O-RAN.WG2.A1AP-R003-v04.00

Technical Specification

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

A1 interface: Application Protocol

Contents

Foreword

This Technical Specification (TS) has been produced by O-RAN Alliance Working Group 2 (Non-RT RIC and A1 interface WG). It is part of a TS-family covering the A1 interface as identified below:

- "A1 interface: General Aspects and Principles";
- "A1 interface: Use Cases and Requirements";
- "A1 interface: Transport Protocol";
- "A1 interface: Application Protocol";
- "A1 interface: Type Definitions"; and
- "A1 interface: Test Specification".

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the O-RAN Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in O-RAN deliverables except when used in direct citation.

1 Scope

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

version xx.yy.zz

where:

xx: the first digit-group is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc. (the initial approved document will have xx=01). Always 2 digits with leading zero if needed.

yy: the second digit-group is incremented when editorial only changes have been incorporated in the document. Always 2 digits with leading zero if needed.

zz: the third digit-group included only in working versions of the document indicating incremental changes during the editing process. External versions never include the third digit-group. Always 2 digits with leading zero if needed.

The present document specifies the application protocol of the A1 interface It includes service definitions and API definitions for the A1 policy management service (A1-P) and the A1 enrichment informatnin service (A1-EI).

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, O-RAN cannot guarantee their long-term validity.

The following referenced documents are necessary for the application of the present document.

[1] O-RAN TS: "Non-RT RIC & A1 interface: Use Cases and Requirements" (“A1UCR”)

[2] O-RAN TS: "A1 interface: General Aspects and Principles" ("A1GAP")

[3] O-RAN TS: "A1 interface: Transport Protocol" ("A1TP")

[4] O-RAN TS: "A1 interface: Type Definitions" ("A1TD")

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

[6] 3GPP TS 29.501: "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; 5G System; Principles and Guidelines for Services Definition; Stage 3"

[7] IETF RFC 8259: "The JavaScript Object Notation (JSON) Data Interchange Format"

[8] SemVer: "Semantic Versioning 2.0.0", https://semver.org

[9] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax"

[10] IETF RFC 7807: "Problem Details for HTTP APIs"

[11] 3GPP TS 29.500: "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; 5G System; Technical Realization of Service Based Architecture; Stage 3"

[12] OpenAPI Initiative: "OpenAPI 3.0.1 Specification", http://spec.openapis.org/oas/v3.0.1.html

[13] IANA: "Hypertext Transfer Protocol (HTTP) Status Code Registry", https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, O-RAN cannot guarantee their long-term validity.

The following referenced documents are not necessary for the application of the present document, but they assist the user with regard to a particular subject area.

[i1] 3GPP 29.xxx-SBI-Stage3-Template, https://www.3gpp.org/ftp/information/All_Templates/29.xxx-SBI-Stage3-Template.zip

[i2] 3GPP TS 32.158: "3rd Generation Partnership Project; Technical Specification Group Management and orchestration; Design rules for REpresentational State Transfer (REST) Solution Sets (SS)"

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in A1GAP [2] and the following apply:

EiJobId: Simple Data Type representing the EI job identifier.

EI job identifier: identifier of an EI job that is used for requesting and delivering A1 enrichment information.

EI job result: the resulting enrichment information delivered based on an EI job.

EiTypeId: Simple Data Type representing the EI type identifier.

EI type identifier: identifier of an EI type.

PolicyId: Simple Data Type representing the policy identifier.

policy identifier: identifier of an A1 policy that is used in policy operations.

PolicyObject: representation of an A1 policy in JSON format used as payload in HTTP based policy procedures.

policy statement: expression of a goal in an A1 policy that is related to policy objectives and/or policy resources and is to be applied to/for the entities identified by the scope identifier.

PolicyStatusObject: representation of the status of an A1 policy in JSON format used as payload in HTTP based policy procedures.

PolicyTypeId: Simple Data Type representing the policy type identifier.

policy type: the model on which a PolicyObject and a PolicyStatusObject is based.

policy type identifier: identifier of a policy type.

scope identifier: identifier of what the statements in the policy or the EI job applies to (UE, group of UEs, slice, QoS flow, network resource or combinations thereof).

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in A1GAP [2] and the following apply:

Id	Identifier
ML	Machine Learning
QoS	Quality of Service
QoE	Quality of Experience
REST	REpresentational State Transfer
RAN	Radio Access Netework
RRM	Radio Resouirce Management
S-NSSAI	Single Network Slice Selection Assistance Information
SPID	Subscriber Profile IDentity
UEId	UE Identity
URI	Uniform Resource Identifier

4 A1 Application Protocol

4.1 Introduction

The present document specifies a REST realisation of the A1 interface architecture, and the policy and EI procedures identified in A1GAP [2]. It is based on HTTP as defined in A1 interface: Transport Protocol [3] and an application data model defined in A1TD [4].

This definition of the A1 Application Protocol (A1AP) is based on the 3GPP service framework for network functions specified in 3GPP TS 23.501 [5]. It corresponds to a REST-based Solution Set and is based on the structure in the 3GPP specification TS 29.501 [6] and the related TS template [i1]. The design patterns for HTTP procedures and JSON objects are based on 3GPP TS 32.158 [i2].

4.2 Compatibility of A1 versions

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

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

The compatibility of A1 implementations in Non/Near-RT RICs depends on the A1 services that are implemented and which version(s) of each A1 service that are implemented. The version of an A1 service is indicated by the API version in the URI (see clauses 6.2.1 and 6.3.1) and compatibility is governed by the version of the OpenAPI document for the A1 service (see Annex A). The present document handles the service compatibility aspects while A1TD [4] handles the compatibility for data types used by the A1 services.

5 A1 services

5.1 Introduction

The present document specifies the APIs for the following services defined in A1GAP [2]:

A1-P: A1 policy management service;

A1-EI: A1 enrichment information service

NOTE: Service definition and API for A1-ML are not defined in the present document.

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

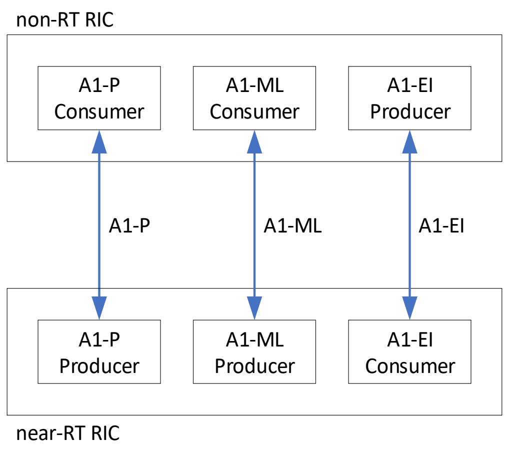


Figure 5.1-1 Service framework for the A1 services.

The interactions between Service Consumer and Service Producer are based on the service framework used for 3GPP Network Functions specified in 3GPP TS 23.501 [5] section 7.1.2 where requests are sent from the Consumer and responses and notifications are sent from the from the Producer. It is the Producer that handles the resources on which the Consumer performs operations. The terms consumer and producer do not refer to the direction of the data transfer over the A1 interface.

5.2 Policy management service

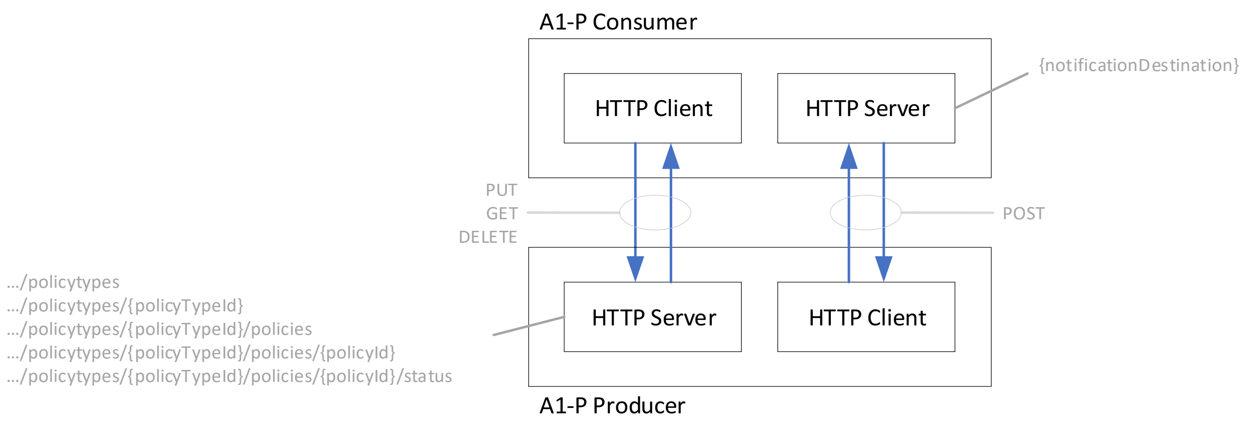
5.2.1 Introduction

The A1-P service defines service operations that are used with policy types defined in A1TD [4].

5.2.2 Service description

5.2.2.1 Functional elements

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


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

NOTE: Arrows indicate direction of HTTP requests sent from HTTP Client to HTTP Server and HTTP responses sent from HTTP Server to HTTP Client.

Figure 5.2.2.1-1 HTTP roles in service framework.

The present document specifies the A1 policy procedures defined in A1GAP [2] using HTTP operations in accordance with A1TP [3] where a policy is represented as a JSON object in accordance with IETF RFC 8259 [7] as defined in A1TD [4].

5.2.2.2 Policy representation

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

- A policy corresponds to a resource (in the REST sense);
- A policy is represented as a JSON object referred to as a PolicyObject;
- A PolicyObject contains a scope identifier and at least one policy statement (e.g., one or more policy objective statements and/or one or more policy resource statements);
- A policy is identified by a policyId that is included in the URI when an operation is for a single policy;
- The policyId is assigned by the A1-P Consumer when the policy is created;

- The A1-P Producer cannot modify or delete a policy;
- Policy feedback for a specifc policy is subscribed to when the policy is created by providing a callback URI in the Create policy operation;
- A PolicyObject does not contain any information related to which internal function in the Near-RT RIC that is to evaluate the policy;
- The A1-P Producer indicates for which policy types policy creation is supported, and the JSON schemas for policy types can be retrieved by the A1-P Consumer; and
- The A1-P Consumer cannot create, modify, or delete policy types.

5.2.2.3 Representation objects

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

PolicyTypeObject

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

PolicyObject

The PolicyObject is the JSON representation of an A1 policy.

PolicyStatusObject

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

ProblemDetails

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

5.2.2.4 Resource identifiers

The URI for A1 policy types is:

…/policytypes

A single policy type is identified by adding the value of the policy type identifier to the URI:

…/policytypes/{policyTypeId}

The URI for A1 policies is:

…/policytypes/{policyTypeId}/policies

A single policy is identified by adding the value of the policy identifier to the URI:

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

The URI for status of a single policy is:

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

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

5.2.3 Service operations for A1 policy types

5.2.3.1 Introduction

Table 5.2.3.1-1 describes the mapping between the A1 policy type operations and the HTTP methods used to realise them, and the mandatory HTTP status codes for the operations.

Table 5.2.3.1-1 A1 policy operations to HTTP methods mapping.

Service operation	HTTP method	HTTP status codes
Query policy type identifiers	GET	200
Query policy type	GET	200, 404

The following clauses describe the policy type operations.

NOTE: The present document does not define any limits for how many policyTypeId that can be transferred in a single A1 message.

5.2.3.2 Query policy type identifiers

5.2.3.2.1 General

The A1-P Consumer uses the Query policy type identifiers operation to discover policy types.

