EI procedure** | **HTTP method** |
| Query EI type identifiers | GET |
| Query EI type | GET |

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

#### 3.3.2.1 Introduction

1. The following sections describe the EI discovery operations. For further information on the EI objects transferred in the
2. HTTP message bodies, see A1 Interface: Type Definitions [5].
3. The purpose of the EI discovery procedures is for the A1-EI Consumer to
4.  identify which EI types that are available from the A1-EI producer. Each specific type of enrichment
5. information is identified by a unique EI type identifier (EiTypeId);
6.  request detailed information related to a specific EI type that can be used to create an EI job and to handle the
7. delivery of results from the EI job.

#### 1 3.3.2.2 Query EI types

##### 2 3.3.2.2.1 General

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

##### 3.3.2.2.2 Query EI type identifiers

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

8



1. GET eitypes
2. 200 OK (array(EiTypeId))

A1-EI Consumer

A1-EI Consumer

A1-EI Producer

A1-EI Producer

9

1. The procedure is as follows:

Figure 3.3.3.2.2-1 Query EI type identifiers procedure.

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

##### 3.3.2.2.3 Query EI type

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

19



1. GET eitypes/{eiTypeId}
2. 200 OK (EiTypeObject)

A1-EI Consumer

A1-EI Consumer

A1-EI Producer

A1-EI Producer

20

1. The procedure is as follows:

Figure 3.3.3.2.3-1 Query EI type procedure.

1. 1) The A1-EI Consumer sends a HTTP GET request to the A1-EI Producer. The target URI identifies the EI type
2. to be read based on the eiTypeId under the parent resource “/eitypes”. The message body is empty.
3. 2) The A1-EI Producer returns the HTTP GET response. On success, “200 OK” is returned. The message body
4. carries an EITypeObject representing the read EI type. On failure, the appropriate error code is returned, and
5. the message response body may contain additional error information.
6. The procedure can be used to query that a certain EI type is available for creation of EI jobs even if no detailed
7. information is expected in the EI type object.

### 3.3.3 EI Job Control Service Operations

1. The following table describes the mapping between the A1 EI job control procedures and the HTTP methods used to
2. 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 |

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

#### 3.3.3.1 Introduction

1. The following sections describe the EI job control operations. For further information on the EI job objects transferred
2. in the HTTP message bodies, see A1 Interface: Type Definitions [5].
3. The EI job contains a definition of the content and conditions for the delivery of the EI job result.
4. The A1-EI Producer allows the A1-EI Consumer to create EI jobs for specific EI types. The A1-EI Consumer can
5. discover the supported EI types using the Query EI types procedures. The A1-EI Consumer then indicates the EI type
6. identifier in all EI job related operations.

#### 3.3.3.2 Query EI jobs

##### 13 3.3.3.2.1 General

1. 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

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

19



1. GET .../eijobs
2. 200 OK (array(EiJobId))

A1-EI Consumer

A1-EI Consumer

A1-EI Producer

A1-EI Producer

20

21 The procedure is as follows:

Figure 3.3.3.2.2-1 Query EI job identifiers procedure.

1. 1) The A1-EI Consumer sends a HTTP GET request to the A1-EI Producer. The target URI identifies the parent
2. resource “/eijobs”. The message body is empty.
3. 2) The A1-EI Producer returns the HTTP GET response. On success, “200 OK” is returned. The message body
4. carries an array of EIJobIdentitiers representing all available EI jobs of the given EI type, or of all EI types. On
5. failure, the appropriate error code is returned, and the message response body may contain additional error
6. information.
7. If the A1-EI Consumer likes to receive EI job identifiers only related to a specific EI type, it includes the eiTypeId as a
8. query parameter in the GET request.

#### 3.3.3.3 Manage EI jobs

##### 10 3.3.3.3.1 General

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

##### 3.3.3.3.2 Create EI job

1. 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
2. object is checked, and the request is either accepted or rejected. If accepted, delivery of EI results will start based on the
3. content and conditions defined in the EI job.

17



1. PUT eijobs(EiJobObject)
2. 201 Created OK (EiJobObject)

A1-EI Consumer

A1-EI Consumer

A1-EI Producer

A1-EI Producer

18

1. The procedure is as follows:

Figure 3.3.3.3.2-1 Create EI job procedure.

1. 1) The A1-EI Consumer generates the eiJobId and sends a HTTP PUT request to the A1-EI Producer. The target
2. URI identifies the resource (“/eijobs”) under which the new EI job shall be created. The message body carries an EI
3. job object.
4. 2) The A1-EI Producer returns the HTTP PUT response. On success, “201 Created” is returned. The location
5. header is present and carries the URI of the new EI job and the message body carries the EIJobObject. On failure,
6. the appropriate error code is returned, and the message body may contain additional error information.
7. As the A1-EI Producer has indicated which EI types it supports, when creating an EI job the A1-EI Consumer includes
8. an eiTypeId in the EiJobObject. The eiTypeId is used by the A1-EI Producer to select the appropriate schemas to use
9. for validation of the EI job object and for EI job status.
10. If the provided eiTypeId is not supported or validation of the EI job object fails, “400 Bad Request” is returned and the
11. message body may contain additional error information.
12. If the A1-EI Consumer likes to receive EI job status updates related to the created EI job, it includes the
13. jobStatusNotificationUri in the EiJobObject.

##### 3.3.3.3.3 Query EI job

1. 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
2. includes the eiJobId while the message body is empty, and the response returns the EI job object.

1



1. GET .../eijobs/{eiJobId}
2. 200 OK (EiJobObject)

A1-EI Consumer

A1-EI Consumer

A1-EI Producer

A1-EI Producer

2

1. The procedure is as follows:

Figure 3.3.3.3.3-1 Query EI job procedure.

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

##### 3.3.3.3.4 Update EI job

1. 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
2. includes the eiJobId and the message body contains the EI job object for the updated EI job.

12



1. PUT .../eijobs/{eiJobId}(EiJobObject)
2. 200 OK (EiJobObject)

A1-EI Consumer

A1-EI Consumer

A1-EI Producer

A1-EI Producer

13

1. The procedure is as follows:

Figure 3.3.3.3.4-1 Update EI job procedure.

