SDMX RESTful API – Extension to support more HTTP verbs for data/metadata maintenance

Applying the HTTP methods to the SDMX REST API for data maintenance

The business logic of inserting, updating or deleting data in SDMX is specified in the technical specifications (SDMX_2-1_SECTION_3A_PART_IV), under paragraph 4 (SPECIAL DATA FUNCTIONS); see below:

```
1118 4 SPECIAL DATA FUNCTIONS
1119 4.1 Updates

1120 Both the generic and the structure-specific data messages allow for incremental
updating of data. This
1121 purpose is noted in the action for the data set, which is either inherited from the
header of the data message
1122 or explicitly stated at the data set level.
1123
1124 A dataset with an action of Append is assumed to be an incremental update. This
means that one the
1125 information provided explicitly in the message should be altered. Any data attribute
or observation value
1126 that is to be changed must be provided. However, the absence of an observation value
or a data attribute
1127 at any level does not imply deletion; instead it is simply implied that the value is
to remain unchanged.
1128 Therefore, it is valid and acceptable to send a data message with an action of
Append which contains only
1129 a Series elements with attribute values. In this case, the values for the attributes
will be updated. Note that
1130 it is not permissible to update data attributes using partial keys (outside of those
associated with defined
1131 groups). In order to update an attribute, a full key must always be provided even if
the message format
1132 does not require this.

1133 4.2 Deletes

1134 Both the generic and the structure-specific data messages allow for incremental
deletion of data. This
1135 purpose is noted in the action for the data set, which is either inherited from the
header of the data message
1136 or explicitly stated at the data set level.
1137
1138 A dataset with an action of Delete is assumed to be an incremental deletion. The
deletion is assumed to
1139 take place of the lowest level of detail provided in the message. For example, if a
delete message is sent
1140 with only a data set element, the entire data set will be deleted. On the other
hand, if that data set contains
1141 a data attribute, only that data attribute value will be deleted. This same dynamic
continues through the
1142 data set hierarchy. A data set containing only a series with no data attributes or
observations will result in
```

```
1143 that entire series (all observations and data attributes) being deleted. If the
series contains data attributes,
1144 only the supplied data attributes for that series will be deleted. Finally, if a
series contains observations,
1145 then only the specified observations will be deleted. If an entire observation is to
be deleted (value and data
1146 attributes), only the observation dimension should be provided. If only the
observation value or particular
1147 data attributes are to be deleted, then these should be specified for the
observation. Note that a group can
1148 only be used to delete the data attributes associated with it. Although the format
might not require it, a full
1149 key must be provided to delete a series or observation. It is not permissible to
wild card a key in order to
1150 delete more than one series or observation. Finally, to delete a data attribute or
observation value it is
1151 recommended that the value to be deleted be supplied; however, it is only required
that any valid value be
1152 provided.
```

Thus, inserting/creating and updating are handled in the same manner, i.e. by the action `Append`. For deleting data, as well as data attributes, action `Delete` must be used. The actions `Information` and `Replace` are not foreseen by the standard to be used for data maintenance.

From a RESTful point of view, for updating or deleting data, the HTTP verbs `POST` or `DELETE` may be used, respectively. Due to the fact that all the information for what to do with an SDMX Dataset resided within the latter, i.e. the actions to be taken, `POST` could be used to support all data maintenance (updating and deleting). Nevertheless, for the sake of keeping the RESTful semantics close to those of the Dataset actions, it is proposed to allow `POST`ing for Datasets with the SDMX action `Append` and `DELETE`ing for Datasets with the SDMX action `Delete`. Thus, multiple SDMX actions are not foreseen, for now.

CREATE & UPDATE

Creating or inserting data via SDMX means using the `Append` action in the SDMX `Header` or `Dataset` element, according to the SDMX specification. This allows for incremental updating of data.

The HTTP verb `POST` is used either for submitting new data or for updating existing data. The idea is that data may be submitted under resource `/data` or `/data/{dataidentifier}`, where `{dataidentifier}` may be a Dataflow (or a DSD or ProvisionAgreement in SDMX 3.0) indicating the structure of the submitted data. The body of the message submitted must be an SDMX Dataset (structure specific or generic) with value `Append` in field `/Header/DataSetAction/` or `/Dataset/@action`. The latter overrides the former. When action is different that `Append`, as foreseen by SDMX, the request will fail.