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

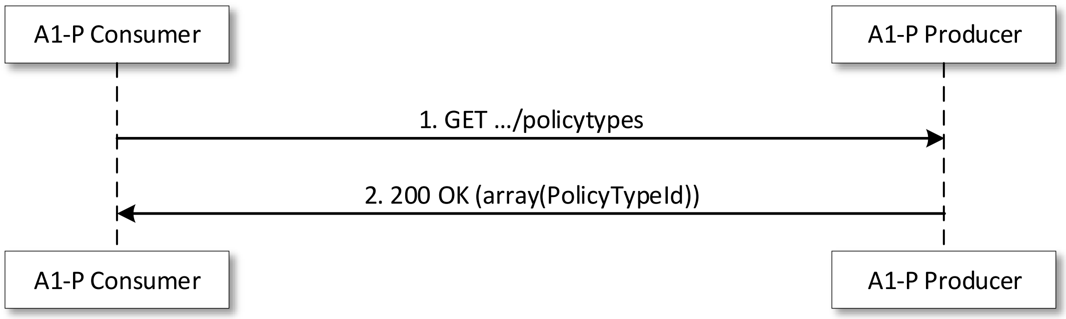


Figure 5.2.3.2.1-1 Query policy type identifiers operation.

1) The A1-P Consumer shall send an HTTP GET request to the A1-P Producer. The target URI shall identify the resource "/policytypes". The message body shall be empty.

2) The A1-P Producer shall return the HTTP GET response. On success, "200 OK" shall be returned. The message body shall carry an array of policy type identifiers representing all available policy types. On failure, the appropriate error code shall be returned, and the response message body may contain additional error information.

5.2.3.2.2 Query all policy type identifiers procedure

The procedure to query all policy type identifiers is based on the Query policy type identifiers operation illustrated in figure 5.2.3.2.1-1.

5.2.3.3 Query policy type

5.2.3.3.1 General

A1-P Consumer uses the Query policy type procedures to read the schemas for a specific policy type or for all policy types. The Query policy type operation is used in the following procedures:

- Query single policy type;
- Query multiple policy types; and
- Query all policy types.

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

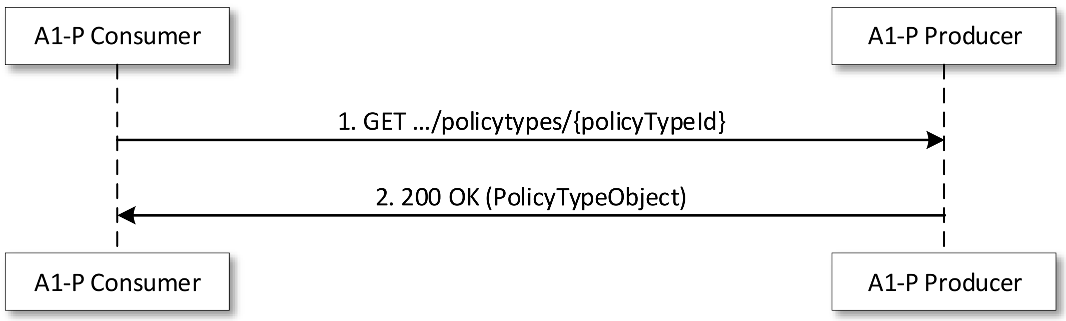

A1-P Consumer
A1-P Producer
1. GET …/policytypes/{policyTypeId}
2. 200 OK (PolicyTypeObject)
A1-P Consumer
A1-P Producer

Figure 5.2.3.3.1-1 Query policy type operation.

1) The A1-P Consumer shall send an HTTP GET request to the A1-P Producer. The target URI shall identify the policy type to be read based on the policyTypeId under the resource "/policytypes". The message body shall be empty.
2) The A1-P Producer shall return the HTTP GET response. On success, "200 OK" shall be returned. The message body shall carry a PolicyTypeObject representing the read policy type. On failure, the appropriate error code shall be returned, and the response message body may contain additional error information.

On reception of a policyTypeId that does not exist, "404 Not Found" shall be returned.

5.2.3.3.2 Query single policy type procedure

The procedure to query single policy type is based on the Query policy type operation illustrated in figure 5.2.3.3.1-1.

5.2.3.3.3 Query multiple policy types procedure

The procedure to query multiple policy types is a sequence of Query policy type operations.

5.2.3.3.4 Query all policy types procedure

The procedure to query all policy types is a sequence of Query policy type operations for each policy type identifier retrieved as described in clause 5.2.3.2.1.

5.2.4 Service operations for A1 policies

5.2.4.1 Introduction

Table 5.2.4.1-1 describes the mapping between the A1 policy operations and the HTTP methods used to realise them, and the mandatory HTTP status codes for the operations.

Table 5.2.4.1-1 A1 policy operations to HTTP methods mapping.

Service operation	HTTP method	HTTP status codes
Query policy identifiers	GET	200, 404
Create policy	PUT	201, 400, 404, 409
Update policy	PUT	200, 400, 409
Query policy	GET	200, 404
Delete policy	DELETE	204, 404
Query policy status	GET	200, 404
Notify policy status change	POST	204, 400

The following clauses describe the policy operations. For details on the PolicyObjects (in JSON format) transferred in the HTTP message bodies, see A1TD [4].

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

The A1-P Producer enables the A1-P Consumer to create policies of specific types and the A1-P Consumer can discover the supported policy types. The A1-P Consumer indicates the policyTypeId when creating or updating a policy and when querying for a specific policy.

NOTE: The present document does not define any limits for how many policyId that can be transferred in a single A1 message.

5.2.4.2 Query policy identifiers

5.2.4.2.1 General

The A1-P Consumer uses the Query policy identifiers operation to discover policies for a specific policy type or for all policy types. The Query policy identifiers operation is used in the following procedures:

- Query policy identifiers; and
- Query all policy identifiers.

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

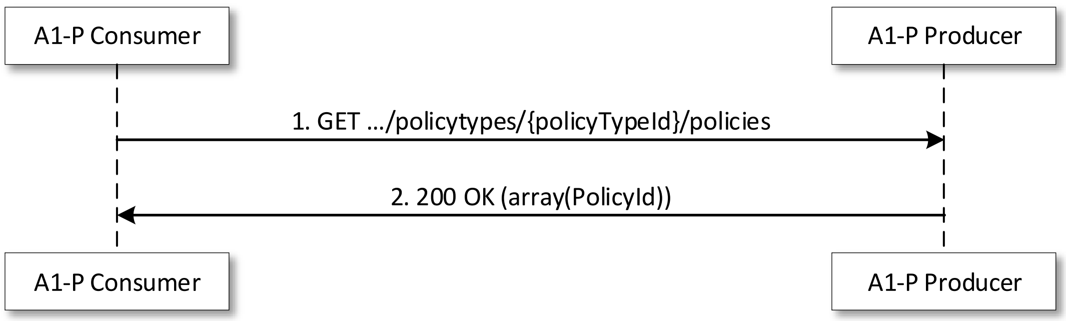

A1-P Consumer
A1-P Producer
1. GET .../policytypes/{policyTypeId}/policies
2. 200 OK (array(PolicyId))
A1-P Consumer
A1-P Producer

Figure 5.2.4.2.1-1 Query policy identifiers operation.

1) The A1-P Consumer shall send an HTTP GET request to the A1-P Producer. The target URI shall identify the resource "/policytypes/{policyTypeId}/policies". The message body shall be empty.
2) The A1-P Producer shall return the HTTP GET response. On success, "200 OK" shall be returned. The message body shall carry an array of policy identifiers representing all available policies of the given policy type. On failure, the appropriate error code shall be returned, and the response message body may contain additional error information.

On reception of a policyTypeId that does not exist, "404 Not Found" shall be returned.

5.2.4.2.2 Query policy identifiers procedure

The procedure to query policy identifiers is based on the Query policy identifiers operation illustrated in figure 5.2.4.2.1-1.

5.2.4.2.3 Query all policy identifiers procedure

The procedure to query all policy identifiers is based on the Query policy identifiers operation. The operation is performed for each policy type for which policies have been created, or for each policy type identifier discovered using the Query policy type identifiers operation defined in clause 5.2.3.2.1.

5.2.4.3 Create policy

5.2.4.3.1 General

The A1-P Consumer uses the Create policy procedure to create an A1 policy. The Create policy operation is used in the following procedures:

- Create single policy; and
- Create multiple policies.

The operation to create a policy is based on HTTP PUT. The policy to be created is identified with a URI that includes the policyTypeId and the policyId and the message body contains the PolicyObject.

If the policy creation request is accepted, the policy shall be enforced.

In case a policy already exists for the provided URI, the PUT request shall be handled as for Update single policy (see clause 5.2.4.4.1).

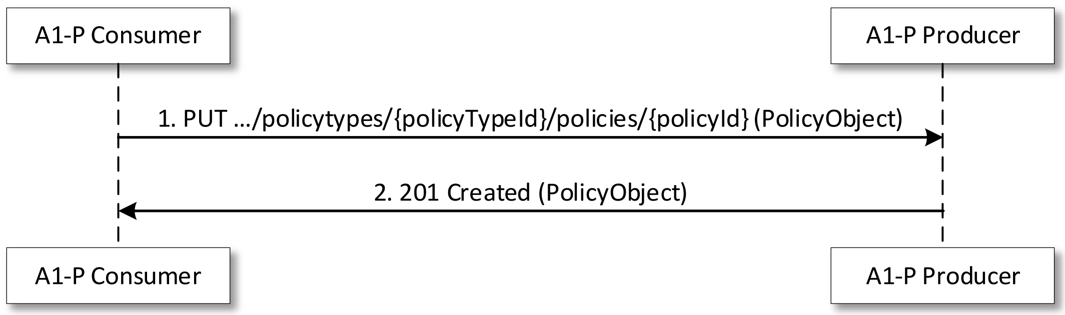

A1-P Consumer
A1-P Producer
1. PUT …/policytypes/{policyTypeId}/policies/{policyId} (PolicyObject)
2. 201 Created (PolicyObject)
A1-P Consumer
A1-P Producer

Figure 5.2.4.3.1-1 Create policy operation.

1) The A1-P Consumer shall generate the policyId and send an HTTP PUT request to the A1-P Producer. The target URI shall identify the resource (policyId) to be created under the resource "/policytypes/{policyTypeId}/policies". The message body shall carry a PolicyObject.
2) The A1-P Producer shall return the HTTP PUT response. On success, "201 Created" shall be returned. The "Location" HTTP header shall be present and shall carry the URI of the new policy and the message body shall carry the PolicyObject. On failure, the appropriate error code shall be returned, and the message body may contain additional error information.

When creating a policy, the A1-P Consumer shall include a policyTypeId in the URI for the PUT request. The policyTypeId shall be used by the A1-P Producer to select the appropriate schemas to use for validation of the PolicyObject and for PolicyStatus.

The A1-P Consumer may subscribe to policy status updates related to the created policy. Policy status updates are subscribed to by including the notificationDestination as a query parameter in the PUT request.

On reception of a policyTypeId that does not exist, "404 Not Found" shall be returned.

On failure to validate the PolicyObject, "400 Bad Request" shall be returned.

In case the new policy would be identical to, or would be overlapping or conflicting with, an existing policy, "409 Conflict" shall be returned.

5.2.4.3.2 Create single policy procedure

The procedure to create single policy is based on the Create policy operation illustrated in figure 5.2.4.3.1-1.

5.2.4.3.3 Create multiple policies procedure

The procedure to create multiple policies is a sequence of Create policy operations.

5.2.4.4 Update policy

5.2.4.4.1 General

The A1-P Consumer uses the Update single policy procedure to update an A1 policy. The Update policy operation is used in the following procedures:

- Update single policy; and
- Update multiple policies.

