**3GPP TSG-CT WG4 Meeting #99eC4-204abc**

**E-Meeting, 18th – 28th June 2020 *was* C4-204307**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *CR-Form-v12.0* | | | | | | | | |
| **CHANGE REQUEST** | | | | | | | | |
|  | | | | | | | | |
|  | **29.501** | **CR** | **0090** | **rev** | **1** | **Current version:** | **16.4.0** |  |
|  | | | | | | | | |
| *For* [***HE******LP***](http://www.3gpp.org/3G_Specs/CRs.htm#_blank)*on using this form: comprehensive instructions can be found at* [*http://www.3gpp.org/Change-Requests*](http://www.3gpp.org/Change-Requests)*.* | | | | | | | | |
|  | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Proposed change affects:*** | UICC apps |  | ME |  | Radio Access Network |  | Core Network | **X** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | |
| ***Title:*** | Miscellaneous corrections | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Source to WG:*** | Huawei | | | | | | | | | |
| ***Source to TSG:*** | CT4 | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Work item code:*** | SBIProtoc16 | | | | |  | ***Date:*** | | | 2020-08-11 |
|  |  | | | |  | |  | | |  |
| ***Category:*** | **F** |  | | | | | ***Release:*** | | | Rel-16 |
|  | *Use one of the following categories:* ***F*** *(correction)* ***A*** *(mirror corresponding to a change in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)*  Detailed explanations of the above categories can be found in 3GPP [TR 21.900](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | | | | | | | | *Use one of the following releases: Rel-8 (Release 8) Rel-9 (Release 9) Rel-10 (Release 10) Rel-11 (Release 11) Rel-12 (Release 12)* *Rel-13 (Release 13) Rel-14 (Release 14) Rel-15 (Release 15) Rel-16 (Release 16)* | |
|  |  | | | | | | | | | |
| ***Reason for change:*** | | 1.Correct a reference error.  2.Correct some editorial errors  3.Correct some delimiters error with < > instead of { }  4.Correct some description errors. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Summary of change:*** | | 1.Correct some reference errors  2.Correct some editorial errors  3.Correct some delimiters error with < > instead of { }  4.Correct some description errors. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Consequences if not approved:*** | | The errors in the specification are confusing or misleading. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Clauses affected:*** | | 4.4.2, 4.6.1.4, 4.6.2.4, 5.3.6, 5.2.2, 5.3.3, 5.3.4, 5.3.5, 5.3.7, 5.3.16, 6.2, C.4 | | | | | | | | |
|  | |  | | | | | | | | |
|  | | **Y** | **N** |  | | | |  | | |
| ***Other specs*** | |  | **X** | Other core specifications | | | | TS/TR ... CR ... | | |
| ***affected:*** | |  | **X** | Test specifications | | | | TS/TR ... CR ... | | |
| ***(show related CRs)*** | |  | **X** | O&M Specifications | | | | TS/TR ... CR ... | | |
|  | |  | | | | | | | | |
| ***Other comments:*** | | This CR won’t introduce any impacts to the OpenAPI specification files. | | | | | | | | |
|  | |  | | | | | | | | |
| ***This CR's revision history:*** | | Rev1:  1. in clause 5.3.3 curly quotes in version: ‘1.0.0’ was corrected. | | | | | | | | |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*The start of changes\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 4.4.2 Custom operations URI structure

The custom operation definition is in Annex C.

The URI of a custom operation which is associated with a resource shall have the following structure:

**{apiRoot}/<apiName>/<apiVersion>/<apiSpecificResourceUriPart>/<custOpName>**

Custom operations can also be associated with the service instead of a resource. The URI of a custom operation which is not associated with a resource shall have the following structure:

**{apiRoot}/<apiName>/<apiVersion>/<custOpName>**

In the above URI structures, "apiRoot", "apiName", "apiVersion" and "apiSpecificResourceUriPart" are as defined in clause 4.4.1 and "custOpName" represents the name of the custom operation as defined in clause 5.1.3.2.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Next change\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#### 4.6.1.4 Special provisions to support the seamless change of AMF as NF service producer

Services provided by the AMF can be transferred seamlessly to a new AMF when the corresponding UE context is transferred to that AMF.