Client

In order to insert or update data, the client:

- MAY set the `Accept` header to indicate the preferred response format;
- MUST set the `Content-type` header according to the format of the submitted Dataset(s);
- MUST include in the request body, one or more Datasets in the SDMX format indicated in the `Content-type` header and comply with:
  - any DSD, under resource `/data/`;
  - the DSD identified by the `{dataidentifier}`, under resource `/data/{dataidentifier}`, e.g. `/data/ECB/EXR/1.0/` for the DSD identified by the Dataflow `ECB:EXR(1.0)`.

Server

In response to a data insertion/updating, the server:

- MUST return `200` upon successful appending (or `207` for partial success);
- MUST return a `DataResponse` message describing the result of the action (see Data response message below), according to the `Accept` header, or the default, if the `Accept` type is not supported;
- MUST set the `Content-type` according to the response format;

Response

The `DataResponse` message must be returned in any case (success, partial success, failure). The details of the message are explained in section Data response message, below.

Especially when different results occur on the Artefacts (e.g. partial success), the following should occur:

- Return a multi-status return code (like `207`);
- Return a `JSON` or `XML` response message with the details of the result;

The table below details the conditions for generating the proper HTTP response code. The column `Structure found` refers to matching the structure (as identfied by the `{dataidentifier}`, or the header of the submitted message) by the server. The column `Dataset match` refers to matching the contained Dataset to the structure.

Method	Structure found	Dataset match	Response Code	comment
POST	T	T	`204`	Success; The normal case of data update
POST	-	-	`206`	Partial content; only some of the data were found (and updated) in the store
POST	-	-	`207`	Mixed case of successful Dataset action combined with any of the following error codes
POST	I	I	`400`	Bad request
POST	I	I	`401`	Unauthorised; invalid or missing credentials
POST	T	T	`403`	Forbidden; credentials OK, but not authorised to perform action
POST	F	I	`404`	Structure resource was not found
POST	T	F	`409`	Conflict; Dataset identifier does not match structure identifier
POST	I	I	`415`	Unsupported media type
POST	I	I	`422`	Unprocessable entity; Dataset not valid according to structure

T: True, F: False, I: Irrelevant, -: Not applicable

DELETE

Deleting data via SDMX means using the `Delete` action in the SDMX `Header` or `Dataset` element, according to the SDMX specification. This allows for incremental deletion of data.

Given the way SDMX handles data deletion, i.e. by submitting a Dataset message, the approach used for data update could be used as-is; thus, `POST` may be used. Nevertheless, in order to follow the REST principles for using HTTP verbs to manage resources, the `DELETE` verb is proposed to be used. In addition to that, not allowing to delete with a `POST` message may provide safety that a `Delete` message will not appear by mistake in a set of `Update` messages.

The HTTP verb `DELETE` is used for deleting data. The idea is that an SDMX Dataset including a `Delete` action may be submitted under resource `/data` or `/data/{dataidentifier}`, where `{dataidentifier}` may be a Dataflow (or a DSD or ProvisionAgreement in SDMX 3.0) indicating the structure of the data to be deleted. The body of the message submitted must be an SDMX Dataset (structure specific or generic) with value `Delete` in field `/Header/DataSetAction/` or `/Dataset/@action`. The latter overrides the former. When action is different that `Delete`, as foreseen by SDMX, the request will fail.

Client

In order to delete data, the client:

- MAY set the `Accept` header to indicate the preferred response format;
- MUST set the `Content-type` header according to the format of the submitted Dataset(s);
- MUST include in the request body, one or more Datasets in the SDMX format indicated in the `Content-type` header and comply with:
  - any DSD, under resource `/data/`;
  - the DSD identified by the `{dataidentifier}`, under resource `/data/{dataidentifier}`, e.g. `/data/ECB/EXR/1.0/` for the DSD identified by the Dataflow `ECB:EXR(1.0)`.

Server

In response to a data deletion, the server:

- MUST return `200` upon successful deletion (or `207` for partial success);
- MUST return a `DataResponse` message describing the result of the action (see Data response message below), according to the `Accept` header, or the default, if the `Accept` type is not supported;
- MUST set the `Content-type` according to the response format;

Response

The `DataResponse` message must be returned in any case (success, partial success, failure). The details of the message are explained in section Data response message, below.