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

If the policy update request is accepted, the policy shall be enforced. In case a policy does not exist for the provided URI, the PUT request shall be handled as for Create single policy (see clause 5.2.4.3.1).

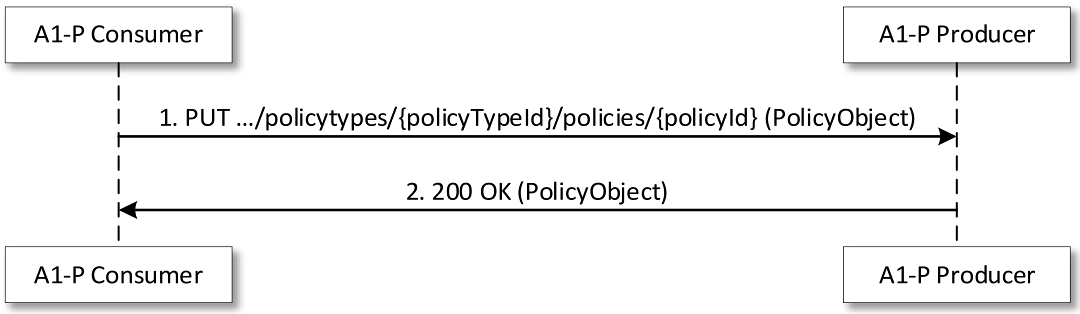


Figure 5.2.4.4.1-1 Update policy operation procedure.

1) The A1-P Consumer shall send an HTTP PUT request to the A1-P Producer. The target URI shall identify the policy to be updated based on the policyId under the resource "/policytypes/{policyTypeId}/policies". The message body shall contain a PolicyObject.
2) The A1-P Producer shall return the HTTP PUT response. On success, "200 OK" shall be returned. The message body shall carry a PolicyObject representing the updated policy. On failure, the appropriate error code shall be returned, and the response message body may contain additional error information.

The A1-P Consumer may subcribe to policy status updates related to the updated policy. Policy status updates are subscribed to by including the notificationDestination as a query parameter in the PUT request. The A1-P Consumer may change the notificationDestination for policy status updates in an update policy request. The A1-P Consumer may cancel policy status updates. Policy status updates are cancelled by omitting notificationDestination in the PUT request.

On failure to validate the PolicyObject fails, "400 Bad Request" shall be returned.

In case the policy after update would be identical to, or would be overlapping or conflicting with, an existing policy, "409 Conflict" shall be returned.

5.2.4.4.2 Update single policy procedure

The procedure to update single policy is based on the Update policy operation illustrated in figure 5.2.4.4.1-1.

5.2.4.4.3 Update multiple policies procedure

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

5.2.4.5 Query policy

5.2.4.5.1 General

The A1-P Consumer uses the Query policy operation to read an A1 policy. The Query policy operation is used in the following procedures:

- Query single policy;
- Query multiple policies; and
- Query all policies.

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

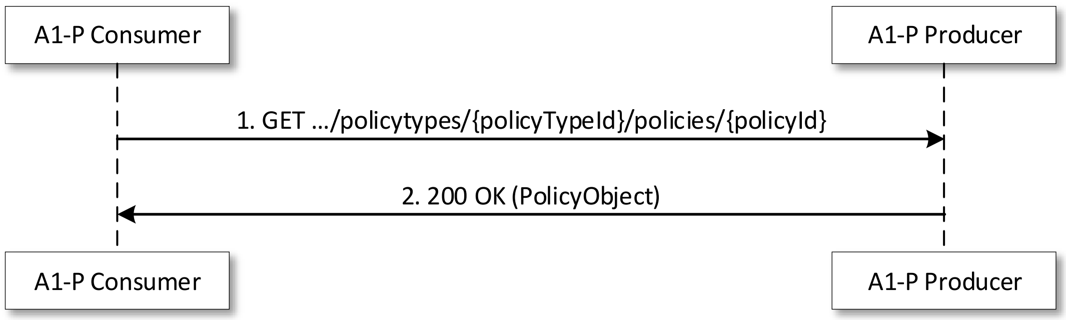


Figure 5.2.4.5.1-1 Query policy operation.

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

On reception of policyTypeId and policyId for which no policy exists, "404 Not Found" shall be returned.

5.2.4.5.2 Query single policy procedure

The procedure to query single policy is based on the Query policy operation illustrated in figure 5.2.4.5.1-1.

5.2.4.5.3 Query multiple policies procedure

The procedure to query multiple policies is a sequence of Query policy operations.

NOTE: To query all policies applicable to e.g., a dynamically defined group of UEs, a slice or a cell, the A1-P Consumer identifies applicable policyId(s) and makes a sequence of single policy queries.

5.2.4.5.4 Query all policies procedure

The procedure to query all policies is, for each policyTypeId retrieved as described in clause 5.2.3.2.1, a sequence of Query policy operations for each policyId retrieved as described in clause 5.2.4.2.1.

5.2.4.6 Delete policy

5.2.4.6.1 General

The A1-P Consumer uses the Delete policy procedure to delete an A1 policy. The Delete policy operation is used in the following procedures:

- Delete single policy; and
- Delete multiple policies.

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

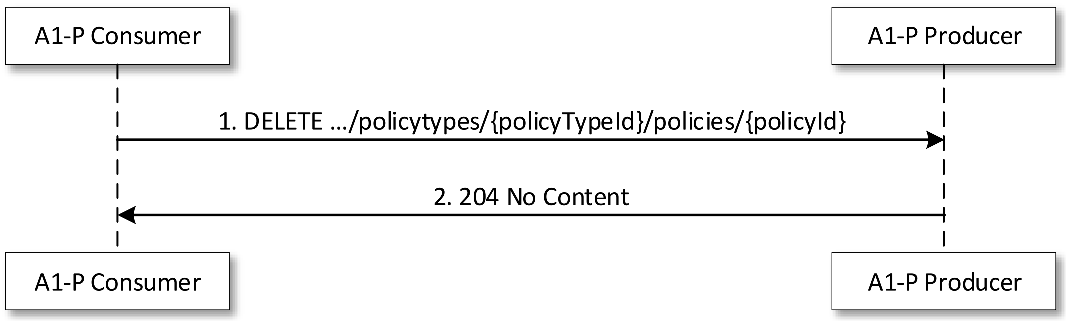


Figure 5.2.4.6.1-1 Delete policy operation.

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

On the reception of policyTypeId and policyId for which no policy exists, "404 Not Found" shall be returned.

5.2.4.6.2 Delete single policy procedure

The procedure to delete single policy is based on the Delete policy operation illustrated in figure 5.2.4.6.1-1.

5.2.4.6.3 Delete multiple policies procedure

The procedure to delete multiple policies is a sequence of Delete policy operations.

5.2.4.7 Query policy status

5.2.4.7.1 General

The A1-P Consumer uses the Query policy status operation to query the status of an A1 policy.

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

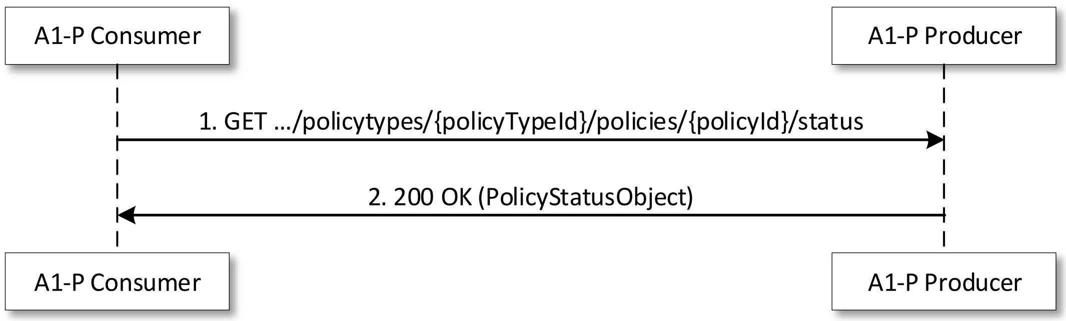

A1-P Consumer
A1-P Producer
1. GET .../policytypes/{policyTypeId}/policies/{policyId}/status
2. 200 OK (PolicyStatusObject)
A1-P Consumer
A1-P Producer

Figure 5.2.4.7.1-1 Query policy status operation.

1) The A1-P Consumer shall send an HTTP GET request to the A1-P Producer. The target URI shall identify the policy for which status is to be read based on the policyId under the resource "/policytypes/{policyTypeId}/policies". The message body shall be empty.

2) The A1-P Producer shall return the HTTP GET response. On success, "200 OK" shall be returned. The message body shall carry a PolicyStatusObject representing the status of the policy. On failure, the appropriate error code shall be returned, and the response message body may contain additional error information.

On reception of policyTypeId and policyId for which no policy status exists, "404 Not Found" shall be returned.

5.2.4.7.2 Query policy status procedure

The procedure to query policy status is based on the Query policy status operation illustrated in figure 5.2.4.7.1-1.

5.2.4.8 Notify policy status change

5.2.4.8.1 General

The A1-P Producer uses the Notify policy status change operation to update the A1-P Consumer about changes of the status of an A1 policy.

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

The A1-P Consumer uses the Create single policy operation defined in clause 5.2.4.3.1, or the Update single policy operation defined in clause 5.2.4.4.1, to subscribe to policy status updates for a policy.

The policy status change notifications are sent to the notifcationDestination provided when creating or updating the policy. The PolicyStatusObject contains the information about status changes and may contain information about causes.

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

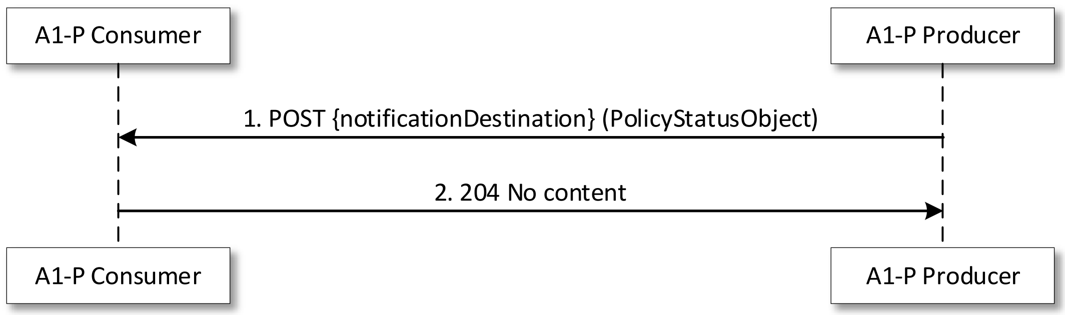


Figure 5.2.4.8.1-1 Feedback policy operation.

1) The A1-P Producer shall send an HTTP POST request to the A1-P Consumer. The target URI (notificationDestination) identifies the sink for policy notifications. The message body shall contain a PolicyStatusObject.

2) The A1-P Consumer shall return the HTTP POST response with "204 No Content". The message body shall be empty.

On failure to validate the PolicyStatusObject, "400 Bad Request" shall be returned, and the response message body may contain additional error information.

5.2.4.8.2 Feedback policy procedure

The procedure to feedback policy is based on the Notify policy status change operation illustrated in figure 5.2.4.8.1-1.