To support a seamless change of the AMF as NF service producer, the procedures in clause 4.6.1 are applied with the following special provisions:

1. When becoming aware that a new AMF is serving the resource, the NF service consumer shall exchange the authority part of resource URIs with the address of a new NF service producer and shall use that URI in subsequent communication.

NOTE: An NF service consumer can become aware of an AMF change via Namf\_Communication service AMFStatusChange Notifications, via Error response from old AMF, via link level failures (e.g. no response from the AMF), or via a notification from the NRF that the AMF has deregistered and can then determine the new AMF either via information received within those services or by selecting an AMF from an earlier received AMF set or the backup AMF.

2. Each AMF within a set of AMFs supporting seamless changes shall be prepared to receive updates for resource URIs constructed according to bullet 1 with the own IP address as authority part from the NF service consumer, by either handling the updates, or by replying with an HTTP "307 temporary redirect" error response pointing to new NF service producer, or by replying with another HTTP error such as an "404 Not found".

3. For a service that includes notifications from the AMF, the NF service consumer shall be prepared to receive notifications for the that service from any NF service producer within a set of NF service producers supporting seamless changes.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Next change\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#### 4.6.2.4 Special provisions to support the seamless change of AMF as NF service consumer

Services consumed by an AMF can be transferred seamlessly to a new AMF when the corresponding UE context is transferred to that AMF.

To support a seamless change of AMF as NF service consumer, the procedures in clause 4.6.2 are applied with the following special provisions:

1. When becoming aware that a new AMF is requiring notifications related to a subscription resource, the NF service producer shall exchange the authority part of the corresponding Notification URI with the address of that new NF service consumer and shall use that URI in subsequent communication.

NOTE: An NF service producer can become aware of an AMF change via Namf\_Communication service AMFStatusChange Notifications, via Error response from old AMF, via link level failures (e.g no response from the AMF), or via a notification from the NRF that the AMF has deregistered and can then determine the new AMF either via information received within those services or selecting an AMF from an earlier received AMF set or the backup AMF.

2. Each AMF within a set of AMFs supporting seamless changes shall be prepared to receive notifications at the Notification URI constructed according to bullet 1 with the own IP address as authority part from the NF service producer, by either handling the notifications, or by replying with an HTTP "307 temporary redirect" error response pointing to new NF service consumer, or by replying with another HTTP error such as an "404 Not found".

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Next change\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 5.3.6 References to other 3GPP-defined OpenAPI specification files

Open API specification files may contain references to fragments of other 3GPP-defined Open API specification files.

Such references shall be formatted to refer to local files stored on the same folder.

NOTE 1: For the purpose of referencing, it is assumed that each OpenAPI specification file contained in a 3GPP specification is stored as separate physical file and that all OpenAPI specification files are stored in the same directory on the local server.

The referenced file names for other 3GPP-defined Open API specification files shall comply with the following convention, unless a specific file name is indicated in the Annex of a 3GPP specification defining an OpenAPI specification file. The file name shall consist of (in the order below):

- the 3GPP specification number in the format "TSxxyyy";

- an "\_" character;

- if the OpenAPI specification file contains an API definition: the API name which shall be taken from the heading of the relevant annex A.x as defined in the corresponding 3GPP TS of that API.

- if the OpenAPI specification file contains a definition of CommonData: the string "CommonData"; and

- the string ".yaml".

NOTE 2: The informative copies of OpenAPI specification files contained in 3GPP Technical Specifications at the public 3GPP file server (see clause 5.3.1) follow the above conventions and can be copied into a local folder in order to resolve references.

Such a reference to another OpenAPI specification file shall be interpreted as refering to the related OpenAPI specification file contained in the version of the corresponding 3GPP TS indicated in the reference clause of the specification, i.e. for a non-specific reference the latest version of that 3GPP TS in the same Release as the specification.

EXAMPLE: Reference to Data Type "Xxx" defined in the same file

$ref: '#/components/schemas/Xxx'

EXAMPLE: Reference to Data Type "Xxx" defined as Common Data in 3GPP TS 29.571:

$ref: 'TS29571\_CommonData.yaml#/components/schemas/Xxx'

EXAMPLE: Reference to Data Type "Xxx" defined within API "Nudm\_UEAU" in 3GPP "TS 29.503":

$ref: 'TS29503\_Nudm\_UEAU.yaml#/components/schemas/Xxx'

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Next change\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 5.2.2 Resources and HTTP Methods

Resources and HTTP methods shall specify the resource URI, resource URI variables for the resource and the standard HTTP methods supported by the resource.

