O-RAN.WG2.R1AP-R003-v01.00

Technical Specification

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

Application Protocols for R1 Services

O-RAN ALLIANCE e.V., Buschkauler Weg 27, 53347 Alfter, Germany

Register of Associations, Bonn VR 11238, VAT ID DE321720189

Contents

Foreword

This Technical Specification (TS) has been produced by O-RAN Alliance.

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 Protocols for R1 Services. It is part of a TS-family covering the WG2: R1 Interface Specifications.

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] 3GPP TS 29.501 "5G System; Principles and Guidelines for Services Definition; Stage 3(Release 16)", September 2020.

[2] 3GPP TS 29.500: "5G System; Technical Realization of Service Based Architecture; Stage 3".

[3] OpenAPI: "OpenAPI 3.0.0 Specification", http://spec.openapis.org/oas/v3.0.1.html.

[4] ETSI GS NFV-SOL 013 Rel3: "Protocols and Data Models; Specification of common aspects for RESTful NFV MANO APIs".

[5] O-RAN TS: "R1 General Aspects and Principles" ("R1GAP").

[6] O-RAN TS: "R1 Use Cases and Requirements" ("R1UCR").

[7] O-RAN TS: "Transport Protocols for R1 services"("R1TP").

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

[9] 3GPP TS 29.222: "Common API Framework for 3GPP Northbound APIs".

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

[11] Semantic Versioning 2.0.0: https://semver.org.

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

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.

Void

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

API consumer: A Service Consumer consuming one or more services using APIs

NOTE: The Service Consumer role is introduced in R1GAP [5].

API producer: The Service Producer that offers its services for consumption via APIs

NOTE: The Service Producer role is introduced in R1GAP [5].

3.2 Symbols

Void

3.3 Abbreviations

For the purposes of the present document, the follow abbreviations apply:

API	Application programming interface
FQDN	Fully qualified domain name

4 Application protocol for the R1 services

4.1 Introduction

The present document contains a realization for the procedures identified in R1GAP [5]. It is based on transport protocols as defined in R1TP [7]. This definition of the R1 Application Protocols (R1AP) defined in the present document is based on the 3GPP service framework for network functions specified in 3GPP TS 29.501 [1] .

4.2 Version convetions for the present document

The version number of the present document follows the "major.minor" versioning scheme. There could be implications for the interoperability between rApps and R1 service API implementations in SMO/Non-RT RIC framework that are based on different versions of this specification.

An incremented "major" version field of the present document could indicate that a new major feature (e.g., a new R1 service) has been added or that an incompatible change has been made to one or more R1 service APIs. An incremented "minor" version field could indicate that an optional feature has been added, a technical issue has been fixed, or that clarifications or editorial corrections have been made.

There are separate version conventions defined for the RESTful R1 service APIs (see clause 5.2) and for the related Open API documents (see Annex A).

5 RESTful R1 service APIs

5.1 Overview

The present document defines the protocol for the following R1 services interfaces in the form of RESTful Application Programming Interface (API) specifications, as listed in table 5.1-1

Table 5.1-1: RESTful R1 service APIs and their versions defined in the present document

Service Group	Service API	API version	Open API Version
Service management and exposure	Service registration	0.0.0	-
Service management and exposure	Service discovery	0.0.0	-

The design of the above APIs is based on the procedures and requirements defined in R1UCR [6] and R1GAP [5].

This definition of the RESTful R1 service APIs is based on the 3GPP service framework specified in 3GPP TS 29.501 [1] .

5.2 URI structure and supported content formats

This clause specifies the URI prefix and the supported formats applicable to the RESTful R1 service APIs.

All resource URIs of the APIs shall have the following prefix, except the "API versions" resource which shall follow the rules specified in clause 9.3 of ETSI GS NFV-SOL 013 [4]:

{apiRoot}/<apiName>/<apiMajorVersion>/

The request URIs used in HTTP requests from the API service consumer towards the API service producer shall have the resource URI structure defined in clause 4.4.1 of 3GPP TS 29.501 [1] , i.e.:

{apiRoot}/<apiName>/<apiMajorVersion>/<apiSpecificResourceUriPart>

with the following components:

- The {apiRoot} shall be set as described in clause 4.4.1 of 3GPP TS 29.501 [1] ; however, the restrictions w.r.t the operator specific FQDN of the host portion defined there do not apply.
- The <apiName> indicates the API name of the service interface in an abbreviated form. It is defined in the clause specifying the corresponding RESTful R1 service API.
- The <apiMajorVersion> indicates the major version of the API and is defined in the clause specifying the corresponding RESTful R1 serviceAPI.
- Each <apiSpecificResourceUriPart> represent a specific resource of the API. It is defined in the corresponding RESTful R1 service API for each one of the defined resources.