5.3 Enrichment information service

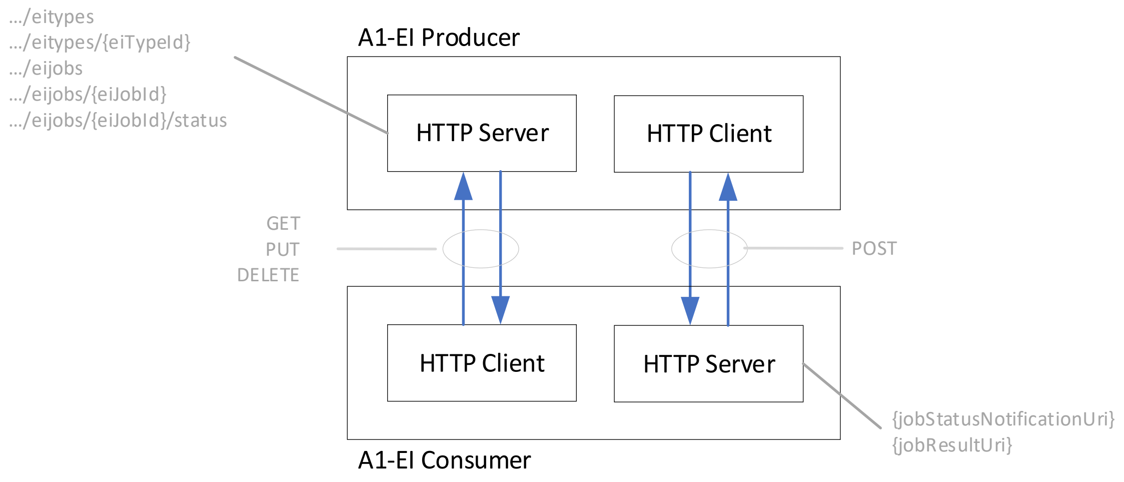
5.3.1 Introduction

The A1-EI service defines service operations that are used with EI types defined in A1TD [4].

5.3.2 Service description

5.3.2.1 Functional elements

The A1 application protocol for A1-EI is based on signaling between the A1-EI Consumer residing in the Near-RT RIC and the A1-EI Producer residing in the Non-RT RIC. Both the A1-EI Consumer and the A1-EI Producer contain an HTTP Client and an HTTP Server.



NOTE: Arrows indicate direction of HTTP requests sent from HTTP Client to HTTP Server and HTTP responses sent from HTTP Server to HTTP Client.

Figure 5.3.2.1-1 HTTP roles in service framework.

The present document specifies the A1 EI procedures defined in A1GAP [2] using HTTP operations in accordance with A1TP [3] where EI types, jobs and job results are represented as JSON objects in accordance with RFC 8259 [7] as defined in A1TD [4].

5.3.2.2 A1 EI representation

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

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

5.3.2.3 Representation objects

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

EiTypeObject

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

EiJobObject

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

EiJobStatusObject

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

EiJobResultObject

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

ProblemDetails

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

5.3.2.4 Resource identifiers

The URI for A1 enrichment information is:

…/eitypes

A single EI type is identified by adding the value of the EI type identifier to the URI:

…/eitypes/{eiTypeId}

The URI for A1 EI jobs is:

…/eijobs

A single EI job is identified by adding the value of the EI job identifier to the URI:

…/eijobs/{eiJobId}

The URI for status of an EI job is:

…/eijobs/{eiJobId}/status

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

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

5.3.3 Service operations for A1 EI types

5.3.3.1 Introduction

Table 5.3.3.1-1 describes the mapping between the A1 EI types operations, and the HTTP methods used to realise them, and the mandatory HTTP status codes.

Table 5.3.3.1-1 A1 EI operations to HTTP methods mapping.

Service operation	HTTP method	HTTP status codes
Query EI type identifiers	GET	200
Query EI type	GET	200, 404

The following clauses describe the EI types operations. For further information on the EI objects transferred in the HTTP message bodies, see A1TD [4].

The purpose of the EI types operations is to enable the A1-EI Consumer to

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

NOTE: The present document does not define any limits for how many eiTypeId/eiJobId that can be transferred in a single A1 message.

5.3.3.2 Query EI type identifiers

5.3.3.2.1 General

The A1-EI Consumer uses the Query EI type identifiers operation to query which EI types that are currently supported.

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

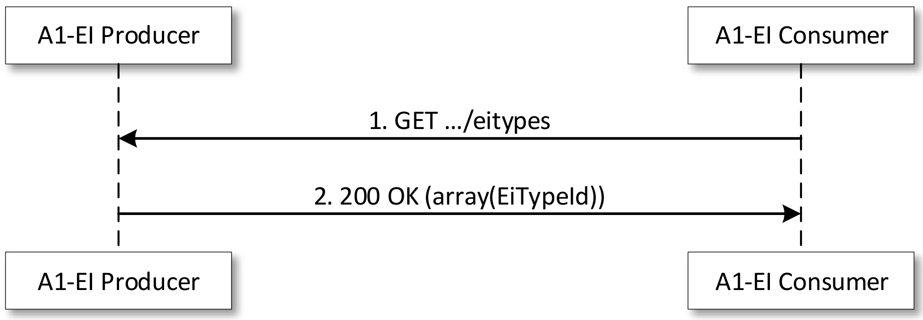


Figure 5.3.3.2.2-1 Query EI type identifiers operation.

1) The A1-EI Consumer shall send an HTTP GET request to the A1-EI Producer. The target URI shall identify the resource "/eitypes". The message body shall be empty.

2) The A1-EI Producer shall return the HTTP GET response. On success, "200 OK" shall be returned. The message body shall carry an array of EI type identifiers representing all available EI types. On failure, the appropriate error code shall be returned, and the response message body may contain additional error information.

5.3.3.2.2 Query EI type identifiers procedure

The procedure to query EI type identifiers is based on the Query EI type identifiers operation illustrated in figure 5.3.3.2.1-1.

5.3.3.3 Query EI type

5.3.3.3.1 General

The A1-EI Consumer uses the Query EI type operation to read the schemas for an EI type.

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

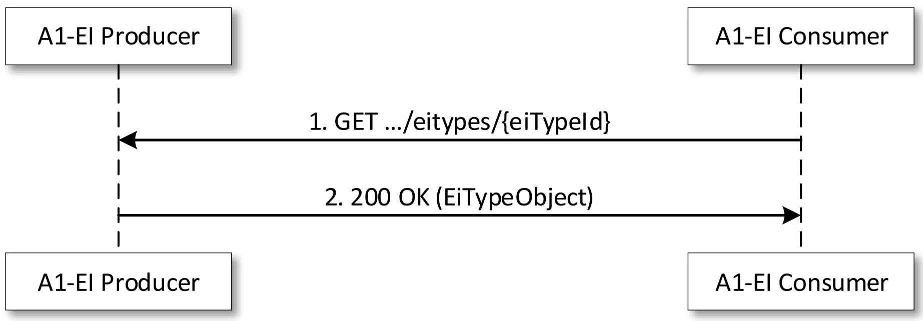


Figure 5.3.3.3.1-1 Query EI type operation.

1) The A1-EI Consumer shall send an HTTP GET request to the A1-EI Producer. The target URI shall identify the EI type to be read based on the eiTypeId under the resource "/eitypes". The message body shall be empty.

2) The A1-EI Producer shall return the HTTP GET response. On success, "200 OK" shall be returned. The message body shall carry an EITypeObject representing the read EI type. On failure, the appropriate error code shall be returned, and the response message body may contain additional error information.

On reception of an eiTypeId that does not exist, "404 Not Found" shall be returned.

5.3.3.3.2 Query EI type procedure

The procedure to query EI type is based on the Query EI type operation illustrated in figure 5.3.3.3.1-1.

5.3.4 Service operations for A1 EI jobs

5.3.4.1 Introduction

Table 5.3.4.1-1 describes the mapping between the A1 EI jobs operations and the HTTP methods used to realise them, and the mandatory HTTP status codes.

Table 5.3.4.1-1 A1 EI operations to HTTP methods mapping.

Service operation	HTTP method	HTTP status codes
Query EI job identifiers	GET	200
Create EI job	PUT	201, 400, 404, 409
Query EI job	GET	200, 404
Update EI job	PUT	200, 400, 409
Delete EI job	DELETE	204, 404
Query EI job status	GET	200, 404
Notify EI job status change	POST	204, 400

The following clauses describe the EI jobs operations. For further information on the EI job objects transferred in the HTTP message bodies, see A1TD [4].

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

The A1-EI Producer enables the A1-EI Consumer to create EI jobs for specific EI types andthe A1-EI Consumer can discover the supported EI types. The A1-EI Consumer indicates the eiTypeId in all EI job related operations.

NOTE: The present document does not define any limits for how many eiTypeIid/eiJobId that can be transferred in a single A1 message.

5.3.4.2 Query EI job identifiers

5.3.4.2.1 General

The A1-EI Consumer uses the Query EI job identifiers operation to check which EI jobs that exist.

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

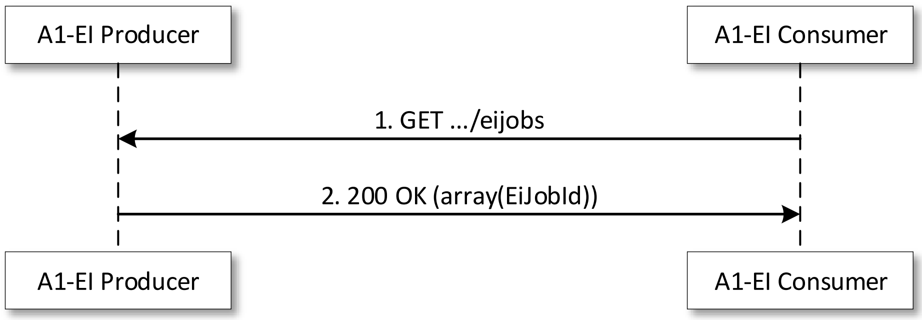


Figure 5.3.4.2.2-1 Query EI job identifiers operation.

1) The A1-EI Consumer shall send an HTTP GET request to the A1-EI Producer. The target URI shall identify the resource "/eijobs". The message body shall be empty.

2) The A1-EI Producer shall return the HTTP GET response. On success, "200 OK" shall be returned. The message body shall carry an array of EI job identifiers representing all available EI jobs of the given EI type, or of all EI types. On failure, the appropriate error code shall be returned, and the response message body may contain additional error information.

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

5.3.4.2.2 Query EI job identifiers procedure

The procedure to query EI job identifiers is based on the Query EI job identifiers operation illustrated in figure 5.3.4.2.2-1.

5.3.4.3 Create EI job

5.3.4.3.1 General

The A1-EI Consumer uses the Create EI job operation to create an EI job.

The operation to create an EI job is based on HTTP PUT. The EI job to be created is identified with a URI that includes the eiJobId and the message body contains the EiJobObject. The format of the EiJobObject is checked, and the request is either accepted or rejected. If accepted, delivery of EI results will start based on the content and conditions defined in the EI job.

NOTE: In case an EI job already exists for the provided URI, the PUT request is handled as for Update EI job (see clause 5.3.4.3.4).

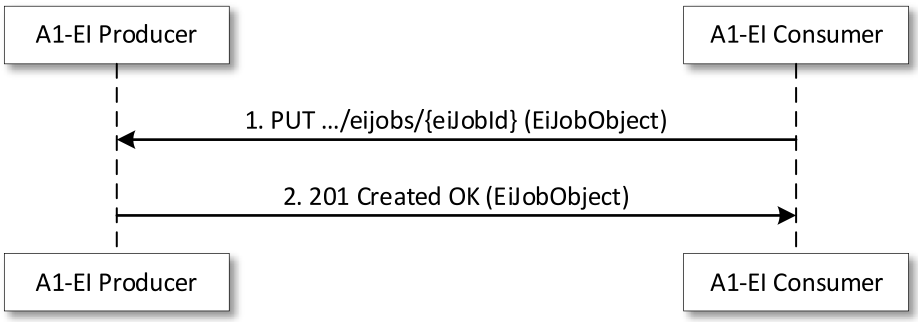

A1-EI Producer
A1-EI Consumer
1. PUT .../eijobs/{eiJobId} (EiJobObject)
2. 201 Created OK (EiJobObject)
A1-EI Producer
A1-EI Consumer

Figure 5.3.4.3.1-1 Create EI job operation.

