# NGSI-10 Context Management RESTful binding

**V1.0**

Contributors: NEC, Orange, NSN, SAP

## ChangeLog

| **Date** | **Versión** | **Author** | **Comment** |
|---|---|---|---|
| December 9th, 2013 | 0.99.1 | Fermín Galán (TID) | Adding changelog table<br>Proper use of "contextSubscription" in section 3.10 and 3.11<br>Change in 3.1.2 regarding value ID mandatoriness, as discussed in https://lists.fi-ware.eu/private/fiware-ngsi/2013-November/000507.html<br>Fix list of Allow verbs in section 3.4.3 |
| December 10th, 2013 | 0.99.2 | Tobias Jacobs (NEC) | Fixed a number of typos |
| December 12th, 2013 | 0.99.3 | Tobias Jacobs (NEC) | Changed support of interactions with **/contextEntities/{EntityID}/attributes** and **/contextEntityTypes/{typeName}/attributes** to optional. |
| December 13th, 2013 | 0.99.4 | Tobias Jacobs (NEC) | Made enforcement of attribute value ids optional. |
| December 13th, 2013 | 1.0 | Tobias Jacobs (NEC) | corrected updateContextElementResponse and appendContextElementResponse |

# 1. Introduction

This document outlines a RESTful binding of the OMA NGSI-10 standard. The specifications are based on Candidate Version 1.0 of the Context Management Interface of the Next Generation Service Interface (NGSI-10).

## 1.1 Design Principles of the Binding

The mapping of NGSI-10 functionality to a resource tree follows a twofold approach. On the one hand, there is one resource per NGSI-10 operation which supports the respective functionality by providing a POST operation. On the other hand, a number of additional resources support convenience functionality. The latter resource structure more closely follows the REST approach and typically supports more operations (GET PUT, POST, and DELETE). The operation scope of the GET operation on these resources can further be limited by a URI parameter.

The convenience functions typically only support a subset of the functionality of the corresponding NGSI operations. Nevertheless, they enable simpler and more straightforward access.

Throughout the document it is assumed that all data structures, as well as the input and output messages of NGSI-10 are represented by xml types. The definition of these types can be found in the xml schema files provided with this document.

## 1.2 Remarks on the data model and operations of NGSI-9 and NGSI-10

This subsection clarifies a number of details about the data model of NGSI-9 and NGSI-10. These details are discussed at this point because they have significant impact on the RESTful binding.

### 1.2.1 Relation between Entities and Context Elements

Entities are the first-class citizens of the data model. Each entity is unambiguously determined by an `EntityId`, which contains an `Id` String and a `Type`. According to the nature of the convenience functions in the RESTful binding, the uniqueness of the context elements is guaranteed by the `Id` element.

An Entity may contain multiple *attribute domains*. An attribute domain is a logical grouping of context information attributes. The `ContextElement` data structure contains context information about one Entity. It is a container for one or more attribute values and respective metadata. If the attribute domain is specified within a `ContextElement,` then it may contain only values and metadata of attribute from that domain. Thus, a `ContextElement` does not necessarily contain *all* available information about an Entity. The set of convenience function resources includes a type of resource for representing an Entity. The URI of that sort of resource is determined by the ID of the represented entity. Note that two Entities having the identical name but differing type will be represented by the same resource.

### 1.2.2 Attribute Value Instances

In typical IoT environments there can be multiple providers delivering values of the same attribute. Therefore, NGSI-10 allows the existence of multiple attributes having the same name. The way to distinguish these attribute values - called *value instances* in this document - is to include an ID in the form of metadata. More specifically, the ID of an attribute value is determined by the unique value of a metadata with `Name` "ID".