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

##### 3.3.3.3.5 Delete EI job

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

1



1. DELETE .../eijobs/{eiJobId}
2. 204 No content

A1-EI Consumer

A1-EI Consumer

A1-EI Producer

A1-EI Producer

2

1. The procedure is as follows:

Figure 3.3.3.3.5-1 Delete EI job procedure.

1. 1) The A1-EI Consumer sends a HTTP DELETE request to the A1-EI Producer. The target URI identifies the EI
2. job to be deleted based on the eiJobId under the parent resource “/eijobs”. The message body is empty.
3. 2) The A1-EI Producer returns the HTTP DELETE response. On success, “204 No Content” is returned. The
4. message body is empty. On failure, the appropriate error code is returned, and the message response body may
5. contain additional error information.

#### 3.3.3.4 Status of EI jobs

##### 10 3.3.3.4.1 General

1. 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
2. the URI of the EI job resource.
3. The A1-EI Producer uses the notify EI job status operation to notify the A1-EI Consumer about changes in status of an
4. EI job. All notifications are sent to the URI for notification handling provided during EI job creation and the
5. EiJobStatusObject contains the information about the status of the EI job.

##### 3.3.3.4.2 Query EI job status

1. 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
2. with a URI that includes the eiJobId while the message body is empty, and the response returns an EI job status object.

19



1. GET .../eijobs/{eiJobId}/status
2. 200 OK (EiJobStatusObject)

A1-EI Consumer

A1-EI Consumer

A1-EI Producer

A1-EI Producer

20

1. The procedure is as follows:

Figure 3.3.3.4.2-1 Query EI job status procedure.

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

##### 3.3.3.4.3 Notify EI job status

* 1. The operation to notify EI job status is based on HTTP POST. The URI contains the target resource for EI job
  2. notification handling. The notification content is represented in an EI job status object that is included in the message
  3. body and can contain one notification.
  4. The procedure is used to notify about a status change of an EI job.

6



1. POST {jobStatusNotificationUri}(EiJobStatusObject)
2. 204 No Content

A1-EI Consumer

A1-EI Consumer

A1-EI Producer

A1-EI Producer

7

1. The procedure is as follows:

Figure 3.3.3.4.3-1 Notify EI job status procedure.

1. 1) The A1-EI Producer sends a HTTP POST request to the A1-EI Consumer. The target URI
2. (jobStatusNotificationUri) identifies the sink for EI job status notifications. The message body contains an EI job
3. status object.
4. 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

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

|  |  |
| --- | --- |
| **A1 EI procedure** | **HTTP method** |
| Deliver EI job result | POST |

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

#### 3.3.4.1 Introduction

1. The following sections describe the EI delivery operations. For further information on the EI job result objects
2. transferred in the HTTP message bodies, see A1 Interface: Type Definitions [5].
3. The purpose of the EI delivery procedures is for the A1-EI Producer to set up appropriate connections and deliver EI
4. job results according to the service description agreed during job creation. The URL to which the EI job result is
5. delivered is transferred from the A1-EI consumer in the EI job object.

#### 3.3.4.2 Push based delivery

##### 24 3.3.4.2.1 General

1. The push-based delivery method of EI is based on subscribe-notify paradigm where the EI job creation corresponds to
2. the subscription and the EI delivery is made using HTTP POST in the same way as notifications.
3. During an EI job, the EI job results can be delivered in a single push or in several that are repeated with regular
4. intervals or irregularly based on events.

##### 3.3.4.2.2 Deliver EI job result

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

4



1. POST {jobResultUri}(EIJobResultObject)
2. 204 No Content

A1-EI Consumer

A1-EI Consumer

A1-EI Producer

A1-EI Producer

5

1. The procedure is as follows:

Figure 3.3.4.2.2-1 Deliver EI job result procedure.

1. 1) The A1-EI Producer sends a HTTP POST request to the A1-EI Consumer. The target URI (jobResultUri)
2. identifies the sink for EI job result delivieries. The message body contains an EI job result object.
3. 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

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

# 12 Chapter 4 API Definitions

## 4.1 Introduction

#### 4.1.1 Encoding of attributes in A1 data types

1. Identifiers and parameters that has been defined as integers are, when used over the A1 interface, encoded as JSON
2. "number".
3. Identifiers that have a hexadecimal or octet string representation are, when used over the A1 interface, encoded as JSON
4. "string" with character ordering preserved and zeros filling rules followed.
5. 4.1.2 Compatibility of API versions for A1 services
6. The API version and API name for each of the A1 services are defined in the following chapters. The API version is a
7. single digit that corresponds to the major version of the corresponding OpenAPI document in Annex A. Based on the
8. versioning rules for the OpenAPI documents, this implies that implementations of an A1 service in the Non/Near-RT
9. RICs are
10.  compatible if the API version is the same and any difference between the sets of supported features is handled
11. within the API version itself;
12.  not compatible in case the API versions are different.
13. The history of the introduction of an A1 service, and new API versions, is captured in the revision history of the present
14. specification. The services and versions specified in the present version of the specification is summarized in clause 29 A.1.2.
15. Note: Non/Near-RT RIC products that implement various API versions of an A1 service can still be made compatible
16. 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
17. addressed by separate URIs.

## 4.2 A1-P (policy management)

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

### 4.2.1 Introduction

1. The A1-P service shall use the A1-P API.
2. The present version of the present specification defines API version 2 (v2) of the A1-P API.
3. 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
4. the A1-P consumer towards the A1-P producer shall have the following structure:

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

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

### 4.2.2 Usage of HTTP

#### 4.2.2.1 General

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

#### 4.2.2.2 HTTP standard headers

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

#### 4.2.2.3 HTTP custom headers

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

### 4.2.3 Resources

#### 4.2.3.1 Overview

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

#### /policytypes

{apiRoot}/A1-P/v2

/{policyTypeId}

/policies

/{policyId}

/status

21

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

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

###### 1 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 |

2

1. 4.2.3.1.1 Policy type identifier
2. The PolicyTypeId is constructed based on two parts separated by “\_” (underscore):
3. typename\_version
4. where
5. typename is the unique label of the policy type;
6. version is the version of the policy type defined as major.minor.patch as described in SemVer [12].
7. The typename and version is assigned, and their uniqueness ensured, by the organizational entity that is responsible for
8. the definition and maintenance of the policy type definition.
9. Note: the typename can be based on a prefix that indicates the organizational entity (e.g. ORAN or a company
10. 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