1) The A1-EI Consumer shall generate the eiJobId and send an HTTP PUT request to the A1-EI Producer. The target URI shall identify the resource (eiJobId) be created under the resource "/eijobs". The message body shall carry an EiJobObject.

2) The A1-EI Producer shall return the HTTP PUT response. On success, "201 Created" shall be returned. The "Location" HTTP header shall be present and shall carry the URI of the new EI job and the message body shall carry the EiJobObject. On failure, the appropriate error code shall be returned, and the message body may contain additional error information.

The A1-EI Consumer may subscribe to EI job status updates related to the created EI job. EI job status updates are subscribed to by including the jobStatusNotificationUri in the EiJobObject.

On reception of an eiTypeId that does not exist, "404 Not Found" shall be returned.

On failure to validate the EiJobObject, "400 Bad Request" shall be returned.

In casethe new EI job would be identical to, or would be overlapping or conflicting with, an existing EI job, "409 Conflict" shall be returned.

5.3.4.3.2 Create EI job procedure

The procedure to create EI job is based on the Create EI job operation illustrated in figure 5.3.4.3.1-1.

5.3.4.4 Update EI job

5.3.4.4.1 General

The A1-EI Consumer uses the Update EI job operation to update an EI job.

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

NOTE: In case an EI job does not exist for the provided URI, the PUT request is handled as for Create EI job (see clause 5.3.4.3.2).

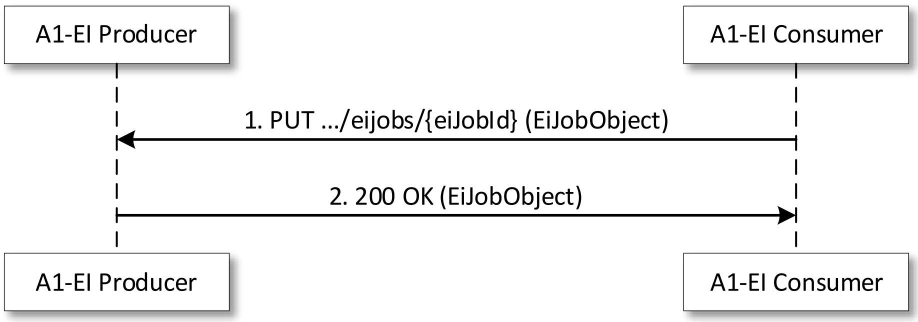


Figure 5.3.4.4.1-1 Update EI job operation.

1) The A1-EI Consumer shall send an HTTP PUT request to the A1-EI Producer. The target URI shall identiy the EI job to be updated based on the eiJobId under the resource "/eijobs". The message body shall contain an EiJobObject.

2) The A1-EI Producer shall return the HTTP PUT response. On success, "200 OK" shall be returned. The message body shall carry an EiJobObject representing the updated EI job. On failure, the appropriate error code shall be returned, and the response message body may contain additional error information.

The A1-EI Consumer may subscribe to EI job status updates related to the updated EI job. EI job status updates are subscribed to by including the jobStatusNotificationUri in the EiJobObject. The A1-EI Consumer may change the jobStatusNotificationUri for EI job status updates in an update ei job request. The A1-EI Consumer may cancel EI job status updates. EI job status updates are cancelled by omitting jobStatusNotificationUri in the EiJobObject.

On failure to validatei the EiJobObject, "400 Bad Request" shall be returned.

In case the EI job after update would be identical to, or would be overlapping or conflicting with, an existing EI job, "409 Conflict" shall be returned.

5.3.4.4.2 Update EI job procedure

The procedure to update EI job is based on the Update EI job operation illustrated in figure 5.3.4.4.1-1.

5.3.4.5 Query EI job

5.3.4.5.1 General

The A1-EI Consumer uses the Query EI job operation to read an EI job.

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

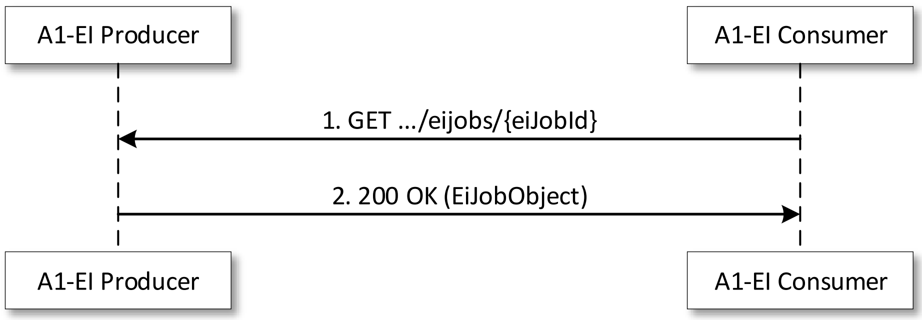


Figure 5.3.4.5.1-1 Query EI job operation.

1) The A1-EI Consumer shall send an HTTP GET request to the A1-EI Producer. The target URI shall identify the EI job to be read based on the eiJobId under the resource "/eijobs". The message body shall be empty.

2) The A1-EI Producer shall return the HTTP GET response. On success, "200 OK" shall be returned. The message body shall carry an EiJobObject representing the read EI job. On failure, the appropriate error code shall be returned, and the response message body may contain additional error information.

On reception of eiJobId for which no EI job exists, "404 Not Found" shall be returned.

5.3.4.5.2 Query EI job procedure

The procedure to query EI job is based on the Query EI job operation illustrated in figure 5.3.4.5.1-1.

5.3.4.6 Delete EI job

5.3.4.6.1 General

The A1-EI Consumer uses the Delete EI job operation to delete an EI job.

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

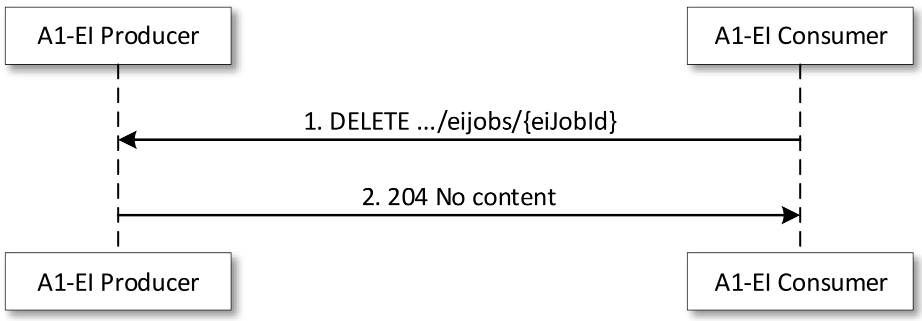


Figure 5.3.4.6.1-1 Delete EI job operation.

1) The A1-EI Consumer shall send an HTTP DELETE request to the A1-EI Producer. The target URI shall identify the EI job to be deleted based on the eiJobId under the resource "/eijobs". The message body shall be empty.

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

On reception of eiJobId for which no EI job exists, "404 Not Found" shall be returned.

5.3.4.6.2 Delete EI job procedure

The procedure to delete EI job is based on the Delete EI job operation illustrated in figure 5.3.4.6.1-1.

5.3.4.7 Query EI job status

5.3.4.7.1 General

The A1-EI Consumer uses the Query EI job status operation to query the status of an EI job.

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

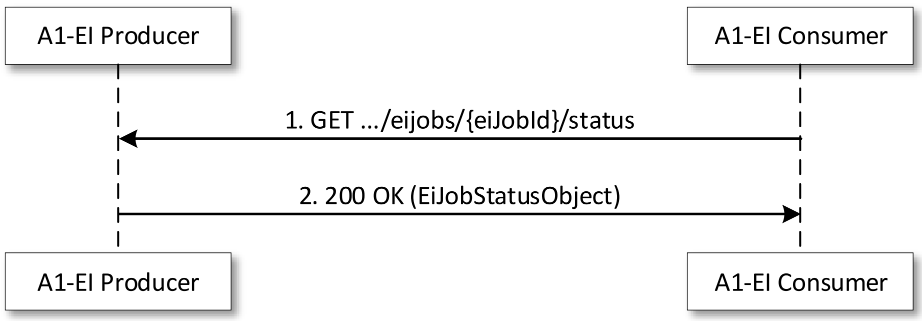

A1-EI Producer
A1-EI Consumer
1. GET .../eijobs/{eiJobId}/status
2. 200 OK (EiJobStatusObject)
A1-EI Producer
A1-EI Consumer

Figure 5.3.4.7.1-1 Query EI job status operation.

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

2) The A1-EI Producer shall return the HTTP GET response. On success, "200 OK" shall be returned. The message body shall carry an EI job status object representing the status of the EI job. On failure, the

appropriate error code shall be returned, and the response message body may contain additional error information.

On reception of eiJobId for which no EI job status exists, "404 Not Found" shall be returned.

5.3.4.7.2 Query EI job status procedure

The procedure to query EI job status is based on the Query EI job status operation illustrated in figure 5.3.4.7.1-1.

5.3.4.8 Notify EI job status change

5.3.4.8.1 General

The A1-EI Consumer uses the Notify EI job status change operation to notify status change of an EI job.

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

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

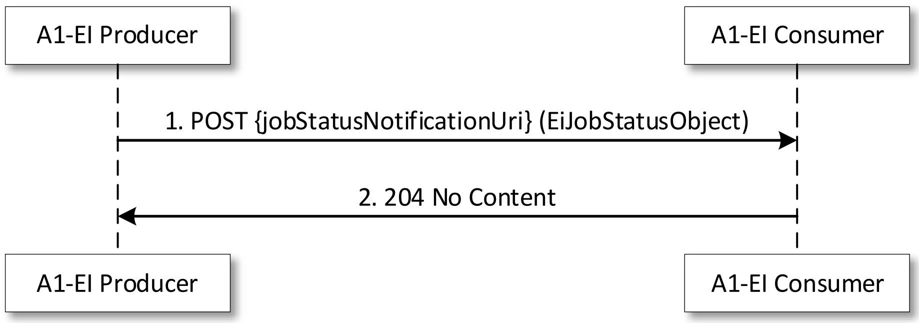


Figure 5.3.4.8.1-1 Notify EI job status change operation.

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

2) The A1-EI Consumer shall return the HTTP POST response with "204 No Content". The response message body shall be empty.

On failure to validate the EiJobStatusObject fails, "400 Bad Request" shall be returned, and the response message body may contain additional error information.

5.3.4.8.2 Notify EI job status change procedure

The procedure to notify EI job status change is based on the Notify EI job status change operation illustrated in figure 5.3.4.8.1-1.

5.3.5 Service operations for A1 EI job result

5.3.5.1 Introduction

Table 5.3.5.1-1 describes the mapping between the A1 EI job result operations, and the HTTP methods used to realise them, and the mandatory HTTP status codes.

Table 5.3.5.1-1 A1 EI operations to HTTP methods mapping.

Service operation	HTTP method	HTTP status codes
Deliver EI job result	POST	204, 400

The following clauses describe the A1 EI job result operations. For further information on the EI job result objects transferred in the HTTP message bodies, see A1TD [4].

The purpose of the A1EI job result operations is to enable the A1-EI Producer to deliver EI job results according to the service description agreed during job creation. The URL to which the EI job result is delivered is transferred from the A1-EI consumer in the EI job object.

5.3.5.2 Deliver EI job result

5.3.5.2.1 General

The A1-EI Consumer uses the Deliver EI job result operation to deliver EI job results using push-based method.