For HTTP requests and responses that have message content, the content format JSON (see IETF RFC 8259 [12]) shall be supported. The JSON format shall be signalled by the content type "application/json".

All resource URIs of the API shall comply with the URI syntax as defined in IETF RFC 3986 [8]. An implementation that dynamically generates resource URI parts (individual path segments, sequences of path segments that are separated by "/", query parameter values) shall ensure that these parts only use the character set that is allowed by IETF
RFC 3986 [8] for these parts.

5.3 General considerations for RESTful R1 service APIs

5.3.1 Usage of HTTP header fields

HTTP headers are components of the headers section of the HTTP request and response messages. The usage of HTTP header fields shall follow the definitions in clause 4.2 of ETSI GS NFV-SOL 013 [4].

5.3.2 Handling of large query results

The handling of large query results shall be supported by RESTful R1 service APIs as specified in clause 5.4.2 of ETSI GS NFV-SOL 013 [4].

5.3.3 Error reporting

In RESTful interfaces, application errors are mapped to HTTP errors. Since HTTP error information is generally not enough to discover the root cause of the error, additional application specific error information is typically delivered.

The RESTful R1 service APIs shall support the error reporting as specified in clause 6 of ETSI GS NFV-SOL 013 [4].

NOTE: ETSI GS NFV-SOL 013 [4] refers to "RESTful NFV-MANO API" specification; and wherever such reference is provided "RESTful R1 service API" is to be considered instead for the purpose of the present document.

If an HTTP method is not defined for a particular resource in the present document, that method is not supported. When that method is requested on the resource, the API producer shall return a "405 Method Not Allowed" response as defined in clause 6.4 of ETSI GS NFV-SOL 013 [4].

6 Service management and exposure services

6.1 Service registration API

6.1.1 Usage of HTTP header fields

This API allows the API producer to manage service registrations based on the procedures for “Registration of services” defined in R1GAP [5].

6.1.2 API version

For the service registration API as specified in the present document, the MAJOR version field shall be 0, the MINOR version field shall be 0 and the PATCH version field shall be 0 (see clause 9.1 of ETSI GS NFV-SOL 013 [4] for a definition of the version fields). Consequently, the <apiMajorVersion> URI variable shall be set to "v0".

6.1.3 Resource structure and methods

The request URIs used in HTTP requests from the API consumer towards the API producer shall have the resource URI structure as defined in clause 5.2. The <apiName> resource URI variable shall be "serviceregistration". The <apiSpecificResourceUriPart> for each resource shall be set as described in clause 6.1.5.

Figure 6.1.3-1 shows the overall resource URI structure defined for the service registration API.

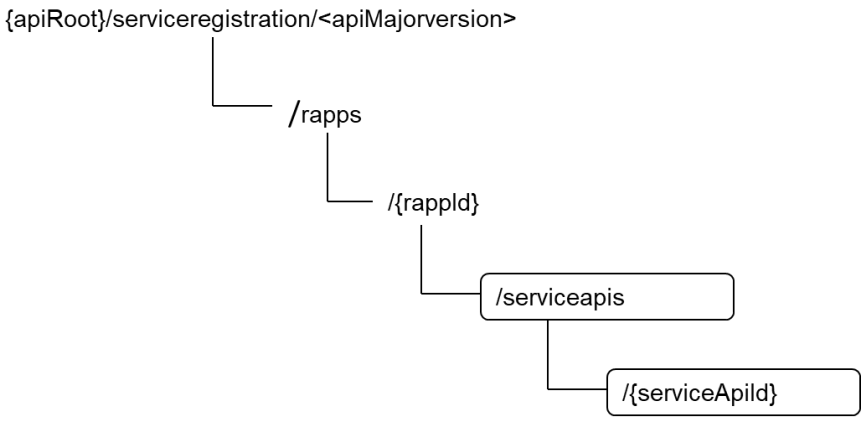

Figure 6.1.3-1: Resource URI structure of the service registration API

Table 6.1.3-1 lists the individual resources defined for the API, the applicable HTTP methods and the associated service operations.