1. 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

1. The resource represents an A1 policy.

##### 4.2.3.2.2 Resource Definition

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

##### 4.2.3.2.3 Resource Standard Methods

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

21 4.2.3.2.3.1 HTTP PUT

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

* 1. **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 |

3

1. 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 |

6 4.2.3.2.3.2 HTTP GET

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

1. **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 |

11 4.2.3.2.3.3 HTTP DELETE

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

1. **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 |

16

17 4.2.3.2.3.4 HTTP POST

1. This method is not supported on the resource.

##### 4.2.3.2.4 Resource Custom Operations

1. No custom operations are defined.

#### 4.2.3.3 Individual Policy Status Object

##### 4.2.3.3.1 Description

1. The resource represents the status of an A1 policy.

##### 4.2.3.3.2 Resource Definition

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

##### 4.2.3.3.3 Resource Standard Methods

1. The following subclauses specifies the standard methods supported by the resource.
2. Note: URI query parameters are not specified in the current version.

9 4.2.3.3.3.1 HTTP PUT

10 Method is not supported on this resource.

11 4.2.3.3.3.2 HTTP GET

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

1. **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 |

16

17 4.2.3.3.3.3 HTTP DELETE

18 Method is not supported on this resource.

19 4.2.3.3.3.4 HTTP POST

1. Method is not supported on this resource.

##### 4.2.3.3.4 Resource Custom Operations

1. No custom operations are defined.

#### 4.2.3.4 All Policy Identifiers

##### 4.2.3.4.1 Description

1. The resource represents A1 policy identifiers.

##### 4.2.3.4.2 Resource Definition

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

##### 4.2.3.4.3 Resource Standard Methods

1. The following subclauses specifies the standard methods supported by the resource.
2. Note: URI query parameters are not specified in the current version.

4 4.2.3.4.3.1 HTTP PUT

5 Method is not supported on this resource.

6 4.2.3.4.3.2 HTTP GET

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

1. **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 |

11

12 4.2.3.4.3.3 HTTP DELETE

13 Method is not supported on this resource.

14 4.2.3.4.3.4 HTTP POST

1. Method is not supported on this resource.

##### 4.2.3.4.4 Resource Custom Operations

1. No custom operations are defined.

#### 4.2.3.5 All Policy Type Identifiers

##### 4.2.3.5.1 Description

1. The resource represents A1 policy type identifiers.

##### 4.2.3.5.2 Resource Definition

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

##### 4.2.3.5.3 Resource Standard Methods

1. The following subclauses specifies the standard methods supported by the resource.
2. Note: URI query parameters are not specified in the current version.

26 4.2.3.5.3.1 HTTP PUT

27 Method is not supported on this resource.

1 4.2.3.5.3.2 HTTP GET

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

1. **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 |

6

7 4.2.3.5.3.3 HTTP DELETE

8 Method is not supported on this resource.

9 4.2.3.5.3.4 HTTP POST

1. Method is not supported on this resource.

##### 4.2.3.5.4 Resource Custom Operations

1. No custom operations are defined.

#### 4.2.3.6 Individual Policy Type Object

##### 4.2.3.6.1 Description

1. The resource represents an A1 policy type.

##### 4.2.3.6.2 Resource Definition

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

##### 4.2.3.6.3 Resource Standard Methods

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

20 4.2.3.6.3.1 HTTP PUT

21 Method is not supported on this resource.

22 4.2.3.6.3.2 HTTP GET

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

1. **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 |

3

4 4.2.3.6.3.4 HTTP DELETE

5 This method is not supported on the resource.

6 4.2.3.6.3.5 HTTP POST

1. This method is not supported on the resource.

##### 4.2.3.6.4 Resource Custom Operations

1. No custom operations are defined.

### 4.2.4 Custom Operations without associated resources

1. No custom operations are defined.

### 4.2.5 Notifications

#### 4.2.5.1 Notify Policy Status

##### 4.2.5.1.1 Description

1. The resource represents the destination for policy status notifications.

##### 4.2.5.1.2 Resource Definition

1. 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

1. The following subclauses specifies the standard methods supported by the resource.
2. Note: URI query parameters are not specified in the current version.

21 4.2.5.1.3.1 HTTP PUT

22 Method is not supported on this resource.

23 4.2.5.1.3.2 HTTP GET

24 Method is not supported on this resource.

25 4.2.5.1.3.3 HTTP DELETE

26 Method is not supported on this resource.

1 4.2.5.1.3.4 HTTP POST

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

1. **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 |

6

### 4.2.6 Data Model

1. This subclause specifies the application protocol data model supported by the A1-P API.
2. 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

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

##### 4.2.6.1.1 Simple data types

1. 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 |

16

##### 4.2.6.2 Structured data types

1. 4.2.6.2.1 Problem details
2. In case a policy request is not accepted, additional information can be provided in the response in addition to the normal
3. HTTP status code.
4. 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 |  |

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

### 4.2.7 Error Handling

#### 4.2.7.1 General

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

#### 4.2.7.2 Protocol Errors

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

#### 4.2.7.3 Application Errors

* 1. 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. |

12

## 4.3 A1-EI (enrichment information)

* + 1. This section contains the definition of the REST based API for the Enrichment Information Service referred to as A1-
    2. EI.

### 4.3.1 Introduction

* + 1. The A1-EI service shall use the A1-EI API.
    2. The present version of the present specification defines API version 1 (v1) of the A1-EI API.
    3. 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
    4. the A1-EI consumer towards the A1-EI producer shall have the following structure:

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

* + 1. where the "ResourceUriPart" shall be as be defined in subclause 4.3.3.

### 4.3.2 Usage of HTTP

#### 4.3.2.1 General

* + 1. The A1 Transport, HTTP protocol and security requirements, is described in A1 interface: Transport Protocol [4].

#### 4.3.2.2 HTTP standard headers

* + 1. Note: the encodings and applicable MIME media type for the related Content-Type header are not specified in the
    2. current version.

#### 4.3.2.3 HTTP custom headers

* + 1. No HTTP custom headers are introduced in this version of the specification.