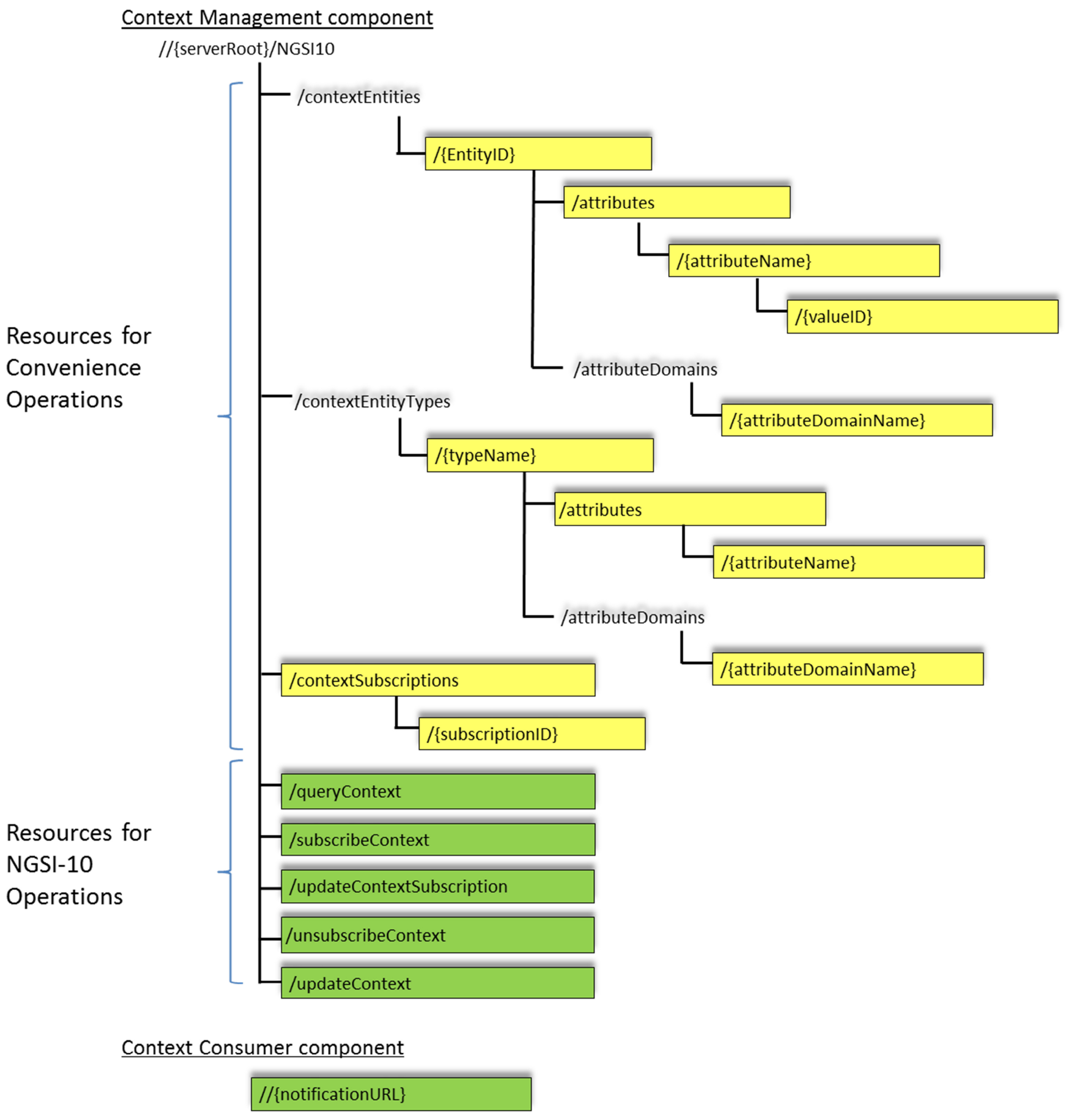
Each attribute value instance is represented by a convenience function resource in the RESTful binding.

In order to also enable simpler systems which do not distinguish between several attribute value instances, it is not mandatory for servers to enforce the usage of attribute value instances.