Example:

Resource URI: **{apiRoot}/<apiName>/<apiVersion>/<apiSpecificResourceUriPart>**

The resource URI variables supported by the resource shall be defined as table 5.2.2-1 illustrates.

Table 5.2.2-1: Resource URI variables for the resource

|  |  |
| --- | --- |
| Name | Definition |
| apiRoot | See clause 4.4.1 |
| < Name of resource URI variables for resource> | < Definition of resource URI variables for resource > |

The {apiRoot} URI variable should be defined in clauses and this definition should be referenced to ease a possible update of the apiVersion value.

Each method supported by the resource shall be described including the URI query parameters supported by the method, data structures supported by the request body, and the data structures supported by the response body.

URI query parameters supported by the method shall be defined as table 5.2.2-2 illustrates.

Table 5.2.2-2: URI query parameters supported by a method on the resource

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Data type | P | Cardinality | Description | Applicability |
| <name> or n/a | <type> or <leave empty> | <M, C or O> | 0..1 or 1 or 0..N or 1..N or <leave empty> | <only if applicable> |  |

**Name**: Name of query parameter in URI. If no query parameters are defined for the URI, the name should be marked as "n/a".

**Data type**: Data type of URI query parameters, i.e. a data type defined in the specification. If no query parameter is defined for the URI, the column is left empty.

**P**: Presence condition of URI query parameters. It shall be one of "M" (for Mandatory), "C" (for Conditional) and "O" (for Optional). If no query parameter is defined for the URI, the column is left empty.

**Cardinality**: Defines the allowed number of occurrence. It shall be "0..1", "1", "0..N", "1..N" . If no query parameter is defined for the URI, the column is left empty.

**Description**: Additional information for URI query parameter, i.e. describes the use of the parameter or the presence condition of the parameter and so on.

**Applicability**: If the URI query parameter is only applicable for optional feature(s) negotiated using the mechanism defined in clause 6.6.2 of 3GPP TS 29.500 [2], the name of the corresponding feature(s) shall be indicated in this column. If no feature is indicated. the attribute can be used with any feature.

NOTE 1: If no optional features are defined for an API, the applicability column can be omitted for that API.

Data structures supported by the request body of the method shall be specified as table 5.2.2-3 illustrates.

Table 5.2.2-3: Data structures supported by the request body on the resource

|  |  |  |  |
| --- | --- | --- | --- |
| Data type | P | Cardinality | Description |
| "<type>" or "array*(<type>*)" or "map*(<type>*)" or n/a | "M", "C" or "O" | "0..1", "1", or "M..N", or <leave empty> | <only if applicable> |

**Data type**: Data type of the data structure in the request body. If the data type is indicated as "*<type>*", the request body shall be of data type *<type>.* If the data type is indicated as "array(*<type>*)", the request body shall be an array (see IETF RFC 8259 [3]) that contains elements of data type *<type>*. If the data type is indicated as "map(*<type>*)", the request body shall be an object (see IETF RFC 8259 [3]) encoding a map (see clause 5.2.4.2) that contains as values elements of data type *<type>*. *<type>* can either be "integer", "number", "string" or "boolean" (as defined in the OpenAPI specification [4]), or a data type defined in a 3GPP specification. If no request body is allowed, the Data type shall be marked as "n/a".

**P**: Presence condition of a data structure in request body. It shall be one of "M" (for Mandatory), "C" (for Conditional) and "O" (for Optional).

**Cardinality**: Defines the allowed number of occurrence of data type *<type>*. A cardinality of "*M*..*N*", is only allowed for data types "array*(<type>*)" and "map*(<type>*)" and indicates the number of elements within the array or map; the values *M* and *N* can either be the characters "M" and "N", respectively, or integer numbers with M being greater than or equal 0, and N being greater than 0 and M. For data type "*<type>*", the cardinality shall be set to "0..1" if the Presence condition is "C" or "O", and to "1" if the Presence condition is "M". The Cardinality shall be left empty if no request body is allowed.

**Description**: Additional information for a data structure, i.e. describes the use of the data structure or the presence condition of the data structure and so on.

NOTE 2: The cardinality of "0..N" does not imply that the presence condition of the array or map is optional or conditional, i.e. the presence condition can be "M" while the cardinality is "0..N", the presence condition can be "O" or "C" while the cardinality is "1..N".