### 4.3.3 Resources

#### 4.3.3.1 Overview

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

#### /eitypes

{apiRoot}/A1-EI/v1

/{eiTypeId}

/eijobs

/{eiJobId}

/status

22

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

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

###### 1 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 |

2

1. 4.3.3.1.1 EI type identifier
2. The EiTypeId is constructed based on two parts separated by “\_” (underscore):
3. typename\_version
4. where
5. typename is the unique label of the EI type;
6. version is the version of the EI type defined as major.minor.patch as described in SemVer [12].
7. The typename and version is assigned, and their uniqueness ensured, by the organizational entity that is responsible for
8. the definition and maintenance of the EI type definition.
9. Note: the typename can be based on a prefix that indicates the organizational entity (e.g. ORAN or a company
10. 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

1. 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

1. The resource represents EI type identifiers.

##### 4.3.3.2.2 Resource Definition

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

##### 4.3.3.2.3 Resource Standard Methods

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

22 4.3.3.2.3.1 HTTP GET

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

* 1. **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 |

3

##### 4.3.3.2.4 Resource Custom Operations

1. No custom operations are defined.

#### 4.3.3.3 Individual EI Type

##### 4.3.3.3.1 Description

1. The resource represents an EI type.

##### 4.3.3.3.2 Resource Definition

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

##### 4.3.3.3.3 Resource Standard Methods

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

13 4.3.3.3.3.1 HTTP GET

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

1. **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 |

18

##### 4.3.3.3.4 Resource Custom Operations

1. No custom operations are defined.

#### 4.3.3.4 All EI Jobs

##### 4.3.3.4.1 Description

1. The resource represents EI job identifiers.

##### 4.3.3.4.2 Resource Definition

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

##### 4.3.3.4.3 Resource Standard Methods

1. The following subclauses specifies the standard methods supported by the resource.
2. Note: URI query parameters are not specified in the current version.

6 4.3.3.4.3.1 HTTP GET

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

1. **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 |

11

1. 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 |

14

#### 4.3.3.5 Individual EI Job

##### 4.3.3.5.1 Description

1. The resource represents an EI job.

##### 4.3.3.5.2 Resource Definition

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

##### 4.3.3.5.3 Resource Standard Methods

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

22 4.3.3.5.3.1 HTTP PUT

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

* 1. **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 |

3

4 4.3.3.5.3.2 HTTP GET

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

1. **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 |

9

10 4.3.3.5.3.3 HTTP DELETE

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

1. **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 |

15

##### 4.3.3.5.4 Resource Custom Operations

1. No custom operations are defined.

#### 4.3.3.6 Individual EI Job Status

##### 4.3.3.6.1 Description

1. The resource represents the status of an EI job.

##### 4.3.3.6.2 Resource Definition

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

##### 4.3.3.6.3 Resource Standard Methods

1. The following subclauses specifies the standard methods supported by the resource.
2. Note: URI query parameters are not specified in the current version.

6 4.3.3.6.3.1 HTTP GET

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

1. **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 |

11

##### 4.2.3.3.4 Resource Custom Operations

1. No custom operations are defined.

### 4.3.4 Custom Operations without associated resources

1. No custom operations are defined.

### 4.3.5 Notifications

#### 4.3.5.1 Notify EI Job Status

##### 4.3.5.1.1 Description

1. The resource represents the destination for EI job status notifications.

##### 4.3.5.1.2 Resource Definition

1. 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

1. The following subclauses specifies the standard methods supported by the resource.
2. Note: URI query parameters are not specified in the current version.

25 4.3.5.1.3.1 HTTP POST

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

* 1. **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 |

3

### 4.3.6 Delivery

#### 4.3.6.1 Deliver EI

##### 4.3.6.1.1 Description

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

##### 4.3.6.1.2 Resource Definition

1. 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

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

12 4.3.6.1.3.1 HTTP POST

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

1. **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 |

17

### 4.3.7 Data model

1. This subclause specifies the application protocol data model supported by the A1-EI API.
2. 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

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

###### 1 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 |

2

###### 3 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

5

#### 4.3.7.2 Structured data types

##### 4.3.7.2.1 Problem details

1. 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

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

#### 4.3.8.2 Protocol Errors

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

#### 4.3.8.3 Application Errors

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

###### 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. |

2 Annex A (normative): OpenAPI specification

1. A.1 General
2. This Annex specifies the formal definition of the A1 API(s). It consists of OpenAPI documents in YAML format that
3. are based on the OpenAPI 3.0.0 Specification [16].
4. Informative copies of the OpenAPI documents contained in this O-RAN Technical Specification may be available at a
5. later stage.

### A.1.1 Versioning of A1 OpenAPI documents

1. The OpenAPI documents for the A1 services found in this chapter are versioned by Semantic Versioning 2.0.0 [12] as
2. described in the OpenAPI Specification [16]. When included in the present specification, the OpenAPI documents are
3. considered as released and are versioned using three digit major.minor.patch where the main compatibility expectations
4. stated for Sematic Versioning [12] implies:
5. major version is stepped up when incompatible API changes are made to the OpenAPI document. This corresponds
6. to saying that implementations of an A1 service in Non/Near-RT RICs are incompatible in case the API version is
7. different. The major version in the OpenAPI document corresponds to the API version in the URI for of the A1
8. service defined in chapter 4.
9. minor version is stepped up when features are added to the OpenAPI document in way that keeps implementations
10. compatible although all features are not supported by both the service producer and the service consumer of the A1
11. service.
12. patch version is stepped up when errors are corrected in a backward compatible way, when or editorial changes are
13. made to the OpenAPI document, but no features are added.
14. Note: Non/Near-RT RIC products that implement various API versions of an A1 service can be compatible by
15. supporting several API versions of an A1 service. The present specification specifies only one API version, and
16. contains only one OpenAPI document, for each A1 service.