### 1.2.3 Subscriptions and existence of attribute values

In this document it is assumed that subscriptions can also be successful if they refer to entities and/or attributes the system does not have information about. Notifications are then sent as soon as the information becomes available and the condition for notification is satisfied.

# 2. Resources Summary



## 2.1 Resources on Context Management Component

### 2.1.1 Standard NGSI-10 Operation Resources

The five resources listed in the table below represent the five operations offered by systems that implement the NGSI-10 Context Management role. Each of these resources allows interaction via http POST. All attempts to interact by other verbs shall result in an HTTP error status 405; the server should then also include the ‘Allow: POST’ field in the response.

| Resource | Base URI: http://{serverRoot}/NGSI10 | HTTP verbs |
|---|---|---|
| | | POST |
| Context query resource | /queryContext | Generic queries for context information.The expected request body is an instance of `queryContextRequest`; the response body is an instance of `queryContextResponse`. |
| Subscribe context resource | /subscribeContext | Generic subscriptions for context information. The expected request body is an instance of `subscribeContextRequest`; the response body is an instance of `subscribeContextResponse`. |
| Update context subscription resource | /updateContextSubscription | Generic update of context subscriptions. The expected request body is an instance of `updateContexSubscriptiontRequest`; the response body is an instance of `updateContextSubscriptionResponse`. |
| Unsubscribe context resource | /unsubscribeContext | Generic unsubscribe operations. The expected request body is an instance of `unsubscribeContextRequest`; the response body is an instance of `unsubscribeContextResponse`. |
| Update context resource | /updateContext | Generic context updates. The expected request body is an instance of `updateContextRequest`; the response body is an instance of `updateContextResponse`. |

## 2.1.2 Convenience Operation Resources

The table below gives an overview of the resources for convenience operation and the effects of interacting with them via the standard HTTP verbs GET, PUT, POST, and DELETE. The table cells contain only rough descriptions; the details can be found in Chapter 3.

| Resource | Base URI: http://{serverRoot}/NGSI10 | HTTP verbs | | | |
|---|---|---|---|---|---|
| | | GET | PUT | POST | DELETE |
| Individual context entity | /contextEntities/{EntityID} | Retrieve all available information about the context entity | Replace a number of attribute values | Append context attribute values | Delete all entity information |
| Attribute container of individual context entity | /contextEntities/{EntityID}/attributes | Same functionality as parent resource, but implementation is optional. | | | |
| Attribute of individual context entity | /contextEntities/{EntityID}/attributes/{attributeName} | Retrieve attribute value(s) and associated metadata | (optional) Insert/Replace attribute value[1] | Append context attribute value | Delete all attribute values |
| Specific attribute value of individual context entity | /contextEntities/{EntityID}/attributes/{attributeName}/{valueID} | Retrieve specific attribute value | Replace attribute value | | Delete attribute value |
| Attribute domain of individual context entity | /contextEntities/{EntityID}/attributeDomains/{attributeDomainName} | Retrieve all attribute information belonging to attribute domain | - | - | - |
| Context entity type | /contextEntityTypes/{tyeName} | Retrieve all available information about all context entities having that entity type | - | - | - |
| Attribute container of entity type | /contextEntityTypes/{typeName}/attributes | Same functionality as parent resource, but implementation is optional. | | | |

[1] Opertion not to be supported by servers that enforce attribute value identifiers.

| | | | | | |
|---|---|---|---|---|---|
| Attribute of entity type | /contextEntityTypes/{typeName}/attributes/{attributeName} | Retrieve all attribute values of the context entities of the specific entity type | - | - | - |
| Attribute domain of entity type | /contextEntityTypes/{typeName}/attributeDomains/{attributeDomainName} | For all context entities of the specific type, retrieve the values of all attributes belonging to the attribute domain. | - | - | - |
| Subscriptions container | /contextSubscriptions | - | - | Create a new subscription | - |
| Subscription | /contextSubscriptions/{subscriptionID} | - | Update subscription | - | Cancel subscription |

### 2.1.3 Convenience Operation Scopes

The support of convenience operation scopes is optional.