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

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

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

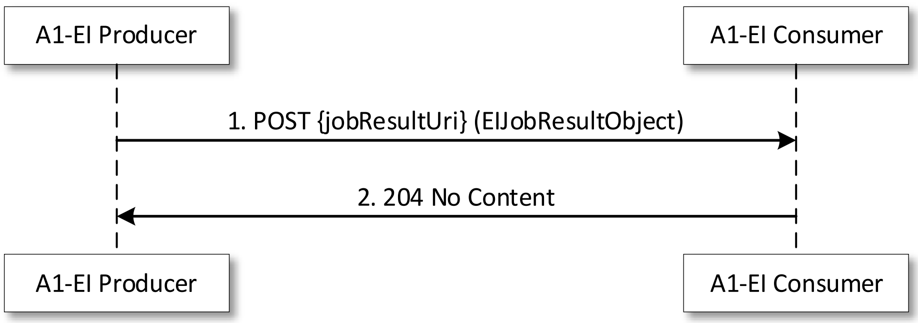


Figure 5.3.5.2.1-1 Deliver EI job result operation.

1) The A1-EI Producer shall send an HTTP POST request to the A1-EI Consumer. The target URI (jobResultUri) identifies the sink for EI job result delivieries. The message body shall contain an EI job result object.

2) The A1-EI Consumer shall return the HTTP POST response with "204 No Content". The response message body shall be empty.

On failure to validate the EiJobResultObject fails, "400 Bad Request" shall be returned, and the response message body may contain additional error information.

5.3.5.2.2 Deliver EI job result procedure

The procedure to deliver EI job result is based on the Deliver EI job result operation illustrated in figure 5.3.5.2.1-1.

5.4 ML model management service

No explicit ML model service operations are defined in the present document.

6 API Definitions

6.1 Introduction

6.1.1 Encoding of attributes in A1 data types

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

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

6.1.2 Compatibility of API versions for A1 services

The API version and API name for each of the A1 services are defined in the following chapters. The API version is a single digit that corresponds to the major version of the corresponding OpenAPI document in Annex A. Based on the versioning rules for the OpenAPI documents defined in the OpenAPI Specification [12], this implies that implementations of an A1 service in the Non/Near-RT RICs are

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

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

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

6.2 A1-P (policy management)

6.2.1 Introduction

This section contains the definition of the REST based API for the A1 policy management service referred to as A1-P.

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

Based on the URI structure defined in clause 4.4.1 of 3GPP TS 29.501 [6], the request URI used in HTTP request from the A1-P consumer towards the A1-P producer shall have the following structure:

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

where the "ResourceUriPart" shall be as be defined in clause 6.2.3.

6.2.2 Usage of HTTP

6.2.2.1 General

The A1 Transport, HTTP protocol and security requirements, is described in A1TP [3].

6.2.2.2 HTTP standard headers

Encodings and applicable MIME media type for the related Content-Type header are not specified in this version of the present document.

6.2.2.3 HTTP custom headers

No HTTP custom headers are specified in this version of the present document.

6.2.3 Resources

6.2.3.1 Overview

6.2.3.1.1 Resource URI structure

The resource URI structure for the A1-P API is illustrated in figure 6.2.3.1.1-1.

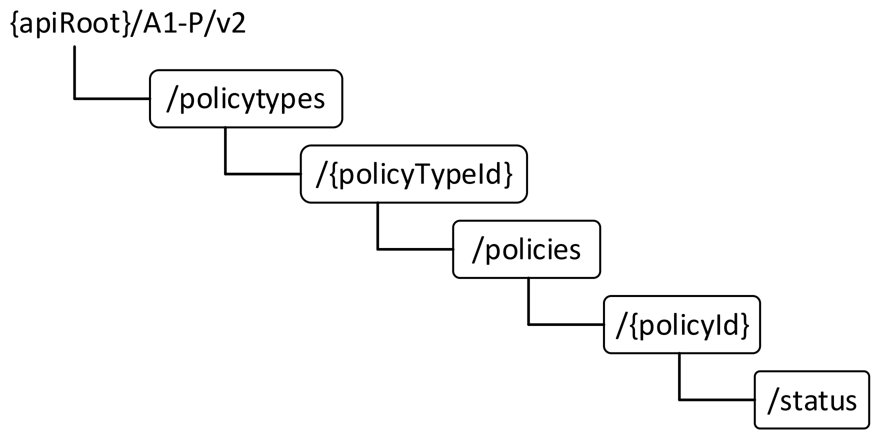


Figure 6.2.3.1.1-1: Resource URI structure of the A1-P API

6.2.3.1.2 Resources and methods

Table 6.2.3.1.2-1 provides an overview of the resources and HTTP methods defined for the A1-P API.

Table 6.2.3.1.2-1: 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 single policy type
Individual Policy Object	/policytypes/{policyTypeId}/policies/{policyId}	PUT	Create single policy, Update single policy
		GET	Query single policy
		DELETE	Delete single 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
Notify Policy Status	{notificationDestination}	POST	Feedback policy

For each combination of a resource and an HTTP method in Table 6.2.3.1.2-1, the HTTP status codes are as defined for the A1 policy procedures listed in the Description column and defined in clause 5.2.3. For any other combination of a resource defined for this API and an HTTP method, including those HTTP methods that are not defined for this API, the HTTP status code 405 (Method Not Allowed) shall be used to indicate that the method is not supported on the resource.

6.2.3.1.3 Policy type identifier

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

```
typename_version
```

where

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

`version` is the version of the policy type defined as major.minor.patch as described in SemVer [8].

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

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

6.2.3.2 Individual Policy Object

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

6.2.3.2.1 Description

The resource represents an A1 policy.

6.2.3.2.2 Resource Definition

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

6.2.3.2.3 Resource Standard Methods

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

6.2.3.2.3.1 HTTP PUT

This method shall support the request data structures specified in table 6.2.3.2.3.1-1 and the response data structures and response codes specified in table 6.2.3.2.3.1-2.

Table 6.2.3.2.3.1-1: Data structures supported by the HTTP PUT Request Body on this resource

Data type	P	Cardinality	Description
PolicyObject	M	1	Create policy

Table 6.2.3.2.3.1-2: Data structures supported by the HTTP PUT Response Body on this resource

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

This method shall support the URI query parameters specified in table 6.2.3.2.3.1-3.

Table 6.2.3.2.3.1-3: 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

6.2.3.2.3.2 HTTP GET

This method shall support the request data structures specified in table 6.2.3.2.3.2-1 and the response data structures and response codes specified in table 6.2.3.2.3.2-2.

Table 6.2.3.2.3.2-1: Data structures supported by the HTTP GET Request Body on this resource

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

Table 6.2.3.2.3.2-2: 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

6.2.3.2.3.3 HTTP DELETE

This method shall support the request data structures specified in table 6.2.3.2.3.3-1 and the response data structures and response codes specified in table 6.2.3.2.3.3-2.

Table 6.2.3.2.3.3-1: Data structures supported by the HTTP DELETE Request Body on this resource

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

Table 6.2.3.2.3.3-2: 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

6.2.3.2.3.4 HTTP POST

This method is not supported on the resource.

6.2.3.2.4 Resource Custom Operations

No custom operations are defined.

6.2.3.3 Individual Policy Status Object

6.2.3.3.1 Description

The resource represents the status of an A1 policy.

6.2.3.3.2 Resource Definition

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

6.2.3.3.3 Resource Standard Methods

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

6.2.3.3.3.1 HTTP PUT

Method is not supported on this resource.

6.2.3.3.3.2 HTTP GET

This method shall support the request data structures specified in table 6.2.3.3.3.2-1 and the response data structures and response codes specified in table 6.2.3.3.3.2-2.

Table 6.2.3.3.3.2-1: Data structures supported by the HTTP GET Request Body on this resource

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

Table 6.2.3.3.3.2-2: 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

6.2.3.3.3.3 HTTP DELETE

Method is not supported on this resource.

6.2.3.3.3.4 HTTP POST

Method is not supported on this resource.

6.2.3.3.4 Resource Custom Operations

No custom operations are defined.

6.2.3.4 All Policy Identifiers

6.2.3.4.1 Description

The resource represents A1 policy identifiers.

6.2.3.4.2 Resource Definition

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

6.2.3.4.3 Resource Standard Methods

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

6.2.3.4.3.1 HTTP PUT

Method is not supported on this resource.

6.2.3.4.3.2 HTTP GET

This method shall support the request data structures specified in table 6.2.3.6.3.2-1 and the response data structures and response codes specified in table 6.2.3.6.3.2-2.

Table 6.2.3.4.3.2-1: Data structures supported by the HTTP GET Request Body on this resource

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

Table 6.2.3.4.3.2-2: 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

6.2.3.4.3.3 HTTP DELETE

Method is not supported on this resource.

6.2.3.4.3.4 HTTP POST

Method is not supported on this resource.

6.2.3.4.4 Resource Custom Operations

No custom operations are defined.

6.2.3.5 All Policy Type Identifiers

6.2.3.5.1 Description

The resource represents A1 policy type identifiers.

6.2.3.5.2 Resource Definition

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

6.2.3.5.3 Resource Standard Methods

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

6.2.3.5.3.1 HTTP PUT

Method is not supported on this resource.

6.2.3.5.3.2 HTTP GET

This method shall support the request data structures specified in table 6.2.3.5.3.2-1 and the response data structures and response codes specified in table 6.2.3.5.3.2-2.

Table 6.2.3.5.3.2-1: Data structures supported by the HTTP GET Request Body on this resource

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

Table 6.2.3.5.3.2-2: 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

6.2.3.5.3.3 HTTP DELETE

Method is not supported on this resource.

6.2.3.5.3.4 HTTP POST

Method is not supported on this resource.

6.2.3.5.4 Resource Custom Operations

No custom operations are defined.

6.2.3.6 Individual Policy Type Object

6.2.3.6.1 Description

The resource represents an A1 policy type.

6.2.3.6.2 Resource Definition

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

6.2.3.6.3 Resource Standard Methods

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

6.2.3.6.3.1 HTTP PUT

Method is not supported on this resource.

6.2.3.6.3.2 HTTP GET

This method shall support the request data structures specified in table 6.2.3.6.3.2-1 and the response data structures and response codes specified in table 6.2.3.6.3.2-2.

Table 6.2.3.6.3.2-1: Data structures supported by the HTTP GET Request Body on this resource

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

Table 6.2.3.6.3.2-2: 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

6.2.3.6.3.4 HTTP DELETE

This method is not supported on the resource.

6.2.3.6.3.5 HTTP POST

This method is not supported on the resource.

6.2.3.6.4 Resource Custom Operations

No custom operations are defined.

6.2.4 Custom Operations without associated resources

No custom operations are defined.

6.2.5 Notifications

6.2.5.1 Notify Policy Status

6.2.5.1.1 Description

The resource represents the destination for policy status notifications.

6.2.5.1.2 Resource Definition

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

6.2.5.1.3 Resource Standard Methods

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

6.2.5.1.3.1 HTTP PUT

Method is not supported on this resource.

6.2.5.1.3.2 HTTP GET

Method is not supported on this resource.

6.2.5.1.3.3 HTTP DELETE

Method is not supported on this resource.

6.2.5.1.3.4 HTTP POST

This method shall support the request data structures specified in table 6.2.5.1.3.4-1 and the response data structures and response codes specified in table 6.2.5.1.3.4-2.

Table 6.2.5.1.3.4-1: Data structures supported by the HTTP POST Request Body on this resource

Data type	P	Cardinality	Description
PolicyStatusObject	M	1	Notify policy

Table 6.2.5.1.3.4-2: 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

6.2.6 Data Model

6.2.6.1 Introduction

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

The data model for the data types transported in the A1-P procedures is defined in A1TD [4].