### A.1.2 Current API versions

1. 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 |

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

## A.2 Policy Management API

* 1. openapi: 3.0.1
  2. info:
  3. title: 'A1-P Policy Management Service'
  4. version: 2.0.1
  5. description: |
  6. API for Policy Management Service.
  7. © 2021, O-RAN Alliance.
  8. All rights reserved.
  9. externalDocs:
  10. description: 'O-RAN.WG2.A1AP-v03.01 A1 interface: Application Protocol'
  11. url: ['https://www.o](http://www.o-ran.org/specifications%27)-[ran.org/specifications'](http://www.o-ran.org/specifications%27)
  12. servers:
  13. - url: '{apiRoot}/A1-P/v2'
  14. variables:
  15. apiRoot:
  16. default: 'https://example.com'
  17. description: 'apiRoot as defined in clause 4.2.1 in ORAN-WG2.A1.AP'
  18. paths:
  19. '/policytypes':
  20. get:
  21. description: 'Get all policy type identifiers'
  22. tags:
  23. - All Policy Type Identifiers
  24. responses:

26 200:

1. description: 'Array of all policy type identifiers'
2. content:
3. application/json:
4. schema:
5. type: array
6. items:
7. "$ref": "#/components/schemas/PolicyTypeId"
8. minItems: 0

35 429:

36 "$ref": "#/components/responses/429-TooManyRequests"

37 503:

38 "$ref": "#/components/responses/503-ServiceUnavailable"

39

1. '/policytypes/{policyTypeId}':
2. parameters:
3. - name: policyTypeId
4. in: path
5. required: true
6. schema:
7. "$ref": "#/components/schemas/PolicyTypeId"
8. get:
9. description: 'Get the schemas for a policy type'
10. tags:
11. - Individual Policy Type
12. responses:

52 200:

1. description: 'The policy type schemas'
2. content:
3. application/json:
4. schema:
5. "$ref": "#/components/schemas/PolicyTypeObject"

58 404:

59 "$ref": "#/components/responses/404-NotFound"

60 429:

61 "$ref": "#/components/responses/429-TooManyRequests"

62 503:

63 "$ref": "#/components/responses/503-ServiceUnavailable"

64

1. '/policytypes/{policyTypeId}/policies':
2. get:
3. description: 'Get all policy identifiers'
4. tags:
5. - All Policy Identifiers
6. parameters:
7. - name: policyTypeId
8. in: path
9. required: true
10. schema:
11. "$ref": "#/components/schemas/PolicyTypeId"

1 responses:

2 200:

1. description: 'Array of all policy identifiers'
2. content:
3. application/json:
4. schema:
5. type: array
6. items:
7. "$ref": "#/components/schemas/PolicyId"
8. minItems: 0

11 429:

12 "$ref": "#/components/responses/429-TooManyRequests"

13 503:

14 "$ref": "#/components/responses/503-ServiceUnavailable"

15

1. '/policytypes/{policyTypeId}/policies/{policyId}':
2. parameters:
3. - name: policyTypeId
4. in: path
5. required: true
6. schema:
7. "$ref": "#/components/schemas/PolicyTypeId"
8. - name: policyId
9. in: path
10. required: true
11. schema:
12. "$ref": "#/components/schemas/PolicyId"
13. put:
14. description: 'Create, or update, a policy'
15. tags:
16. - Individual Policy Object
17. parameters:
18. - name: notificationDestination
19. in: query
20. required: false
21. schema:
22. "$ref": "#/components/schemas/NotificationDestination"
23. requestBody:
24. required: true
25. content:
26. application/json:
27. schema:
28. "$ref": "#/components/schemas/PolicyObject"
29. responses:

45 200:

1. description: 'The policy was updated'
2. content:
3. application/json:
4. schema:
5. "$ref": "#/components/schemas/PolicyObject"

51 201:

1. description: 'The policy was created'
2. content:
3. application/json:
4. schema:
5. "$ref": "#/components/schemas/PolicyObject"
6. headers:
7. Location:
8. description: 'Contains the URI of the created policy'
9. required: true
10. schema:
11. type: string

63 400:

64 "$ref": "#/components/responses/400-BadRequest"

65 409:

66 "$ref": "#/components/responses/409-Conflict"

67 429:

68 "$ref": "#/components/responses/429-TooManyRequests"

69 503:

70 "$ref": "#/components/responses/503-ServiceUnavailable"

71 507:

1. "$ref": "#/components/responses/507-InsufficientStorage"
2. callbacks:
3. policyStatusNotification:
4. '{$request.query.notificationDestination}':
5. post:
6. description: 'Notify about status changes for this policy'

1

2

3

4

5

6

7

8

9

1. get:

requestBody: required: true content:

application/json: schema:

"$ref": "#/components/schemas/PolicyStatusObject" responses:

204:

description: 'Notification received'

1. description: 'Query a policy'
2. tags:
3. - Individual Policy Object
4. responses:

15 200:

1. description: 'The requested policy'
2. content:
3. application/json:
4. schema:
5. "$ref": "#/components/schemas/PolicyObject"

21 404:

22 "$ref": "#/components/responses/404-NotFound"

23 409:

24 "$ref": "#/components/responses/409-Conflict"

25 429:

26 "$ref": "#/components/responses/429-TooManyRequests"

27 503:

1. "$ref": "#/components/responses/503-ServiceUnavailable"
2. delete:
3. description: 'Delete a policy'
4. tags:
5. - Individual Policy Object
6. responses:

34 204:

35 description: 'The policy was deleted'

36 404:

37 "$ref": "#/components/responses/404-NotFound"

38 429:

39 "$ref": "#/components/responses/429-TooManyRequests"

40 503:

41 "$ref": "#/components/responses/503-ServiceUnavailable"

42

1. '/policytypes/{policyTypeId}/policies/{policyId}/status':
2. parameters:
3. - name: policyTypeId
4. in: path
5. required: true
6. schema:
7. "$ref": "#/components/schemas/PolicyTypeId"
8. - name: policyId
9. in: path
10. required: true
11. schema:
12. "$ref": "#/components/schemas/PolicyId"
13. get:
14. description: 'Query a policy status'
15. tags:
16. - Individual Policy Status Object
17. responses:

60 200:

1. description: 'The requested policy status'
2. content:
3. application/json:
4. schema:
5. "$ref": "#/components/schemas/PolicyStatusObject"

66 404:

67 "$ref": "#/components/responses/404-NotFound"

68 409:

69 "$ref": "#/components/responses/409-Conflict"

70 429:

71 "$ref": "#/components/responses/429-TooManyRequests"

72 503:

73 "$ref": "#/components/responses/503-ServiceUnavailable"