As the above table shows, many of the convenience resources allow queries for context information via GET. In order to be able to limit the application scope of these queries, a URI parameter for defining operation scopes is introduced. That URI parameter can be added to any GET operation on the convenience resources.

| Parameter name | Parameter value | Description |
|---|---|---|
| scope | ({ScopeType},{ScopeValue}) | Defines the scope of the query, where a scope is an a priori restriction of the search space. |

The syntax for adding parameters to the URI is "`{resourcePath}?{parameterName}={parameterValue}`".

## 2.2 Resource on Context Consumer Component

This section describes the resource that has to be provided by the context consumer in order to receive notifications. All attempts to interact with it by other verbs than POST shall result in an HTTP error status 405; the server should then also include the 'Allow: POST' field in the response.

| Resource | URI | HTTP verbs |
|---|---|---|
| | | POST |
| Notify context resource | //{notificationURI} | Generic notification.The expected request body is an instance of `notifyContextRequest`; the response body is an instance of `notifyContextResponse`. |

# 3. Convenience Function Resources

In this chapter, the resources which enable convenience functionality are described in details. In the context of this document, convenience functions are simple and intuitive interaction methods that are offered for the sake of convenience for application developers. Such methods are redundant in the sense that after removing them the system would still satisfy all functional requirements.

## 3.1 Resource: Individual Context Entity

The resource used is

**http://{serverRoot}/{apiVersion}/contextEntities/{entityID}**

Instances of this resource type are used for retrieving all available information about a certain context entity via GET and modifying the information via PUT, POST, and DELETE.

The {entitityID} part of the resource URI must denote the ID of an individual entity. The usage of regular expressions in {entityID} is supported neither here nor for any subresource.

### 3.1.1 GET

Retrieve all available information about the context entity represented by the resource, including all attribute values and metadata. The response body is an instance of `ContextElementResponse`, and the `ContextElement` field in this response have to satisfy the following property:

- The `ID` field of the `EntityId` entry MUST coincide with the {entityID} part of the resource URI.

### 3.1.2 PUT

This operation allows updating multiple attribute values of a Context Entity and the respective metadata. The request message may contain multiple `ContextAttribute` instances. For each of these instances, the output message SHALL contain an instance of `ContextAttributeResponse`. In case that a submitted `ContextAttribute` contains a value ID in the form of metadata and the contained value ID does not exist in the system, the respective `ContextAttributeResponse` instance SHALL only consist of an error message.

In case all `ContextAttribute` instance update operations finish with success, the `ContextAttributeResponse` items of the `ContextResponse` field in the response can be omitted and the `ErrorCode` field used instead.

If the request contains domain metadata, then every instance of `ContextMetadata` in the request replaces any existing metadata having the same name and type.

#### 3.1.2.1 Input message: updateContextElementRequest

| Part name | Part type | Optional | Description |
|---|---|---|---|
| Attribute Domain Name | xsd:string | Yes | Name of the attribute domain that logically groups together set of Context Information attributes. Examples of attribute domain are: device info (battery level, screen size, …), location info (position, civil address, …). |
| ContextAttributeList | ContextAttribute [0..unbounded] | Yes | List of Context Information attributes.<br>Note: In case of the attributeDomainName is specified all contextAttribute have to belong to the same attributeDomainName. |
| DomainMetadata | ContextMetadata [0..unbounded] | Yes | Metadata common to all attributes of the logical domain (related to the AttributeDomain) |

#### 3.1.2.2 Output message: updateContextElementResponse

The response SHALL contain exactly one of the following:

| Part name | Part type | Optional | Description |
|---|---|---|---|
| ErrorCode | StatusCode | Yes | Error codes |
| ContextResponse | ContextAttributeResponse | Yes | Response containing the indication of the Context Entity and the related statusCode. |

### 3.1.3 POST

This operation allows adding new information like attribute values and metadata about a Context Entity. The new information is added to the existing information. Note that this may result in multiple values of certain attributes, which will be represented in an NGSI `ContextElement` as multiple attributes having the same name.