Especially when different results occur on the Artefacts (e.g. partial success), the following should occur:

- Return a multi-status return code (like `207`);
- Return a `JSON` or `XML` response message with the details of the result;

The table below details the conditions for generating the proper HTTP response code. The column `Structure found` refers to matching the structure (as identfied by the `{dataidentifier}`, or the header of the submitted message) by

the server. The column `Dataset match` refers to matching the contained Dataset to the structure.

Method	Structure found	Dataset match	Response Code	comment
DELETE	T	T	`204`	Deletion succesful
DELETE	-	-	`206`	Partial content; only some of the data were found (and deleted) in the store
DELETE	-	-	`207`	Mixed case of successful Dataset action combined with any of the following error codes
DELETE	I	I	`400`	Bad request
DELETE	I	I	`401`	Unauthorised; invalid or missing credentials
DELETE	T	T	`403`	Forbidden; credentials OK, but not authorised to perform action
DELETE	F	I	`404`	Structure resource was not found
DELETE	T	F	`409`	Conflict; Dataset identifier does not match structure identifier
DELETE	I	I	`415`	Unsupported media type
DELETE	I	I	`422`	Unprocessable entity; Dataset not valid according to structure

T: True, F: False, I: Irrelevant, -: Not applicable

Data response message

(Description of the proposed data response message in XML and/or JSON)

TBD by TF3

Applying the HTTP methods to the SDMX REST API for Structural maintenance

CREATE

Artefact (Structure)

Submitting SDMX Artefacts in bulk, either of the same or of different types, is achieved with a `POST` method. Creating new Artefact(s) may be issued by:

- `POST` one or more Maintainable Artefacts under the proper resource type, e.g. for Codelists: `/structure/codelist/`
- `POST` one or more Maintainable Artefacts under the abstract structure resource type, e.g. `/structure/`

Client

In order to create Artefacts, the client:

- MAY set the `Accept` header to indicate the preferred response format;
- MUST set the `Content-type` header according to the format of the submitted Artefact;
- MUST include in the request body, one or more Maintainable Artefacts in the SDMX format indicated in the `Content-type` header and of the SDMX type indicated in the resource, i.e.:
  - For `POST`:
    - any set of Maintainable Artefacts under resource `/structure/`
    - a set of specific type of Maintainable Artefacts under the corresponding resource type, e.g. for Codelists: `/structure/codelist/`

Server

In response to an Artefact creation, the server:

- MUST return `201` upon successful creation (or `207` for partial success);
- MUST return a `SubmitStructureResponse` message with the result of the action(s), according to the `Accept` header, or the default, if the `Accept` type is not supported (currently available only in SDMX-ML 2.1);
- MUST set the `Content-type` according to the returned format;
- MAY set the `Location` header to point to the created/primary resource/Artefact (only one instance is allowed);

```
RFC7231
(https://tools.ietf.org/html/rfc7231)

6.3.2. 201 Created
(https://tools.ietf.org/html/rfc7231#section-6.3.2)
[...]
   The primary resource created by the request is identified
   by either a Location header field in the response or, if no Location
   field is received, by the effective request URI.

4.3.3. POST
(https://tools.ietf.org/html/rfc7231#section-4.3.3)
  [...]
   If one or more resources has been created on the origin server as a
   result of successfully processing a POST request, the origin server
   SHOULD send a 201 (Created) response containing a Location header
   field that provides an identifier for the primary resource created
   (Section 7.1.2) and a representation that describes the status of the
   request while referring to the new resource(s).
  [...]
```

Response

The `SubmitStructureResponse` message must be returned in any case (success, partial success, failure). In SDMX 2.1 this is defined as part of the `RegistryInterface` messages. The details of the message are explained in section Structure response message, below.

Especially when different results occur on the Artefacts (e.g. partial success), the following should occur:

- Return a multi-status return code (like `207`);
- Return a `JSON` or `XML` message with the details of the result (currently available only in SDMX-ML 2.1);

Method	Exists	Is Final	Is Referenced	Refs exist/ provided	Response Code
POST	F	-	-	T	`201` (successful) or `207` (partially successful)
POST	F	-	-	F	`409` failed references

T: True, F: False, I: Irrelevant, -: Not applicable

UPDATE

Replace Artefacts

Following the current SDMX practices, updating (replacing) means providing the new version of any Maintainable Artefact. According to RFC7231, `PUT` is the proper way to update a resource, as identified by the URL, but `POST` may also be used in case more than one resources need to be updated.