74

1. components:
2. schemas:

77 #

1 # Representation objects

2 #

1. PolicyObject:
2. description: 'A generic policy object that can be used to transport any policy. Additionally,
3. a policy shall be valid according to the schema of its specific policy type.'
4. type: object

7

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

12

1. PolicyTypeObject:
2. description: 'A definition of a policy type, i.e. the schemas for a policy respectively its
3. status'
4. type: object
5. properties:
6. policySchema:
7. "$ref": "#/components/schemas/JsonSchema"
8. statusSchema:
9. "$ref": "#/components/schemas/JsonSchema"
10. required:
11. - policySchema

24

1. ProblemDetails:
2. description: 'A problem detail to carry details in a HTTP response according to RFC 7807'
3. type: object
4. properties:
5. type:
6. type: string
7. title:
8. type: string
9. status:
10. type: number
11. detail:
12. type: string
13. instance:
14. type: string

39

40 #

41 # Simple data types

42 #

1. JsonSchema:
2. description: 'A JSON schema following [http://json-schema.org/draft-07/schema'](http://json-schema.org/draft-07/schema%27)
3. type: object

46

1. NotificationDestination:
2. description: 'A complete callback URI defined according to IETF RFC 3986 where to send
3. notifications'
4. type: string

51

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

55

1. PolicyTypeId:
2. description: 'Policy type identifier assigned by the A1-P Provider'
3. type: string

59

1. responses:
2. 400-BadRequest:
3. description: 'Object in payload not properly formulated or not related to the method'
4. content:
5. application/problem+json:
6. schema:
7. "$ref": "#/components/schemas/ProblemDetails"

67

1. 404-NotFound:
2. description: 'No resource found at the URI'
3. content:
4. application/problem+json:
5. schema:
6. "$ref": "#/components/schemas/ProblemDetails"

74

1. 405-MethodNotAllowed:
2. description: 'Method not allowed for the URI'
3. content:

1

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"

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

## 35 A.3 Enrichment Information API

36

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](http://www.o-ran.org/specifications)-[ran.org/specifications](http://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:

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

1. description: Unauthorized
2. content: {}

3 403:

1. description: Forbidden
2. content: {}

6 404:

1. description: Enrichment Information job is not found
2. content:
3. application/json:
4. schema:
5. $ref: '#/components/schemas/ProblemDetails'
6. deprecated: false
7. put:
8. tags:
9. - A1-EI (enrichment information)
10. summary: Individual EI job
11. operationId: putIndividualEiJobUsingPUT
12. parameters:
13. - name: eiJobId
14. in: path
15. description: eiJobId
16. required: true
17. schema:
18. type: string
19. requestBody:
20. description: eiJobObject
21. content:
22. application/json:
23. schema:
24. $ref: '#/components/schemas/EiJobObject'
25. required: true
26. responses:

33 200:

1. description: Job updated
2. content: {}

36 201:

1. description: Job created
2. content: {}

39 401:

1. description: Unauthorized
2. content: {}

42 403:

1. description: Forbidden
2. content: {}

45 404:

1. description: Enrichment Information type is not found
2. content:
3. application/json:
4. schema:
5. $ref: '#/components/schemas/ProblemDetails'
6. deprecated: false
7. x-codegen-request-body-name: eiJobObject
8. callbacks:
9. jobStatusNotification:
10. '{$request.body.jobStatusNotificationUri}':
11. post:
12. description: 'Notify about status changes for this EI job'
13. requestBody:
14. required: true
15. content:
16. application/json:
17. schema:
18. "$ref": "#/components/schemas/EiJobStatusObject"
19. responses:

65 204:

1. description: 'Notification received'
2. jobResult:
3. '{$request.body.jobResultUri}':
4. post:
5. description: 'Deliverance of EI'
6. requestBody:
7. required: true
8. content:
9. application/json:
10. schema:
11. "$ref": "#/components/schemas/EiResultObject"
12. responses:

1

2

* 1. delete:
  2. tags:

204:

description: 'Information received'

* 1. - A1-EI (enrichment information)
  2. summary: Individual EI job
  3. operationId: deleteIndividualEiJobUsingDELETE
  4. parameters:
  5. - name: eiJobId
  6. in: path
  7. description: eiJobId
  8. required: true
  9. schema:
  10. type: string
  11. responses:

16 200:

1. description: Not used
2. content: {}

19 204:

1. description: Job deleted
2. content: {}

22 401:

1. description: Unauthorized
2. content: {}

25 403:

1. description: Forbidden
2. content: {}

28 404:

1. description: Enrichment Information job is not found
2. content:
3. application/json:
4. schema:
5. $ref: '#/components/schemas/ProblemDetails'
6. deprecated: false
7. /A1-EI/v1/eitypes/{eiTypeId}:
8. get:
9. tags:
10. - A1-EI (enrichment information)
11. summary: Individual EI type
12. operationId: getEiTypeUsingGET
13. parameters:
14. - name: eiTypeId
15. in: path
16. description: eiTypeId
17. required: true
18. schema:
19. type: string
20. responses:

49 200:

1. description: EI type
2. content:
3. application/json:
4. schema:
5. $ref: '#/components/schemas/EiTypeObject'

55 401:

1. description: Unauthorized
2. content: {}

58 403:

1. description: Forbidden
2. content: {}

61 404:

1. description: Enrichment Information type is not found
2. content:
3. application/json:
4. schema:
5. $ref: '#/components/schemas/ProblemDetails'
6. deprecated: false
7. /A1-EI/v1/eijobs:
8. get:
9. tags:
10. - A1-EI (enrichment information)
11. summary: EI job identifiers
12. description: query for EI job identifiers
13. operationId: getEiJobIdsUsingGET
14. parameters:
15. - name: eiTypeId
16. in: query
17. description: selects EI jobs of matching EI type
18. allowEmptyValue: false
19. schema:
20. type: string
21. - name: owner
22. in: query
23. description: selects EI jobs for one EI job owner
24. allowEmptyValue: false
25. schema:
26. type: string
27. responses:

12 200:

1. description: EI job identifiers
2. content:
3. application/json:
4. schema:
5. type: array
6. items:
7. type: string

20 401:

1. description: Unauthorized
2. content: {}

23 403:

1. description: Forbidden
2. content: {}

26 404:

1. description: Enrichment Information type is not found
2. content:
3. application/json:
4. schema:
5. $ref: '#/components/schemas/ProblemDetails'
6. deprecated: false
7. /A1-EI/v1/eijobs/{eiJobId}/status:
8. get:
9. tags:
10. - A1-EI (enrichment information)
11. summary: EI job status
12. operationId: getEiJobStatusUsingGET
13. parameters:
14. - name: eiJobId
15. in: path
16. description: eiJobId
17. required: true
18. schema:
19. type: string
20. responses:

47 200:

1. description: EI job status
2. content:
3. application/json:
4. schema:
5. $ref: '#/components/schemas/EiJobStatusObject'

53 401:

1. description: Unauthorized
2. content: {}

56 403:

1. description: Forbidden
2. content: {}

59 404:

1. description: Enrichment Information job is not found
2. content:
3. application/json:
4. schema:
5. $ref: '#/components/schemas/ProblemDetails'
6. deprecated: false
7. /A1-EI/v1/eitypes:
8. get:
9. tags:
10. - A1-EI (enrichment information)
11. summary: EI type identifiers
12. operationId: getEiTypeIdentifiersUsingGET
13. responses:

73 200:

1. description: EI type identifiers
2. content:
3. application/json:
4. schema:

1

2

3

4 401:

type: array items:

type: string

* 1. description: Unauthorized
  2. content: {}

7 403:

1. description: Forbidden
2. content: {}

10 404:

1. description: Not Found
2. content: {}
3. deprecated: false
4. components:
5. schemas:
6. EiTypeObject:
7. title: EiTypeObject
8. type: object
9. description: Information for an EI type
10. ProblemDetails:
11. title: ProblemDetails
12. type: object
13. properties:
14. detail:
15. type: string
16. description: A human-readable explanation specific to this occurrence of
17. the problem.
18. example: EI job type not found
19. status:
20. type: integer
21. description: The HTTP status code generated by the origin server for this
22. occurrence of the problem.
23. format: int32
24. example: 404
25. type:
26. type: string
27. title:
28. type: string
29. instance:
30. type: string
31. description: A problem detail to carry details in a HTTP response according
32. to RFC 7807
33. EiJobStatusObject:
34. title: EiJobStatusObject
35. required:
36. - eiJobStatus
37. type: object
38. properties:
39. eiJobStatus:
40. type: string
41. description: |-
42. values:
43. ENABLED: the A1-EI producer is able to deliver EI result for the EI job
44. DISABLED: the A1-EI producer is unable to deliver EI result for the EI job
45. enum:
46. - ENABLED
47. - DISABLED
48. description: Status for an EI job
49. EiJobObject:
50. title: EiJobObject
51. required:
52. - eiTypeId
53. - jobDefinition
54. - jobResultUri
55. type: object
56. properties:
57. eiTypeId:
58. type: string
59. description: EI type Idenitifier of the EI job
60. jobResultUri:
61. type: string
62. description: The target URI of the EI data
63. jobStatusNotificationUri:
64. type: string
65. description: The target of EI job status notifications
66. jobDefinition:
67. type: object

1

properties: {}

description: EI type specific job data

description: Information for an Enrichment Information Job EiResultObject:

title: EiResultObject

2

3

4

5

6

# 1 Annex ZZZ : O-RAN Adopter License Agreement

* 1. BY DOWNLOADING, USING OR OTHERWISE ACCESSING ANY O-RAN SPECIFICATION, ADOPTER
  2. AGREES TO THE TERMS OF THIS AGREEMENT.
  3. This O-RAN Adopter License Agreement (the “Agreement”) is made by and between the O-RAN Alliance and the
  4. entity that downloads, uses or otherwise accesses any O-RAN Specification, including its Affiliates (the “Adopter”).
  5. This is a license agreement for entities who wish to adopt any O-RAN Specification.

## Section 1: DEFINITIONS

* 1. 1.1 “Affiliate” means an entity that directly or indirectly controls, is controlled by, or is under common control with
  2. another entity, so long as such control exists. For the purpose of this Section, “Control” means beneficial ownership of
  3. fifty (50%) percent or more of the voting stock or equity in an entity.
  4. 1.2 “Compliant Implementation” means any system, device, method or operation (whether implemented in hardware,
  5. software or combinations thereof) that fully conforms to a Final Specification.
  6. 1.3 “Adopter(s)” means all entities, who are not Members, Contributors or Academic Contributors, including their
  7. Affiliates, who wish to download, use or otherwise access O-RAN Specifications.
  8. 1.4 “Minor Update” means an update or revision to an O-RAN Specification published by O-RAN Alliance that does
  9. not add any significant new features or functionality and remains interoperable with the prior version of an O-RAN
  10. Specification. The term “O-RAN Specifications” includes Minor Updates.
  11. 1.5 “Necessary Claims” means those claims of all present and future patents and patent applications, other than design
  12. patents and design registrations, throughout the world, which (i) are owned or otherwise licensable by a Member,
  13. Contributor or Academic Contributor during the term of its Member, Contributor or Academic Contributorship; (ii)
  14. such Member, Contributor or Academic Contributor has the right to grant a license without the payment of
  15. consideration to a third party; and (iii) are necessarily infringed by a Compliant Implementation (without considering
  16. any Contributions not included in the Final Specification). A claim is necessarily infringed only when it is not possible
  17. on technical (but not commercial) grounds, taking into account normal technical practice and the state of the art
  18. generally available at the date any Final Specification was published by the O-RAN Alliance or the date the patent
  19. claim first came into existence, whichever last occurred, to make, sell, lease, otherwise dispose of, repair, use or operate
  20. a Compliant Implementation without infringing that claim. For the avoidance of doubt in exceptional cases where a
  21. Final Specification can only be implemented by technical solutions, all of which infringe patent claims, all such patent
  22. claims shall be considered Necessary Claims.
  23. 1.6 “Defensive Suspension” means for the purposes of any license grant pursuant to Section 3, Member, Contributor,
  24. Academic Contributor, Adopter, or any of their Affiliates, may have the discretion to include in their license a term
  25. allowing the licensor to suspend the license against a licensee who brings a patent infringement suit against the
  26. licensing Member, Contributor, Academic Contributor, Adopter, or any of their Affiliates.