6.2.6.2 Simple data types and enumerations

6.2.6.2.1 Simple data types

The resource identifiers defined in clause 5.2.2.4 include policy type identifier and policy identifier based on the simple data types specified in table 6.2.6.2.1-1.

Table 6.2.6.2.1-1: 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 (A1TD [4])	used in URI
PolicyId	string	policy identifier assigned by the A1-P Consumer when a policy is created	used in URI

6.2.6.3 Structured data types

6.2.6.3.1 Problem details

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

The ProblemDetails statement specified in table 6.2.6.3-1 contains attributes defined in IETF RFC 7807 [10]:

Table 6.2.6.3.1-1: 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 [9] 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	

6.2.7 Error Handling

6.2.7.1 General

HTTP error handling shall be supported as specified in clause 5.2.4 of 3GPP TS 29.500 [11] and according to the principles in 3GPP TS 29.501 [6].

6.2.7.2 Protocol Errors

No protocol errors are described in the present document.

6.2.7.3 Application Errors

The application errors defined for the A1-P service are listed in table 6.2.7.3-1.

Table 6.2.7.3-1: Application errors

Application Error	HTTP status code	Description
Bad Request	400	Used when the Near-RT RIC or the Non-RTR RIC cannot or will not process a request, e.g., when the validation of PolicyObject towards a policy type schema, or the validation of PolicyStatusObject towards a policy status schema, fails.
Not Found	404	Used when the Near-RT RIC did not find a current representation for the resource representing a policy type or a policy, e.g., for a policy type that is not available or a policy that does not exist.
Method Not Allowed	405	Used when the HTTP method is not supported by the resource defined for the A1-P API.
Conflict	409	Used if the Near-RT RIC detects that a policy requested to be created or updated may be overlapping or conflicting with a policy that exists, e.g., if the policy in the request is identical to an existing policy.

Application errors should be mapped to the most applicable 4xx/5xx HTTP error status code. If no such status code is applicable, one of the status codes 400 (Bad Request) or 500 (Internal Server Error) should be used.

The HTTP status codes listed in table 6.2.7.3-1 shall be used as defined in clause 5.2.3 for the A1-P procedures and clause 6.2.3 for the resources.

Implementations may use additional HTTP error status codes in addition to those listed in table 6.2.7.3-1, as long as they are valid HTTP status codes.

A list of all valid HTTP status codes and their specification documents can be obtained from the HTTP status code registry [13].

In addition, the response body may contain a JSON representation of a "ProblemDetails" data structure in the payload body as defined in clause 6.2.6.2.1. In that case, as defined by IETF RFC 7807 [10], the "Content-Type" HTTP header shall be set to "application/problem+json".

6.3 A1-EI (enrichment information)

6.3.1 Introduction

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

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

Based on the URI structure defined in clause 4.4.1 of 3GPP TS 29.501 [6], the request URI used in HTTP request from the A1-EI consumer towards the A1-EI producer shall have the following structure:

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

where the "ResourceUriPart" shall be as be defined in clause 6.3.3.

6.3.2 Usage of HTTP

6.3.2.1 General

The A1 Transport, HTTP protocol and security requirements, is described in A1TP [3].

6.3.2.2 HTTP standard headers

Encodings and applicable MIME media type for the related Content-Type header are not specified in this version of the present document.

6.3.2.3 HTTP custom headers

No HTTP custom headers are specified in the present document.

6.3.3 Resources

6.3.3.1 Overview

6.3.3.1.1 Resource URI structure

The resource URI structure for the A1-EI API is illustrated in figure 6.3.3.1.1-1.

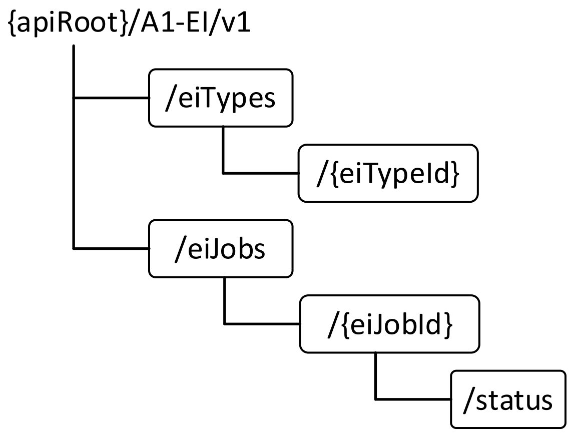


Figure 6.3.3.1.1-1: Resource URI structure of the A1-EI API

6.3.3.1.2 Resources and methods

Table 6.3.3.1.2-1 provides an overview of the resources and HTTP methods defined for the A1-EI API.

Table 6.3.3.1.2-1: 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

For each combination of a resource and an HTTP method in table 6.3.3.1.2-1, the HTTP status codes are as defined for the A1-EI procedures listed in the Description column and defined in clauses 5.3.3-5.3.5. For any other combination of a resource defined for this API and an HTTP method, including those HTTP methods that are not defined for this API, the HTTP status code 405 (Method Not Allowed) shall be used to indicate that the method is not supported on the resource.

6.3.3.1.3 EI type identifier

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

```
typename_version
```

where

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

`version` is the version of the EI type defined as major.minor.patch as described in SemVer [8].

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

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

6.3.3.1.4 EI job identifier

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

6.3.3.2 All EI Type Identifiers

6.3.3.2.1 Description

The resource represents EI type identifiers.

6.3.3.2.2 Resource Definition

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

6.3.3.2.3 Resource Standard Methods

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

6.3.3.2.3.1 HTTP GET

This method shall support the request data structures specified in table 6.3.3.2.3.1-1 and the response data structures and response codes specified in table 6.3.3.2.3.1-2.

Table 6.3.3.2.3.1-1: Data structures supported by the HTTP GET Request Body on this resource

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

Table 6.3.3.2.3.1-2: 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

6.3.3.2.4 Resource Custom Operations

No custom operations are defined.

6.3.3.3 Individual EI Type

6.3.3.3.1 Description

The resource represents an EI type.

6.3.3.3.2 Resource Definition

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

6.3.3.3.3 Resource Standard Methods

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

6.3.3.3.3.1 HTTP GET

This method shall support the request data structures specified in table 6.3.3.3.3.1-1 and the response data structures and response codes specified in table 6.3.3.3.3.1-2.

Table 6.3.3.3.3.1-1: Data structures supported by the HTTP GET Request Body on this resource

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

Table 6.3.3.3.3.1-2: 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

6.3.3.3.4 Resource Custom Operations

No custom operations are defined.

6.3.3.4 All EI Jobs

6.3.3.4.1 Description

The resource represents EI job identifiers.

6.3.3.4.2 Resource Definition

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

6.3.3.4.3 Resource Standard Methods

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

6.3.3.4.3.1 HTTP GET

This method shall support the request data structures specified in table 6.3.3.4.3.1-1 and the response data structures and response codes specified in table 6.3.3.4.3.1-2.

Table 6.3.3.4.3.1-1: Data structures supported by the HTTP GET Request Body on this resource

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

Table 6.3.3.4.3.1-2: Data structures supported by the HTTP GET Response Body on this resource

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

This method shall support the URI query parameters specified in table 6.3.3.4.3.1-3.

Table 6.3.3.4.3.1-3: 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

6.3.3.5 Individual EI Job

6.3.3.5.1 Description

The resource represents an EI job.

6.3.3.5.2 Resource Definition

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

6.3.3.5.3 Resource Standard Methods

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

6.3.3.5.3.1 HTTP PUT

This method shall support the request data structures specified in table 6.3.3.5.3.1-1 and the response data structures and response codes specified in table 6.3.3.5.3.1-2.

Table 6.3.3.5.3.1-1: Data structures supported by the HTTP PUT Request Body on this resource

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

Table 6.3.3.5.3.1-2: 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

6.3.3.5.3.2 HTTP GET

This method shall support the request data structures specified in table 6.3.3.5.3.2-1 and the response data structures and response codes specified in table 6.3.3.5.3.2-2.

Table 6.3.3.5.3.2-1: Data structures supported by the HTTP GET Request Body on this resource

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

Table 6.3.3.5.3.2-2: 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

6.3.3.5.3.3 HTTP DELETE

This method shall support the request data structures specified in table 6.3.3.5.3.3-1 and the response data structures and response codes specified in table 6.3.3.5.3.3-2.

Table 6.3.3.5.3.3-1: Data structures supported by the HTTP DELETE Request Body on this resource

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

Table 6.3.3.5.3.3-2: 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

6.3.3.5.4 Resource Custom Operations

No custom operations are defined.

6.3.3.6 Individual EI Job Status

6.3.3.6.1 Description

The resource represents the status of an EI job.

6.3.3.6.2 Resource Definition

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

6.3.3.6.3 Resource Standard Methods

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

6.3.3.6.3.1 HTTP GET

This method shall support the request data structures specified in table 6.3.3.6.3.1-1 and the response data structures and response codes specified in table 6.3.3.6.3.1-2.

Table 6.3.3.6.3.1-1: Data structures supported by the HTTP GET Request Body on this resource

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

Table 6.3.3.6.3.1-2: 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

6.2.3.3.4 Resource Custom Operations

No custom operations are defined.

6.3.4 Custom Operations without associated resources

No custom operations are defined.

6.3.5 Notifications

6.3.5.1 Notify EI Job Status

6.3.5.1.1 Description

The resource represents the destination for EI job status notifications.

6.3.5.1.2 Resource Definition

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

6.3.5.1.3 Resource Standard Methods

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

6.3.5.1.3.1 HTTP POST

This method shall support the request data structures specified in table 6.3.5.1.3.1-1 and the response data structures and response codes specified in table 6.3.5.1.3.1-2.

Table 6.3.5.1.3.1-1: Data structures supported by the HTTP POST Request Body on this resource

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

Table 6.3.5.1.3.1-2: 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

6.3.6 Delivery

6.3.6.1 Deliver EI job result

6.3.6.1.1 Description

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

6.3.6.1.2 Resource Definition

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

6.3.6.1.3 Resource Standard Methods

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

6.3.6.1.3.1 HTTP POST

This method shall support the request data structures specified in table 6.3.6.1.3.1-1 and the response data structures and response codes specified in table 6.3.6.1.3.1-2.

Table 6.3.6.1.3.1-1: Data structures supported by the HTTP POST Request Body on this resource

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

Table 6.3.6.1.3.1-2: 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

6.3.7 Data model

6.3.7.1 Introduction

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

The data model for the data types transported in the A1-EI procedures is defined in A1TD [4].

6.3.7.2 Simple data types and enumerations

6.3.7.2.1 Simple data types

The resource identifiers defined in clause 5.3.2.4 include EI type identifier or EI job identifier based on the simple data types specified in table 6.3.7.2.1-1.

Table 6.3.7.2.1-1: General definition of simple data types for URI identifiers

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

Table 6.3.7.2.1-2: General definition of simple data types for callback URIs

Callback URI	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

6.3.7.3 Structured data types

6.3.7.3.1 Problem details

The problem details statement is the same as defined for A1-P, see chapter 6.2.6.3.1.

6.3.8 Error handling

6.3.8.1 General

HTTP error handling shall be supported as specified in clause 5.2.4 of 3GPP TS 29.500 [11] and according to the principles in 3GPP TS 29.501 [6].

6.3.8.2 Protocol Errors

No protocol errors are described in this version of the present document.

6.3.8.3 Application Errors

The application errors defined for the A1-EI service are listed in table 6.3.8.3-1.