The `POST` method may be used like this:

- under `/structure` for different types of Maintainable Artefacts, i.e. an SDMX Structure message;
- under `/structure/{maintainable}` for Maintainable Artefacts of type `{maintainable}`, i.e. an SDMX Structure message including only a specific type of Structures; e.g. an SDMX message with Dataflows, under `/structure/dataflow`;

The `PUT` method may be used like this:

- under `/structure/{maintainable}/{identifier}` for the Maintainable Artefact of type `{maintainable}`, identified by `{identifier}`, i.e. an SDMX DSD message with `ESTAT:NA_MAIN(1.1)` under `/structure/datastructure/ESTAT/NA_MAIN/1.1`

The result, following the current practices of updates in SDMX, is to completely replace the identified Artefact.

Client

In order to update Artefact(s), the client:

- MAY set the `Accept` header to indicate the preferred response format;
- MUST set the `Content-type` header according to the format of the submitted Artefact;
- MUST include in the request body, one or more Maintainable Artefacts in the SDMX format indicated in the `Content-type` header and of the SDMX type indicated in the resource, i.e.:
  - any set of Maintainable Artefacts under resource `/structure/`

- one or more (only `POST`) Maintainable Artefacts of the specific type under the corresponding resource type, e.g. Dataflows under `/structure/dataflow/`

Server

In response to Artefact(s) update, the server:

- MUST respond with `200` in case of successful update;
- MUST return a `SubmitStructureResponse` message with the result of the action(s), according to the `Accept` header, or the default, if the `Accept` type is not supported (currently available only in SDMX-ML 2.1);
- MUST set the `Content-type` according to the returned format;
- MUST respond with `422` in case of resource type mismatch, i.e. the resource type identified in the URL does not match to the Artefact type(s) of the included SDMX Artefact(s).

Response

The `SubmitStructureResponse` message must be returned in any case (success, failure). In SDMX 2.1 this is defined as part of the `RegistryInterface` messages. The details of the message are explained in section Structure response message, below.

The following matrix summarises the returned `HTTP` response codes.

Method	Exists	Is Final	Is Referenced	References exist/provided	Return Code
PUT/POST	T	F	F	T	`200` (successful) or `207` (partially successful)
PUT/POST	T	F	F	F	`409`
PUT/POST	T	F	T	T	`200` if update is possible - `409` if references would break
PUT/POST	T	T	-	-	`409` (in case of structural changes)
PUT/POST	-	-	-	-	`422` see section Server

T: True, F: False, I: Irrelevant, -: Not applicable

Partially update Item Schemes

In the case of Item Schemes, it is possible to submit a partial Item Scheme, including the Items that must be updated. The usage of `POST` and `PUT` is the same as in the case of full replacement. The only difference is that the submitted Items Schemes to be partially updated (i.e. not completely replaced) must inlcude the flag `isPartial="true"`.

In this case, any Item included in the submitted partial Item Scheme:

- fully replaces the Item, if it existed in the stored Item Scheme; the Item stays in the same position;
- is added as a new Item, if it did not exist

- in a selected position, if it is submitted relatively to existing Items;
- else, at the end of the Item Scheme.

For more details on nested Items (like Categories), please refer to section Nested Items below.

In addition to the Items, the Item Scheme details are also updated accordingly, i.e. the names, description, annotations and any other attributes. Names and Descriptions are replaced if they exist for the submitted language (i.e. the value of their `lang` attribute), or simply added, if new. Annotations and other Item Scheme attributes are fully replaced.

Client

In order to partially update an Item Scheme, the client:

- MAY set the `Accept` header to indicate the preferred response format;
- MUST set the `Content-type` header according to the format of the submitted Item Scheme(s);
- MUST include the flag `isPartial="true"` in the submitted Item Scheme(s);
- MUST include in the request body:
  - For `PUT`, one Item Scheme in the SDMX format indicated in the `Content-type` header and of the Item Scheme type indicated in the resource, e.g. a Concept Scheme under resource `/conceptscheme`;
  - For `POST`, one or more Item Schemes in the SDMX format indicated in the `Content-type` header and
    - of the Item Scheme type indicated in the resource, in case of a specific Item Scheme resource, e.g. a Codelist under resource `/structure/codelist/`
    - of any Item Scheme type, in case of the `/structure` resource.