## Section 2: COPYRIGHT LICENSE

* 1. 2.1 Subject to the terms and conditions of this Agreement, O-RAN Alliance hereby grants to Adopter a nonexclusive,
  2. nontransferable, irrevocable, non-sublicensable, worldwide copyright license to obtain, use and modify O-RAN
  3. Specifications, but not to further distribute such O-RAN Specification in any modified or unmodified way, solely in
  4. furtherance of implementations of an ORAN
  5. Specification.
  6. 2.2 Adopter shall not use O-RAN Specifications except as expressly set forth in this Agreement or in a separate written
  7. agreement with O-RAN Alliance.

## Section 3: FRAND LICENSE

* 1. 3.1 Members, Contributors and Academic Contributors and their Affiliates are prepared to grant based on a separate
  2. Patent License Agreement to each Adopter under Fair Reasonable And Non- Discriminatory (FRAND) terms and
  3. conditions with or without compensation (royalties) a nonexclusive, non-transferable, irrevocable (but subject to
  4. Defensive Suspension), non-sublicensable, worldwide patent license under their Necessary Claims to make, have made,
  5. use, import, offer to sell, lease, sell and otherwise distribute Compliant Implementations; provided, however, that such
  6. license shall not extend: (a) to any part or function of a product in which a Compliant Implementation is incorporated

1. that is not itself part of the Compliant Implementation; or (b) to any Adopter if that Adopter is not making a reciprocal
2. grant to Members, Contributors and Academic Contributors, as set forth in Section 3.3. For the avoidance of doubt, the
3. foregoing licensing commitment includes the distribution by the Adopter’s distributors and the use by the Adopter’s
4. customers of such licensed Compliant Implementations.
5. 3.2 Notwithstanding the above, if any Member, Contributor or Academic Contributor, Adopter or their Affiliates has
6. reserved the right to charge a FRAND royalty or other fee for its license of Necessary Claims to Adopter, then Adopter
7. is entitled to charge a FRAND royalty or other fee to such Member, Contributor or Academic Contributor, Adopter and
8. its Affiliates for its license of Necessary Claims to its licensees.
9. 3.3 Adopter, on behalf of itself and its Affiliates, shall be prepared to grant based on a separate Patent License
10. Agreement to each Members, Contributors, Academic Contributors, Adopters and their Affiliates under Fair
11. Reasonable And Non-Discriminatory (FRAND) terms and conditions with or without compensation (royalties) a
12. nonexclusive, non-transferable, irrevocable (but subject to Defensive Suspension), non-sublicensable, worldwide patent
13. license under their Necessary Claims to make, have made, use, import, offer to sell, lease, sell and otherwise distribute
14. Compliant Implementations; provided, however, that such license will not extend: (a) to any part or function of a
15. product in which a Compliant Implementation is incorporated that is not itself part of the Compliant Implementation; or
16. (b) to any Members, Contributors, Academic Contributors, Adopters and their Affiliates that is not making a reciprocal
17. grant to Adopter, as set forth in Section 3.1. For the avoidance of doubt, the foregoing licensing commitment includes
18. the distribution by the Members’, Contributors’, Academic Contributors’, Adopters’ and their Affiliates’ distributors
19. and the use by the Members’, Contributors’, Academic Contributors’, Adopters’ and their Affiliates’ customers of such
20. licensed Compliant Implementations.

## Section 4: TERM AND TERMINATION

1. 4.1 This Agreement shall remain in force, unless early terminated according to this Section 4.
2. 4.2 O-RAN Alliance on behalf of its Members, Contributors and Academic Contributors may terminate this Agreement
3. if Adopter materially breaches this Agreement and does not cure or is not capable of curing such breach within thirty
4. (30) days after being given notice specifying the breach.
5. 4.3 Sections 1, 3, 5 - 11 of this Agreement shall survive any termination of this Agreement. Under surviving Section 3,
6. after termination of this Agreement, Adopter will continue to grant licenses (a) to entities who become Adopters after
7. the date of termination; and (b) for future versions of ORAN Specifications that are backwards compatible with the
8. version that was current as of the date of termination.

## Section 5: CONFIDENTIALITY

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

## Section 6: INDEMNIFICATION

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

## Section 7: LIMITATIONS ON LIABILITY; NO WARRANTY

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

## Section 8: ASSIGNMENT

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

## Section 9: THIRD-PARTY BENEFICIARY RIGHTS

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

## Section 10: BINDING ON AFFILIATES

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

## Section 11: GENERAL

1. This Agreement is governed by the laws of Germany without regard to its conflict or choice of law provisions.
2. This Agreement constitutes the entire agreement between the parties as to its express subject matter and expressly
3. supersedes and replaces any prior or contemporaneous agreements between the parties, whether written or oral, relating
4. to the subject matter of this Agreement.
5. Adopter, on behalf of itself and its Affiliates, agrees to comply at all times with all applicable laws, rules and
6. regulations with respect to its and its Affiliates’ performance under this Agreement, including without limitation, export
7. control and antitrust laws. Without limiting the generality of the foregoing, Adopter acknowledges that this Agreement
8. prohibits any communication that would violate the antitrust laws.
9. By execution hereof, no form of any partnership, joint venture or other special relationship is created between Adopter,
10. or O-RAN Alliance or its Members, Contributors or Academic Contributors. Except as expressly set forth in this
11. Agreement, no party is authorized to make any commitment on behalf of Adopter, or O-RAN Alliance or its Members,
12. Contributors or Academic Contributors.
13. In the event that any provision of this Agreement conflicts with governing law or if any provision is held to be null,
14. void or otherwise ineffective or invalid by a court of competent jurisdiction, (i) such provisions will be deemed stricken
15. from the contract, and (ii) the remaining terms, provisions, covenants and restrictions of this Agreement will remain in
16. full force and effect.
17. Any failure by a party or third party beneficiary to insist upon or enforce performance by another party of any of the
18. provisions of this Agreement or to exercise any rights or remedies under this Agreement or otherwise by law shall not
19. be construed as a waiver or relinquishment to any extent of the other parties’ or third party beneficiary’s right to assert
20. or rely upon any such provision, right or remedy in that or any other instance; rather the same shall be and remain in full
21. force and effect.