The request message may contain multiple `ContextAttribute` instances. For each of these instances, the output message SHALL contain an instance of `ContextAttributeResponse` containing the respective statuscode.

In case all `ContextAttribute` append operations finish with success, the `ContextAttributeResponse` items of the `ContextResponse` field in the response can be omitted and the `ErrorCode` field used instead.

#### 3.1.3.1 Input message: appendContextElementRequest

| Part name | Part type | Optional | Description |
|---|---|---|---|
| Attribute Domain Name | xsd:string | Yes | Name of the attribute domain that logically groups together set of Context Information attributes. Examples of attribute domain are: device info (battery level, screen size, …), location info (position, civil address, …). |
| ContextAttributeList | ContextAttribute [0..unbounded] | Yes | List of Context Information attributes. Note: In case of the attributeDomainName is specified all contextAttribute have to belong to the same attributeDomainName. |
| DomainMetadata | ContextMetadata [0..unbounded] | Yes | Metadata common to all attributes of the logical domain (related to the AttributeDomain) |

Each instance of `ContextAttribute` in the input message corresponds to a new attribute value instance.

The ID of this new value instance can be provided in the request message in the form of a `Metadata` instance with `Name` field "ID". In this case, another value instance with the same ID must not yet exist; otherwise the creation of the new value instance SHALL fail.

If the ID of a new value instance is not provided in the request message, the ID SHALL be assigned by the Context Management component and returned to the request issuer in the response message. This ID MUST be different from the ID of any value instance of the same Entity/Attribute combination that already exists.

#### 3.1.3.2 Output message: appendContextElementResponse

The response SHALL contain exactly one of the following:

| Part name | Part type | Optional | Description |
|---|---|---|---|
| ErrorCode | StatusCode | Yes | Error codes |
| ContextResponse | ContextAttributeResponse | Yes | Response containing the indication of the Context Attribute and the related statusCode. |

### 3.1.4 DELETE

This operation allows deleting all information about the Context Entity represented by the resource, i.e., all related attribute values and metadata. The request does not have a body, and the response body is an instance of `StatusCode`.

## 3.2 Resource: Attribute Container of Individual Context Entity

The resource used is

**http://{serverRoot}/{apiVersion}/contextEntities/{entityID}/attributes.**

The support of interactions with this resource is optional. If supported, then the set of allowed interactions with this resource and their effects SHALL be equivalent to the parent resource.

## 3.3 Resource: Attribute of Individual Context Entity

The resource used is

**http://{serverRoot}/{apiVersion}/contextEntities/{entityID}/attributes/{attributeName}**

Instances of this resource are used for retrieving all available information about a specific attribute of the context entity represented by the parent resource via GET and adding information via POST. Furthermore, requests to delete all available attribute values can be issued using DELETE.

### 3.3.1 GET

Retrieve all available information about the attribute represented by this resource, including all available attribute values and the respective metadata. The context entity the attribute belongs to is represented by the parent resource.

#### 3.3.1.1 OutputMessage: contextAttributeResponse

| Element name | Element type | Optional | Description |
|---|---|---|---|
| ContextAttributeList | ContextAttribute [0..unbounded] | No | Context Attribute related to a Context Entity<br>Note: In case of error, this data structure can contain only the Name of the ContextAttribute that cause the error. In case of success, this data structure contains also Value and needed related ContextMetadata (e.g. ID). |
| StatusCode | StatusCode | No | Identifies the status of the requested operation related to this specific ContextAttribute. |

### 3.3.2 PUT

The support of this operation is optional, as it corresponds to updating an attribute value without specification of the attribute value's identifier.