Table 6.3.8.3-1: Application errors

Application Error	HTTP status code	Description
Bad Request	400	Used when the Near-RT RIC or the Non-RTR RIC cannot or will not process a request, e.g., when the validation of an object towards a schema, fails.
Not Found	404	Used when the Non-RT RIC did not find a current representation for the resource representing an EI type or an EI job.
Method Not Allowed	405	Used when the HTTP method is not supported by the resource defined for the A1-EI API.
Conflict	409	Used if the Non-RT RIC detects that an EI job requested to be created or updated may be overlapping or conflicting with an EI job that exists.

Application errors should be mapped to the most applicable 4xx/5xx HTTP error status code. If no such status code is applicable, one of the status codes 400 (Bad Request) or 500 (Internal Server Error) should be used.

The HTTP status codes listed in table 6.2.8.3-1 shall be used as defined in clauses 5.3.3-5.3.5 for the A1-EI procedures and clause 5.3.3 for the resources.

Implementations may use additional HTTP error status codes in addition to those listed in table 6.2.8.3-1, as long as they are valid HTTP status codes.

A list of all valid HTTP status codes and their specification documents can be tained from the HTTP status code registry [13].

In addition, the response body may contain a JSON representation of a "ProblemDetails" data structure in the payload body as defined in clause 6.2.6.2.1. In that case, as defined by IETF RFC 7807 [10], the "Content-Type" HTTP header shall be set to "application/problem+json".

Annex A (normative): OpenAPI specification

A.1 General

This Annex specifies the formal definition of the A1 API(s). It consists of OpenAPI documents in YAML format that are based on the OpenAPI 3.0.1 Specification [12].

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

A.1.1 Versioning of A1 OpenAPI documents

The OpenAPI documents for the A1 services found in this chapter are versioned as specified in SemVer [8] as described in the OpenAPI Specification [12]. When included in the present document, the OpenAPI documents are considered as released and are versioned using three-digit major.minor.patch where the main compatibility expectations stated in SemVer [8] implies:

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

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

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

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

A.1.2 Current API versions

The present document defines the API versions indicated in table A.1.2-1.

Table A.1.2-1

API name	API version	OpenAPI version
A1-P	v2	2.2.0
A1-EI	v1	1.3.0

NOTE: API names and API versions are defined in clauses 6.2.1 and 6.3.1 and OpenAPI versions are defined by the OpenAPI documents in the following clauses.

A.2 Policy management API

```
openapi: 3.0.1
info:
  title: 'A1-P Policy Management Service'
  version: 2.2.0
  description: |
    API for Policy Management Service.
    © 2022, O-RAN ALLIANCE.
    All rights reserved.
externalDocs:
  description: 'O-RAN.WG2.A1AP-v04.00 A1 interface: Application Protocol'
  url: 'https://www.o-ran.org/specifications'
servers:
  - url: '{apiRoot}/A1-P/v2'
    variables:
      apiRoot:
        default: 'https://example.com'
        description: 'apiRoot as defined in clause 6.2.1 in O-RAN.WG2.A1AP'
paths:
  '/policytypes':
    get:
      description: 'Get all policy type identifiers'
      tags:
      - All Policy Type Identifiers
      responses:
        200:
          description: 'Array of all policy type identifiers'
          content:
            application/json:
              schema:
                type: array
                items:
                  "$ref": "#/components/schemas/PolicyTypeId"
                minItems: 0

  '/policytypes/{policyTypeId}':
    parameters:
      - name: policyTypeId
```

```
        in: path
        required: true
        schema:
          "$ref": "#/components/schemas/PolicyTypeId"
    get:
      description: 'Get the schemas for a policy type'
      tags:
      - Individual Policy Type
      responses:
        200:
          description: 'The policy type schemas'
          content:
            application/json:
              schema:
                "$ref": "#/components/schemas/PolicyTypeObject"
        404:
          "$ref": "#/components/responses/404-NotFound"

  '/policytypes/{policyTypeId}/policies':
    get:
      description: 'Get all policy identifiers'
      tags:
      - All Policy Identifiers
      parameters:
        - name: policyTypeId
          in: path
          required: true
          schema:
            "$ref": "#/components/schemas/PolicyTypeId"
      responses:
        200:
          description: 'Array of all policy identifiers'
          content:
            application/json:
              schema:
                type: array
                items:
                  "$ref": "#/components/schemas/PolicyId"
                minItems: 0
        404:
          "$ref": "#/components/responses/404-NotFound"

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

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

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

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

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

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

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

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

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

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

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

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

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

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

A.3 Enrichment information API

```
openapi: 3.0.1
info:
  title: 'A1-EI Enrichment Information Service'
  version: 1.3.0
  description: |
    API for A1 Enrichment Information Service
    © 2022, O-RAN ALLIANCE
    All rights reserved
externalDocs:
  description: 'O-RAN.WG2.A1AP-v04.00 A1 interface: Application Protocol'
  url: 'https://www.o-ran.org/specifications'
servers:
  - url: '{apiRoot}/A1-EI/v1'
    variables:
      apiRoot:
        default: 'https://example.com'
        description: 'apiRoot as defined in clause 6.3.1 in O-RAN.WG2.A1AP'
paths:
  '/eitypes':
    get:
      description: 'Get all EI type identifiers'
      tags:
      - All EI Type Identifiers
      responses:
        200:
          description: 'Array of all EI type identifiers'
          content:
            application/json:
              schema:
                type: array
                items:
                  "$ref": "#/components/schemas/EiTypeId"
                minItems: 0

  '/eitypes/{eiTypeId}':
    parameters:
      - name: eiTypeId
        in: path
        required: true
        schema:
          "$ref": "#/components/schemas/EiTypeId"
    get:
      description: 'Get the schemas for an EI type'
      tags:
      - EI Type
      responses:
        200:
          description: 'The EI type schemas'
          content:
            application/json:
              schema:
                "$ref": "#/components/schemas/EiTypeObject"
        404:
          "$ref": "#/components/responses/404-NotFound"

  '/eijobs':
    get:
      description: 'Get all EI job identifiers'
      tags:
      - All EI job Identifiers
      parameters:
        - name: eiTypeId
          in: query
          schema:
            "$ref": "#/components/schemas/EiTypeId"
      responses:
        200:
          description: 'Array of all EI job identifiers'
          content:
            application/json:
              schema:
                type: array
                items:
                  "$ref": "#/components/schemas/EiJobId"
                minItems: 0
        404:
          "$ref": "#/components/responses/404-NotFound"

  '/eijobs/{eiJobId}':
    parameters:
      - name: eiJobId
        in: path
        required: true
        schema:
          "$ref": "#/components/schemas/EiJobId"
    put:
      description: 'Create, or update, an EI job'
      tags:
      - Individual EI job
      requestBody:
        required: true
        content:
          application/json:
            schema:
              "$ref": "#/components/schemas/EiJobObject"
      responses:
        200:
          description: 'The EI job was updated'
          content:
            application/json:
              schema:
                "$ref": "#/components/schemas/EiJobObject"
        201:
          description: 'The EI job was created'
          content:
            application/json:
              schema:
                "$ref": "#/components/schemas/EiJobObject"
          headers:
```

```
          Location:
            description: 'Contains the URI of the created EI job'
            required: true
            schema:
              type: string
        400:
          "$ref": "#/components/responses/400-BadRequest"
        404:
          "$ref": "#/components/responses/404-NotFound"
        409:
          "$ref": "#/components/responses/409-Conflict"
      callbacks:
        jobStatusNotification:
          '{$request.body.jobStatusNotificationUri}':
            post:
              description: 'Notify about status changes for this EI job'
              requestBody:
                required: true
                content:
                  application/json:
                    schema:
                      "$ref": "#/components/schemas/EiJobStatusObject"
              responses:
                204:
                  description: 'Notification received'
                400:
                 "$ref": "#/components/responses/400-BadRequest"
        jobResult:
          '{$request.body.jobResultUri}':
            post:
              description: 'Deliver result for this EI job'
              requestBody:
                required: true
                content:
                  application/json:
                    schema:
                      "$ref": "#/components/schemas/EiResultObject"
              responses:
                204:
                  description: 'Information received'
                400:
                 "$ref": "#/components/responses/400-BadRequest"
    get:
      description: 'Query an EI job'
      tags:
      - Individual EI job Object
      responses:
        200:
          description: 'The requested EI job'
          content:
            application/json:
              schema:
                "$ref": "#/components/schemas/EiJobObject"
        404:
          "$ref": "#/components/responses/404-NotFound"
    delete:
      description: 'Delete an EI job'
      tags:
      - Individual EI job
      responses:
        204:
          description: 'The EI job was deleted'
        404:
          "$ref": "#/components/responses/404-NotFound"

  '/eijobs/{eiJobId}/status':
    parameters:
      - name: eiJobId
        in: path
        required: true
        schema:
          "$ref": "#/components/schemas/EiJobId"
    get:
      description: 'Query status for an EI job'
      tags:
      - Individual EI job Object
      responses:
        200:
          description: 'The requested EI job status'
          content:
            application/json:
              schema:
                "$ref": "#/components/schemas/EiJobStatusObject"
        404:
          "$ref": "#/components/responses/404-NotFound"

components:
  schemas:
    #
    # Representation objects
    #
    EiTypeObject:
      description: 'A definition of an EI type, i.e. the JSON schemas for an EI job, its status and
its result, and the job constraints''
      type: object
      properties:
        eiJobDefinitionSchema:
          "$ref": "#/components/schemas/JsonSchema"
        eiJobStatusSchema:
          "$ref": "#/components/schemas/JsonSchema"
        eiJobResultSchema:
          "$ref": "#/components/schemas/JsonSchema"
        eiJobConstraintsSchema:
          "$ref": "#/components/schemas/JsonSchema"

    EiJobObject:
      description: 'A generic EI job object that can be used to transport any EI job.'
      type: object
      properties:
        eiTypeId:
          type: string
        jobResultUri:
          type: string
        jobStatusNotificationUri:
          type: string
        jobDefinition:
          type: EiJobDefinition
      required:
      - eiTypeId
      - jobResultUri
      - jobDefinition

    EiJobDefinition:
      description: 'An object representing an EI job definition.'
      type: object

    EiJobStatusObject:
      description: 'A generic EI job status object that can be used to transport any EI job status.'
      type: object
      properties:
        eiJobStatus:
          type: string
          enum:
          - ENABLED
          - DISABLED
      required:
      - eiJobStatus

    EiResultObject:
      description: 'A generic EI job result object that can be used to transport any EI job result.'
      type: object

    EiJobConstraintsObject:
      description: 'A generic EI job constraints object.'
      type: object
    ProblemDetails:
      description: 'A problem detail to carry details in an HTTP response according to RFC 7807'
      type: object
      properties:
        type:
          type: string
        title:
          type: string
        status:
          type: number
        detail:
          type: string
        instance:
          type: string

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

    EiJobId:
      description: 'EI job identifier assigned by the A1-EI Consumer when an EI job is created'
      type: string

    EiTypeId:
      description: 'EI type identifier assigned by the A1-EI Provider'
      type: string

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

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

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

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

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

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

History

Date	Revision	Description
2022.11.17	04.00	Aligning to O-RAN drafting rules. Enhanced alignment between A1-P and A1-EI, and between A1AP and A1TD.
2022.04.01	03.02	Enhancing alignment and consistency between A1-P and A1-EI OpenAPIs
2021.03.13	03.01	Separation of application protocol from type definitions. Data models and type definitions moved to A1 interface: Type Definitions v01.00.
2020.11.09	03.00	Defining A1-EI/v1 (A1 enrichment information service)
2022.07.30	02.00	Defining A1-P/v2 based on policy types
2020.03.13	01.01	Removal of multi-object operations and PATCH based procedures. Included OpenAPI Specification and aligned text with it.
2019.09.30	01.00	First version with A1-P/v1 (A1 policy management service)