Data structures supported by the response body of the method shall be specified as table 5.2.2-4 illustrates.

Table 5.2.2-4: Data structures supported by the response body on the resource

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data type | P | Cardinality | Response  codes | Description |
| "*<type>*" or "array*(<type>*)" or "map*(<type>*)" or n/a | "M", "C" or "O" | "0..1", "1", or "M..N", or <leave empty> | <list applicable codes with name from the applicable RFCs> | <Meaning of the success case>  or  <Meaning of the error case with additional statement regarding error handling> |

**Data type**: Data type of the data structure in the response body. If the data type is indicated as "*<type>*", the response body shall be of data type *<type>.* If the data type is indicated as "array(*<type>*)", the response body shall be an array (see IETF RFC 8259 [3]) that contains elements of data type *<type>*. If the data type is indicated as "map(*<type>*)", the response body shall be an object (see IETF RFC 8259 [3]) encoding a map (see clause 5.2.4.2) that contains as values elements of data type *<type>*. *<type>* can either be "integer", "number", "string" or "boolean" (as defined in the OpenAPI specification [4]), or a data type defined in a 3GPP specification. If no response body is allowed, the Data type shall be marked as "n/a".

**P**: Presence condition of a data structure in response body. It shall be one of "M" (for Mandatory), "C" (for Conditional) and "O" (for Optional).

**Cardinality**: Defines the allowed number of occurrence of data type *<type>*. A cardinality of "*M*..*N*", is only allowed for data types "array*(<type>*)" and "map*(<type>*)" and indicates the number of elements within the array or map; the values *M* and *N* can either be the characters "M" and "N", respectively, or integer numbers with M being greater than or equal 0, and N being greater than 0 and M. For data type "*<type>*", the cardinality shall be set to "0..1" if the Presence condition is "C" or "O", and to "1" if the Presence condition is "M". The Cardinality shall be left empty if no response body is allowed.

**Response codes**: Lists applicable response codes with name from HTTP Status Code Registry at IANA [12]. Mandatory HTTP status codes listed in Table 5.2.7.1-1 of 3GPP TS 29.500 [2] for the corresponding HTTP method shall only be included if specific clarifications in the description part or special data types of the response body are required. Applicable HTTP status codes in addition to the mandatory HTTP status codes listed in Table 5.2.7.1-1 of 3GPP TS 29.500 [2] for the corresponding HTTP method shall be included.

**Description**: Additional information for a response, i.e. describes the meaning of the success case or meaning of the error case with additional statement regarding error handling.

NOTE 3: The cardinality of "0..N" does not imply that the presence condition of the array or map is optional or conditional, i.e. the presence condition can be "M" while the cardinality is "0..N", the presence condition can be "O" or "C" while the cardinality is "1..N".

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Next change\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 5.3.3 Info

The OpenAPI specification file of an API shall contain an "info" object with the title that should be set to the same value as chosen for the API name in the heading of Annex A.x of the corresponding 3GPP TS, and with the version set as described in clause 4.3.

The "info" object shall also include a "description" field, containing the name of the service that the API implements, and the same copyright notice as included in the front page of the corresponding 3GPP TS that specifies the API. The content of the "description" field shall be formatted using the YAML block notation for scalars (i.e. using the "|" character).

EXAMPLE: "info" object with the title, version and description of the API.

info:

title: 'Nsmf\_PDUSession'

version: '1.0.0'

description: |

SMF PDUSession Service.

© 2019, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).

All rights reserved.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Next change\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 5.3.4 externalDocs

Each OpenAPI specification file shall provide an "externalDocs" object as illustrated in the example below that shall contain:

- within the "description" field the 3GPP TS number, the version number and the name of the 3GPP TS describing the API, and

- within the "url" field a reference to the folder of that TS within the specification archive of the public 3GPP fileserver (i.e. "[https://www.3gpp.org/ftp/Specs/archive/<specSeries>/<SpecNumber>/](https://www.3gpp.org/ftp/Specs/archive/%3cspecSeries%3e/%3cSpecNumber%3e/)").

The version number in the "externalDocs" object shall be updated each time when the TS version contains new changes to the OpenAPI specification file.

NOTE 1: If a new TS version is provided without any changes to the OpenAPI specification file, the TS version number included in the "description" field of the "externalDocs" field in the OpenAPI specification file is not updated.