If supported, then the request body is an instance of `updateContextAttributeRequest,` and the response body is an instance of `StatusCode`.

### 3.3.3 POST

This operation allows adding new value instances, possibly including metadata, for the attribute represented by the resource. The request body is an instance of `updateContextAttributeRequest`; the response body is an instance of `StatusCode.`

#### 3.3.3.1 Input message: updateContextAttributeRequest

| Part name | Part type | Optional | Description |
|---|---|---|---|
| Type | xsd:string | Yes | The type of the Context attribute |
| ContextValue | xsd:any | No | The actual value of the Context Information attribute |
| Metadata | ContextMetadata [0.. unbounded] | Yes | Metadata about the Context Information attribute (information valid only for the specific value instance) |

The ID of the new value instance can be provided in the input message in the form of a `Metadata` instance with `Name` field "ID". In this case, another value instance with the same ID must not yet exist; otherwise the creation of the new value instance SHALL fail.

If the ID of a new value instance is not provided in the request message, the ID SHALL be assigned by the Context Management component and returned to the request issuer in the response message. This ID MUST be different from the ID of any value instance of the same Entity/Attribute combination that already exists.

### 3.3.4 DELETE

This operation allows deleting an Attribute of a Context Entity. It deletes all attribute values and respective metadata.

The request does not have a message body. The response body is an instance of `StatusCode`.

## 3.4 Resource: Attribute Value Instance

The resource used is

**http://{serverRoot}/{apiVersion}/contextEntities/{entityID}/attributes/{attributeName}/{valueID}**

Instances of this resource are used for retrieving and modifying a specific instance of an attribute value.

### 3.4.1 GET

Retrieve the attribute value instance provided by this resource together with all metadata. The response body is an instance of `ContextAttributeResponse` containing a single `ContextAttribute`, which MUST satisfy the following properties:

- The `Name` field corresponds to the {attributeName} part of the access URI.
- It contains a piece of `ContextMetadata` having `Name` "ID", `Type` "xsd:string", and `Value` corresponding to the {valueID} part of the access URI.

### 3.4.2 PUT

Update (overwrite) the attribute value instance and associated metadata.

This operation can only be used on resources that represent already existing attribute value instances. It cannot be used for creating new value instances. The latter can instead be achieved by a POST on the parent resource.

The request body is an instance of `updateContextAttributeRequest`. In case this instance contains a `ContextMetadata` instance with `Name` "ID", the `Value` must correspond to the {valueID} part of the access URI, or else the server SHALL return an error.

The response body is an instance of `StatusCode`.

### 3.4.3 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET, PUT, DELETE' field in the response.

### 3.4.4 DELETE

Delete the attribute value instance. The request does not have a body, and the response body is an instance of `StatusCode`.

## 3.5 Resource: Attribute Domain of Individual Context Entity.

The resource used is

**http://{serverRoot}/{apiVersion}/contextEntities/{EntityID}/attributeDomains/{attributeDomainName}**

Instances of this resource are used for retrieving all available information about the attributes in the domain represented by the resource. Other operations shall not be supported on this resource type.

### 3.5.1 GET

Retrieve all available information about the attributes belonging to the domain represented by this resource, including all attribute values and the respective metadata. The context entity the attributes refer to is represented by the parent resource. The response body is an instance of `ContextAttributeResponse`. Each `ContextAttribute` must have a `Name` that belongs to the attribute domain represented by the {attributeDomainName} part of the access URL.

### 3.5.2 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET' field in the response.

### 3.5.3 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET' field in the response.

### 3.5.4 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET' field in the response.

## 3.6 Resource: Context Entity Type

The resource used is

**http://{serverRoot}/{apiVersion}/contextEntityTypes/{typeName}**

Instances of this resource are used for retrieving information about context entities of the type represented by the resource via GET. The scope of the query can be restricted using a query parameter.