Server

In response to an Item Scheme partial update, the server:

- MUST respond with `200` in case of successful update;
- MUST return a `SubmitStructureResponse` message with the result of the action, according to the `Accept` header, or the default, if the `Accept` type is not supported (currently available only in SDMX-ML 2.1);
- MUST set the `Content-type` according to the returned format;
- MUST respond with `422` in case of resource type mismatch, i.e. the artefact identified in the URL does not match either to the resource type or identification of the included SDMX Artefact.

```
11.2.  422 Unprocessable Entity
(https://tools.ietf.org/html/rfc4918#section-11.2)
   The 422 (Unprocessable Entity) status code means the server
   understands the content type of the request entity (hence a
   415(Unsupported Media Type) status code is inappropriate), and the
   syntax of the request entity is correct (thus a 400 (Bad Request)
   status code is inappropriate) but was unable to process the contained
   instructions.  For example, this error condition may occur if an XML
   request body contains well-formed (i.e., syntactically correct), but
   semantically erroneous, XML instructions.
```

Response

The `SubmitStructureResponse` message must be returned in any case (success, partial success, failure). In SDMX 2.1 this is defined as part of the `RegistryInterface` messages. The details of the message are explained in section Structure response message, below.

The following matrix summarises the returned `HTTP` response codes.

Method	Exists	Is Final	Is Referenced	Refs exist/ provided	Response Code
PUT/POST	T	F	F	T	`200` (successful) or `207` (partially successful)
PUT/POST	T	F	F	F	`409`
PUT/POST	T	F	T	T	`200` if update is possible - `409` if references would break
PUT/POST	T	T	-	-	`409` (in case of structural changes)
PUT/POST	-	-	-	-	`422` see section Server

T: True, F: False, I: Irrelevant, -: Not applicable

DELETE

Always concerns one Maintainable Artefact or one Item. For example:

- A fully identified Maintainable Artefact, e.g. `/structure/codelist/SDMX/CL_FREQ/1.0`
- A fully identified Item, e.g. `/structure/codelist/SDMX/CL_FREQ/1.0/M`

In case the deleted Item was acting as a parent to other Item(s), then the server should make sure that:

- in case of flat Item Schemes the children become orphans, i.e. their parent is removed;
- in case of nested Item Schemes, all children are also deleted.

Client

In order to delete an Artefact, the client:

- MAY set the `Accept` header to indicate the preferred response format;
- MUST fully identify exactly one Maintainable Artefact or one Item, by means of the proper URL;

Server

In response to an Artefact deletion, the server:

- MUST respond with `200` in case of successful deletion;
- MUST return a `SubmitStructureResponse` message with the result of the action, according to the `Accept` header, or the default, if the `Accept` type is not supported (currently available only in SDMX-ML 2.1);
- MUST respond with `404` if the resource was not found;

Response

The `SubmitStructureResponse` message must be returned in any case (success, failure). In SDMX 2.1 this is defined

as part of the `RegistryInterface` messages. The details of the message are explained in section Structure response message, below.

The following matrix summarises the returned `HTTP` response codes.

Method	Exists	Is Final	Is Referenced	Refs exist/ provided	Response Code
DELETE	F	-	-	-	`404`
DELETE	T	F	F	I	`200`
DELETE	T	T	I	I	`409`
DELETE	T	F	T	I	`409`

T: True, F: False, I: Irrelevant, -: Not applicable

Structure response message

The `SubmitStructureResponse` message must be returned in any case (success, partial success, failure). In SDMX 2.1 this is defined as part of the `RegistryInterface` messages. This message includes the following information per submitted Artefact:

- The `action`, e.g. `Append`, `Replace` or `Delete` (`Information` is also available)
- A reference to a specific Maintainable Artefact
- A status message with the result of the action, which contains:
  - The status, e.g. `Success`, `Failure` or `Warning`
  - One or more message texts to explain the result (we need only one per Artefact), which in turn contains:
    - A code (could be the HTTP code)
    - A multilingual text message