NOTE 2: If a new TS version is provided with changes to the OpenAPI specification file, the TS version number included in the "description" field of the "externalDocs" object is updated. However, if the API version in a given release is still the same as in a previous release, the first TS version in the first release that contains that API version is provided as TS version within the "externalDocs" object also in the TS versions of the subsequent releases.

EXAMPLE: "externalDocs" object.

externalDocs:

description: 3GPP TS 29.503 V15.1.0; 5G System; Unified Data Management Services

url: 'http://www.3gpp.org/ftp/Specs/archive/29\_series/29.503/'

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Next change\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 5.3.5 Servers

As defined in clause 4.4.1, the API URI consists of **{apiRoot}/<apiName>/<apiVersion>**. It shall be encoded in the corresponding OpenAPI specification file as "servers" field with **{apiRoot}** as variable.

Example:

servers:

- url: '{apiRoot}/nxxx-yyyy/v1'

variables:

apiRoot:

default: https://example.com

description: apiRoot as defined in clause 4.4.1 of 3GPP TS 29.501

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Next change\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 5.3.7 Server-initiated communication

If an API contains server-initiated communication (see clause 6.2 of 3GPP TS 29.500 [2]), e.g. for notifications as described in clause 4.6.2.3, it should be described as "callbacks" in OpenAPI specification files.

Example:

paths:

/subscriptions:

post:

requestBody:

required: true

content:

application/json:

schema:

type: object

properties:

callbackUrl: # Callback URL

type: string

format: uri

responses:

'201':

description: Success

callbacks:

myNotification: # arbitrary name

'{$request.body#/callbackUrl}': # refers The callback URL in the POST

post:

requestBody: # Contents of the callback message

required: true

content:

application/json:

schema:

$ref: '#/components/schemas/NotificationBody'

responses: # Expected responses to the callback message

'204':

description: xxx

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Next change\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 5.3.16 Security

As indicated in 3GPP TS 33.501 [22] and 3GPP TS 29.500 [2], the access to an 5GC API may be authorized by means of the OAuth2 protocol (see IETF RFC 6749 [n3]), based on local configuration. 5GC APIs thus need to support the OAuth2 protocol.

To reflect this, the OpenAPI specification file of an API shall contain:

- a top-level "security" field, that applies to the overall API unless overridden by a resource/operation-level "security" field, listing as alternatives:

i) "{}" to indicate that usage of security is optional; and

ii) the name of the security scheme for oAuth2, including in the array of scopes the name of the API as the only scope; and optionally

- a resource/operation-level "security" field, that applies to a specific operation on a specific resource, and overrides the top-level "security" field, listing as alternatives:

i) "{}" to indicate that usage of security is optional; and

ii) the name of the security scheme for oAuth2, including in the array of scopes the name of the API as the only scope, and

iii) the name of the security scheme for oAuth2, including in the array of scopes the name of the API and the name of the scope to be used to invoke the specific resource/operation, and

- a "securitySchemes" field in the "components" clause defining a security schema for oAuth2 as follows:

i) to be of type "oauth2"; and

ii) with a "flows" field containing a "clientCredentials" field that contains:

1) a "tokenUrl" field pointing to the Access Token Request service provided by the NRF (see 3GPP TS 29.510 [18]); and

2) a "scopes" field defining all the different scopes applicable to this API, which includes the name of the corresponding API (using the format used within URIs of that API) to be used as the scope required to get access to the overall API, and also including those resource/operation-level scopes to be used as scopes required to invoke a specific operation on a specific resource.

The naming of the resource/operation-level scopes shall consist of the concatenation of the service name, a string representing the resource name or custom operation, and a string indicating the type of access (e.g. read/modify/create), separated by the ":" (colon) character.

Such last string component of the resource/operation-level scope, that represents the type of access for a resource, should comply with the following principles:

- "read": for GET operations for any resource archetype,

- "create": for POST or PUT operations that result in a creation of new resources from a collection or store resource,

- "modify": for PUT, PATCH or DELETE operations that result on an update or deletion of a document resource,

- "invoke": for POST operations that result in the invocation of a custom operation.