Note that in NGSI-9/10, type names have the type `xs:anyURI`. Therefore, the {typeName} part of the resource access URI is a URI by itself. The {typeName} part has to be encoded in the resource URI as defined in RFC 3986, which means for example that any occurrence of the '\' character is encoded as '%2F'.

### 3.6.1 GET

Retrieve all available information about context entities of the type represented by this resource; including all attribute value instances and the respective metadata. The response body is an instance of `queryContextResponse`. The `EntityId` of each `ContextElement` instance inside the response must have a `type` field whose value corresponds to the {typeName} part of the access URI.

### 3.6.2 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET' field in the response.

### 3.6.3 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET' field in the response.

### 3.6.4 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET' field in the response.

## 3.7 Resource: Attribute Container of Context Entity Type

The resource used is

**http://{serverRoot}/{apiVersion}/contextEntityTypes/{typeName}/attributes.**

The support of interactions with this resource is optional. If supported, then the set of allowed interactions with this resource and their effects SHALL be equivalent to the parent resource.

## 3.8 Resource: Attribute of Context Entity Type

The resource used is

**http://{serverRoot}/{apiVersion}/contextEntityTypes/{typeName}/attributes/{attributeName}**

Instances of this resource are used for retrieving all information about a specific attribute of all context entities having a certain type via GET. The scope of the query can be restricted using a query parameter.

### 3.8.1 GET

Retrieve all available information about the attribute represented by the resource, including all attribute value instances and metadata. The context entities of interest are all context entities of the type represented by the {typeName} part of the access URI. Further restrictions can be added using the scope parameter. The returned data type is `queryContextResponse`, and each `ContextElement` instance contained in it must satisfy the following properties:

- If the `EntityId` has a `type` , it SHOULD correspond to the {typeName} fields inside the access URI.
- All `Attribute` instances MUST have a `Name` which corresponds to the {attributeName} part of the access URI.

### 3.8.2 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET' field in the response.

### 3.8.3 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET' field in the response.

### 3.8.4 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET' field in the response.

## 3.9 Resource: Attribute Domain of Context Entity Type

The resource used is

**http://{serverRoot}/{apiVersion}/contextEntityTypes/{typeName}/attributeDomains/{attributeDomainName}**

Instances of this resource are used for retrieving all available information about all those attributes of a certain context entity type that belong to a certain domain. The scope of the query can be restricted using the corresponding query parameter.

### 3.9.1 GET

Retrieve all available information about the set of attributes represented by the domain resource, including value instances and metadata. The context entities of interest are all context entities of the type represented by the parent resource. Further restrictions can be added using the scope parameter (see above). The returned data type is `queryContextResponse`. Each `ContextElement` instance contained in it must satisfy the following properties:

- If the e`ntityId` has a `type` , it SHOULD correspond to the {typeName} fields inside the access URI.
- All `Attribute` instances MUST have a `Name` which belongs to the domain represented by the {attributeDomainName} part of the access URI.

### 3.9.2 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET' field in the response.

### 3.9.3 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET' field in the response.

### 3.9.4 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET' field in the response.

## 3.10 Resource: Subscription Container

The resource used is

**http://{serverRoot}/{apiVersion}/contextSubscriptions**

This is a container resource for subscriptions, where new subscriptions are created via POST. Other operations SHOULD be rejected, as the modification and deletion of subscriptions is done by interaction with the corresponding child resources.

### 3.10.1 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: POST' field in the response.

### 3.10.2 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: POST' field in the response.

### 3.10.3 POST

Create a new subscription. The type of the request and response body is `SubscribeContextRequest` and `SubscribeContextResponse`, respectively. The subscription will be represented by a new child resource, whose URI is determined by the subscription ID inside the response body. The recipient URI of the subscription is to be specified in the `Reference` field of the the request body.

### 3.10.4 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: POST' field in the response.

## 3.11 Resource: Subscription

The resource used is

**http://{serverRoot}/{apiVersion}/contextSubscriptions/{subscriptionID}**

Instances of this resource represent subscriptions. They can be interacted with via PUT and DELETE. Creation of subscriptions is done by interaction with the parent resource.