Table 6.1.3-1: Resources and methods overview of the service registration API

Resource Name	Resource URI	HTTP method	Service Operation
Registered service APIs	...rapps/{rappId}/serviceapis	POST	Register service API

6.1.4 Service operations

6.1.4.1 Register service API

6.1.4.1.1 Operation definition

A Service Producer uses the Register service API operation as API consumer to register the capability to produce a service with service profile information with the API producer.

The operation is based on HTTP POST.

```
@startuml
autonumber
participant "API consumer" as cons
participant "API producer" as prod
cons ->> prod : POST .../rapps/{rappId}/serviceapis (ServiceProfile)
prod -->> cons : 201 Created (ServiceProfile)
@enduml
```

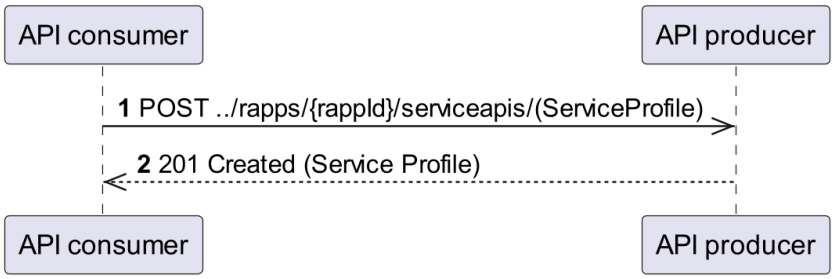

Figure 6.1.4.1.1-1: Register service API operation

The service operation is as follows:

1) The API Consumer shall send an HTTP POST request to the API Producer that includes the rApp identifier and the service profile defining the service to be registered. The API Producer shall process the service registration details received in the HTTP POST message and determine if the request sent by the API Consumer is authorized or not.

2) The API Producer shall return the HTTP POST response. On success "201 Created" is returned. The message body carries the registered service profile and the "Location" HTTP header is present and shall carry the URI for the newly created service resource including the service API identifier generated by the API producer. On failure, the appropriate error code shall be returned, and the message response body may contain additional error information.

6.1.4.1.2 Referenced procedures

6.1.4.1.2.1 Register service procedure

The Register service procedure defined in R1GAP [5] is based on the Register service API operation illustrated in figure 6.1.4.1.1-1.

6.1.4.2 Update Registered service API

NOTE: This operation is not specified in the present document.

6.1.4.3 Deregister service API

NOTE: This operation is not specified in the present document

6.1.5 Resources

6.1.5.1 Overview

This clause defines the resouces for the service registration API.

6.1.5.2 Resource: "Registered service APIs"

6.1.5.2.1 Description

The Registered service APIs resource represents the set of registered services APIs.

6.1.5.2.2 Resource definition

Resource URI**: {apiRoot}/serviceregistration/< apiMajorVersion>/rapps/{rappId}/serviceapis**

The resource URI variables supported by the resource shall be defined as table 6.1.5.2.2-1 illustrates.

Table 6.1.5.2.2-1: Resource URI variables for the resource

Name	Definition
apiRoot	See clause 5.2
apiMajorVersion	See clause 6.1.2
rappId	Identifier of rApp

6.1.5.2.3 Resource standard methods

6.1.5.2.3.1 POST

This method shall support the URI query parameters specified in the table 6.1.5.2.3.1-1, the request data structures specified in table 6.1.6.1.2 -2 and the response data structure and response codes specified in table 6.1.5.2.3.1-3 and the response headers specified in table 6.1.5.2.3.1-4.

Table 6.1.5.2.3.1-1: URI query paramters supported by the POST request on this resource

Attribute Name	Data type	P	Cardinality	Description
none				

Table 6.1.5.2.3.1-2: Data structures supported by the POST request body on this resource

Data type	P	Cardinality	Description
ServiceProfile	M	1	Service profile of the service to be registered, as defined in clause 6.1.6.1.2.1.

NOTE: It is FFS whether the rApp identifier is represented explicitly by a URI parameter or implicitly bound to the access token.

Table 6.1.5.2.3.1-3: Data structures supported by the POST Response Body on this resource

Data type	P	Cardinality	Response codes	Description
ServiceProfile	M	1	201 Created	Service API registered successfully. The URI of the created resource is returned in the "Location" HTTP header

Table 6.1.5.2.3.1-4: HTTP Header for the response

Response code	Header name	Data type	P	Cardinality	Description
201	Location	string	M	1	Contains the URI of the newly created resource, according to the structure: {apiRoot}/serviceregistration/<apiMajorVersion>/apps/{rappId}/serviceapis/{serviceApiId}

6.1.5.2.3.2 Resource custom operations

None

6.1.5.2.4 Custom operations without associated resources

None

6.1.5.2.5 Notifications

None

6.1.6 Data model

6.1.6.1 General

This clause specifies the application data model supported by the API.