Example:

security:

- {}

- oAuth2ClientCredentials:

- nnrf-nfm

paths:

/nf-instances:

get:

security:

- {}

- oAuth2ClientCredentials:

- nnrf-nfm

- oAuth2ClientCredentials:

- nnrf-nfm

- nnrf-nfm:nf-instances:read

parameters:

(...)

responses:

(...)

components:

securitySchemes:

oAuth2ClientCredentials:

type: oauth2

flows:

clientCredentials:

tokenUrl: '{nrfApiRoot}/oauth2/token'

scopes:

nnrf-nfm: Access to the Nnrf\_NFManagement API

nnrf-nfm:nf-instances:read: Read access to the NF Instances (Collection) resource

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Next change\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 6.2 General

The following requirements are intended as general guidance for 3GPP Stage 3 work in order to specify secure protocols and APIs. As such, these guidelines are independent of the specific technology and shall be followed at all times.

- The valid format and range of values (when applicable) for each IE shall be defined unambiguously.

NOTE 1: Explicitly defining format and range of values not only helps to improve the security of a certain implementation, but also allows for reliable interoperability between different protocol implementations. Example: Defining a "lowercase string variable of length 10 and range [a..z]" is much more explicit that just defining a "string of length 10". There are known vulnerabilities such as a denial of service (resulting in the parser converting from a string representing an integer – an attacker can pass in an arbitrarily large integer and trigger an unhandled exception) and such leading to a heap corruption and crash (proof-of-concept available), or potentially remote code execution (no proof-of-concept known). Unicode literals also require special treatment when doing string comparisons to ensure that equivalent strings return true when compared.

- For each message the number of leaf IEs shall not exceed 16K. If a leaf IE is an array of a simple data type, then the whole array shall count as one leaf. If a leaf IE is a data structure or an array of data structures, then it shall be considered a branch, i.e. it shall not be counted as a leaf. The data structure's (branch) attributes determine the number of leaves. For instance, a data structure with e.g. three attributes will count as three leaves.

- The maximum size of the JSON body of any HTTP request/response shall not exceed 16 million octets before compression is applied, if any.

NOTE 2: APIs need to be designed taking care to avoid a too large HTTP payload size for performance reasons.

- The maximum nesting depth of leaves shall not exceed 32. If a leaf IE is an array of a simple data type, then the whole array shall be considered as the first level of nesting. If a leaf IE is a data structure or an array of data structures, then it shall be considered a branch and the first level of nesting. The data structure's attributes (leaves) shall be considered as the second level of nesting. For instance, a data structure with e.g. one attribute-A, which is also a data structure with e.g. one attribute-B, then attribute-B will make the third level of nesting.

NOTE 3: There are resource exhaustion attacks on JSON parsers. Defined maximum numbers of IEs, sizes and nesting depths allow implementations to know an upper bound of required resources. It also allows validation of incoming messages. Recursively processing nested objects leads to stack exhaustion and a denial of service bug.

- For data structures where values are accessible using names (sometimes referred to as keys), e.g. a JSON object, the name shall be unique. The occurrence of the same name (or key) twice within such a structure shall be an error and the message shall be rejected.

NOTE 4: Serialization schemes (e.g. JSON) can leave the handling of repeated names (keys) up to the implementer's discretion. For example, for a repeated name an error can be raised, the pair can be ignored, or the first or last value read can be used, though there is no canonical order in which a parser should treat the data it receives. Failure to adhere to consistent handling rules can lead to vulnerabilities. From a security perspective rejecting objects with repeated names, rather than accepting according to some rule, is the more robust solution, and aids in identification of potentially malicious activity. There are known attacks with specially crafted malicious messages that are designed to confuse implementations of NFs to get fraudulent messages into a PLMN.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Next change\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## C.4 Custom operation

The custom operation archetype can be used to model an unsafe and non-idempotent operation that is not a Create on a collection.

A custom operation does not operate directly on the resource that would be identified by the custom operation URI. Instead, when the custom operation is associated with a resource, the operation is performed on this associated resource. For instance, a custom operation may modify the associated resource in a special way. This associated resource is identified by stripping the suffix string "/<custOpName>" from the custom operation URI template in clause 4.4.2.

When the custom operation is not associated with any resource but with the service, it acts as an executable function with input parameters and returns the result of the executed function in the response body, not modifying any resource.

POST is the only method allowed with a custom operation URI.

The semantic of the custom operation is encoded in the last segment of the URI template in chapter 4.4.2: /<custOpName>.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*The end of changes\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*