### 3.11.1 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: PUT/DELETE' field in the response.

### 3.11.2 PUT

This operation updates a subscription. The request and response body is `updateContextSubscriptionRequest` and `updateContextSubscriptionResponse`, respectively. The `subscriptionId` field inside the request message must equal the {subscriptionID} part of the access URI, otherwise an error shall be returned.

### 3.11.3 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: PUT/DELETE' field in the response.

### 3.11.4 DELETE

This operation is used for deleting a subscription, which corresponds to an `unsubscribeContext` operation. The request contains no body, and the response body is of type `UnsubscribeContextResponse`.

# 4. Datastructures

The datastructures used in the RESTful binding defined in this document consist of following data structures:

- All data structures defined in the NGSI-9/10 specification.
- All input and output messages of NGSI-10
- A number of additional data structures that are defined in the subsections of this chapter. These additional structures are only used by the convenience functions, and their purpose is to avoid redundancy of information.

## 4.1 ContextAttributeResponse

| Element name | Element type | Optional | Description |
|---|---|---|---|
| ContextAttributeList | ContextAttribute [0..unbounded] | No | Context Attribute related to a Context Entity<br>Note: In case of error, this data structure can contain only the Name of the ContextAttribute that cause the error. In case of success, this data structure contains also Value and needed related ContextMetadata (e.g. ID). |
| StatusCode | StatusCode | No | Identifies the status of the requested operation related to this specific ContextAttribute. |

## 4.2 updateContextElementRequest

| Part name | Part type | Optional | Description |
|---|---|---|---|
| Attribute Domain Name | xsd:string | Yes | Name of the attribute domain that logically groups together set of Context Information attributes. Examples of attribute domain are: device info (battery level, screen size, …), location info (position, civil address, …). |
| ContextAttributeList | ContextAttribute [0..unbounded] | Yes | List of Context Information attributes.<br>Note: In case of the attributeDomainName is specified all contextAttribute have to belong to the same attributeDomainName. |
| DomainMetadata | ContextMetadata [0..unbounded] | Yes | Metadata common to all attributes of the logical domain (related to the AttributeDomain) |

## 4.3 updateContextElementResponse

| Part name | Part type | Optional | Description |
|---|---|---|---|
| ErrorCode | StatusCode | Yes | Error codes |
| ContextResponseList | ContextAttributeResponse[0..unbounded] | Yes | Response containing the indication of the Context |

## 4.4 appendContextElementRequest

| Part name | Part type | Optional | Description |
|---|---|---|---|
| Attribute Domain Name | xsd:string | Yes | Name of the attribute domain that logically groups together set of Context Information attributes. Examples of attribute domain are: device info (battery level, screen size, …), location info (position, civil address, …). |
| ContextAttributeList | ContextAttribute [0..unbounded] | Yes | List of Context Information attributes.<br>Note: In case of the attributeDomainName is specified all contextAttribute have to belong to the same attributeDomainName. |
| DomainMetadata | ContextMetadata [0..unbounded] | Yes | Metadata common to all attributes of the logical domain (related to the AttributeDomain) |

## 4.5 updateContextAttributeRequest

| Part name | Part type | Optional | Description |
|---|---|---|---|
| Type | xsd:string | Yes | The type of the Context attribute |
| ContextValue | xsd:any | No | The actual value of the Context Information attribute |
| Metadata | ContextMetadata [0.. unbounded] | Yes | Metadata about the Context Information attribute (information valid only for the specific value instance) |

## 4.6 appendContextElementResponse

| Part name | Part type | Optional | Description |
|---|---|---|---|
| ErrorCode | StatusCode | Yes | Error codes |
| ContextResponseList | ContextAttributeResponse[ 0..unbounded] | Yes | Response containing the indication of the Context Attribute and the related statusCode. |