6.1.6.1.1 Simple data types and enumerations

None

6.1.6.1.2 Structured data types

6.1.6.1.2.1 Type: Service Profile

NOTE: This Data type Service Profile is not specified in the present document.

6.1.6.1.3 Re-used data types

None

6.2 Service discovery API

6.2.1 Introduction

This API allows the API consumer to perform service discovery based on the service discovery procedures defined in R1GAP [5].

6.2.2 API version

For the service discovery API as specified in the present document, the MAJOR version field shall be 0, the MINOR version field shall be 0 and the PATCH version field shall be 0 (see clause 9.1 of ETSI GS NFV-SOL 013 [4] for a definition of the version fields). Consequently, the <apiMajorVersion> URI variable shall be set to "v0".

6.2.3 Resource structure and methods

The request URIs used in HTTP requests from the API consumer towards the API producer shall have the resource URI structure as defined in clause 5.2. The <apiName> resource URI variable shall be "servicediscovery ". The <apiSpecificResourceUriPart> for each resource shall be set as described in clause 6.2.5.

Figure 6.2.3-1 shows the overall resource URI structure defined for the service discovery API.

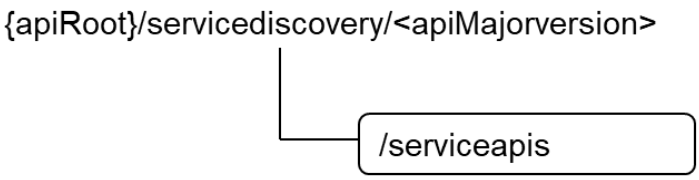

Figure 6.2.3-1: Resource URI structure of the service discovery API

Table 6.2.3-1 lists the individual resources defined for the API, the applicable HTTP methods and the associated service operations.

Table 6.2.3-2: Resources and methods overview of the service discovery API

Resource Name	Resource URI	HTTP method	Service Operation
All service APIs	.../serviceapis	GET	Query service APIs

6.2.4 Service operations

6.2.4.1 Query service APIs

6.2.4.1.1 Operation definition

The API consumer uses the Query service APIs operation to discover service APIs information.

The operation is based on HTTP GET as per figure 6.2.4.1.1-1. The HTTP GET response contains information about all services that the API consumer is authorized to access.

```
@startuml

autonumber
participant "API consumer" as cons

participant "API producer" as prod

cons ->> prod : GET .../serviceapis
prod -->> cons : 200 OK {services API descriptions}
@enduml
```

Figure 6.2.4.1.1-1: Query service APIs operation

This service operation is as follows:

1) The API consumer shall send an HTTP GET request to the API producer that includes the rApp identifier and optional filtering criteria. The API producer shall process the service discovery details received in the HTTP GET message and determine if the request sent by the API consumer is authorized or not.

2) The API producer shall return the HTTP GET response. On success "200 OK" shall be returned and the message content shall carry a list of services API descriptions. On failure, the appropriate error code shall be returned, and the response message content may contain additional error information.

6.2.4.1.2 Referenced procedures

6.2.4.1.2.1 Discovery services procedure

The Discover services procedure as defined in R1GAP [5] is based on the Query service APIs operation illustrated in figure 6.2.4.1.1-1.

6.2.5 Resources

6.2.5.1 Overview

This clause defines the resouces for the Service discovery API.

6.2.5.2 Resource: "All service APIs"

6.2.5.2.1 Description

The "All service APIs" resource represents a collection of discoverable service APIs.

6.2.5.2.2 Resource definition

Resource URI**: {apiRoot}/servicediscovery/< apiMajorVersion>/serviceapis**

The resource URI and the supported resource URI variables are as defined in clause 6.2.3.

The resource URI variables supported by the resource shall be defined as table 6.2.5.2.2-1 illustrates.

Table 6.2.5.2.2-1: Resource URI variables for the resource

Name	Definition
apiRoot	See clause 5.2
apiMajorVersion	See clause 6.2.2

6.2.5.2.3 Resource standard methods

6.2.5.2.3.1 GET

This method shall support the URI query parameters specified in the table 6.2.5.2.3.1-1, the request data structures specified in table 6.2.5.2.3.1-2 and the response data structures and response codes specified in table 6.2.5.2.3.1-3.

Table 6.2.5.2.3.1-1: URI query parameters supported by the GET method on this resource

Name	Data type	P	Cardinality	Description	Applicability
serviceName	string	O	0..1	Filter the response by service name, matching the attribute “serviceName” in the service profile definition in Table [6.2.6.2.6.1.2]	
serviceVersion	string	O	0..1	Filter the response by service version, as defined by the attribute “serviceVersion” in the service profile definition in Table [6.2.6.2.6.1.2]	

NOTE: It is FFS whether serviceName and serviceVersion can also include a list of strings to match

Table 6.2.5.2.3.1-2: Data structures supported by the GET Request Body on this resource

Data type	P	Cardinality	Description
n/a			

Table 6.2.5.2.3.1-3: Data structures supported by the GET Response Body on this resource

Data type	P	Cardinality	Response codes	Description
array(ServiceProfile)	O	0..N	200 OK	The response body contains information about the discovered APIs.

6.2.5.2.3.2 Resource custom operations

None

6.2.5.2.4 Custom operations without associated resources

None

6.2.5.2.5 Notifications

None

6.2.6 Data model

6.2.6.1 General

This clause specifies the application data model supported by the API.

6.2.6.1.1 Simple data types and enumerations

None

6.2.6.1.2 Structured data types

6.2.6.1.2.1 Type: ServiceProfile

Table 6.2.6.1.2.1-1: Definition of type Service Profile

Attribute name	Data type	P	Cardinality	Description	Applicability

NOTE: The DataType ServiceProfile is not specified in the present document

6.2.6.1.3 Re-used data types

Table 6.2.6.1.3-1 specifies data types re-used by the API service:

Table 6.2.6.1.3-1: Re-used Data Types

Data type	Reference	Comments	Applicability

Annex A (normative): OpenAPI specifications

A.1 General

This Annex formally specifies the RESTful R1 services APIs by defining OpenAPI documents in YAML format that comply with the OpenAPI 3.0.0 Specification [4].

A.1.1 Versioning of OpenAPI specifications for the RESTful R1 service APIs

The Open API specifications of the RESTful R1 service APIs provided in this annex are versioned by Semantic Versioning 2.0.0 [11] as defined in the OpenAPI Specification [4]. When included in the present specification, the OpenAPI documents are considered as released and are versioned using three numbers major.minor.patch and optional build metadata where the main compatibility expectations stated for Sematic Versioning [11] imply:

1) The "major" version field is stepped up when incompatible API changes are made to the OpenAPI document. The "major" version in the OpenAPI document corresponds to the major API version in the resource URIs for of the RESTful R1 APIs defined in clause 5.
2) The "minor" version field is stepped up when features are added to the OpenAPI document in a way that keeps implementations compatible even if not all added features are not supported by both the service producer and the service consumer of the R1 service.
3) The "patch" version is stepped up when errors are corrected in a backward compatible way.
4) When applicable, build metadata are used to represent editorial changes to the OpenAPI document, i.e., changes that have no technical impact. According to Semantic Versioning [11], build metadata are denoted by appending a plus ("+") sign followed by an integral number that is increment at every version that includes only the editorial changes, e.g 1.0.0+0, 1.0.0+1 etc.

NOTE: rApps and SMO/Non-RT RIC framework implementations may support multiple API versions of an R1 service.

Annex (informative): Bibliography

ETSI GS NFV-SOL 015v1.2.1: " Protocols and Data Models; Specification of Patterns and Conventions for RESTful NFV-MANO APIs", December 2020.

Revision history

Date	Revision	Description
2022.08.15	00.00.01	Intial proposed skeleton for the Application protocols for R1 Services
2022.08.18	00.00.02	Intial agreed skeleton for the Application protocols for R1 Services
2022.10.29	00.00.03	Added CR 0121 Version 05 on API introduction, Added CR 0131 Version 05 on Open API specification versioning, Added CR0122 Version 05 on RESTful API versioning
2022-11-10	00.00.04	Added CR 0123 Version 04 on Service discovery API Added CR 0133 Version 04 on Service registration API
2022-11-18	00.00.05	Updated the document based on comments from Ericsson,Jio,Nokia,Intel and Qualcomm

History

Date	Revision	Description
2022-11-19	01.00	Published version 01.00