An example is shown below:

```
<reg:SubmissionResult>
  <reg:SubmittedStructure action="Append"> <!-- Append|Delete|Replace -->
    <reg:MaintainableObject>
      <Ref agencyID="SDMX" id="CL_FREQ" version="1.0"
        package="codelist" class="Codelist" />
    </reg:MaintainableObject>
  </reg:SubmittedStructure>
  <reg:StatusMessage status="Success"> <!-- Success|Failure|Warning -->
    <reg:MessageText code="204"> <!-- Could be the HTTP code 200, 201, 204, ... -->
      <com:Text xml:lang="en">Codelist successfully deleted</com:Text>
      <com:Text xml:lang="en">Codelist supprimé avec succès</com:Text>
    </reg:MessageText>
  </reg:StatusMessage>
</reg:SubmissionResult>
```

Especially when different results occur on the Artefacts (e.g. partial success), the following should occur:

- Return a multi-status return code (like `207`);

- Return a JSON/XML message with the results details (currently available only in SDMX-ML 2.1);

In the case of a multi-status response, the `SubmitStructureResponse` message will include the corresponding code per Artefact, e.g.:

```
<reg:SubmissionResult>
  <reg:SubmittedStructure action="Append">
    <reg:MaintainableObject>
      <Ref agencyID="SDMX" id="CODELIST" version="1.0"
        package="codelist" class="Codelist"/>
    </reg:MaintainableObject>
  </reg:SubmittedStructure>
  <reg:StatusMessage status="Warning">
    <reg:MessageText code="201">
      <com:Text xml:lang="en">Successfully created Codelist</com:Text>
    </reg:MessageText>
  </reg:StatusMessage>
</reg:SubmissionResult>

<reg:SubmissionResult>
  <reg:SubmittedStructure action="Delete"> <!-- Append|Delete|Replace -->
    <reg:MaintainableObject>
      <Ref agencyID="SDMX" id="CL_FREQ" version="1.0"
        package="codelist" class="Codelist" />
    </reg:MaintainableObject>
  </reg:SubmittedStructure>
  <reg:StatusMessage status="Success"> <!-- Success|Failure|Warning -->
    <reg:MessageText code="204"> <!-- Could be the HTTP code 200, 201, 204, ... -->
      <com:Text xml:lang="en">Codelist successfully deleted</com:Text>
      <com:Text xml:lang="en">Codelist supprimé avec succès</com:Text>
    </reg:MessageText>
  </reg:StatusMessage>
</reg:SubmissionResult>
```

Nested Items

This section aims at explaining the particularities of nested Items for a subset of the available Item Schemes, namely:

- CategoryScheme (Category)
- ReportingTaxonomy (ReportingCategory)

In all the above cases, Items may contain other Items in a tree-like hierarchy. As a result, the resource for an Item within such an hierarchy need to include the full path of that Item in order to exactly identify it. For example, for the following Category Scheme (excerpt of SDMX:STAT_SUBJECT_MATTER(1.0) from the Global SDMX Registry):

```
<str:CategoryScheme agencyID="SDMX" id="STAT_SUBJECT_MATTER" version="1.0">
        <com:Name xml:lang="en">SDMX Statistical Subject-Matter Domains</com:Name>
        <str:Category id="DEMO_SOCIAL_STAT">
                <com:Name xml:lang="en">Demographic and social statistics</com:Name>
        </str:Category>
        <str:Category id="ECO_STAT">
                <com:Name xml:lang="en">Economic statistics</com:Name>
                <str:Category id="MACROECO_STAT">
                        <com:Name xml:lang="en">Macroeconomic statistics</com:Name>
                </str:Category>
                <str:Category id="SECTORAL_STAT">
                        <com:Name xml:lang="en">Sectoral statistics</com:Name>
```

```
                        <str:Category id="AGRI_FOREST_FISH">
                                <com:Name xml:lang="en">Agriculture, forestry,
fisheries</com:Name>
                        </str:Category>
                        <str:Category id="ENERGY">
                                <com:Name xml:lang="en">Energy</com:Name>
                        </str:Category>
                </str:Category>
                <str:Category id="GOV_FINANCE_PUBLIC_SECTOR">
                        <com:Name xml:lang="en">Government finance, fiscal and public sector
        statistics</com:Name>
                </str:Category>
        </str:Category>
        <str:Category id="ENVIRONMENT_MULTIDOMAIN_STAT">
                <com:Name xml:lang="en">Environment and multi-domain statistics</com:Name>
        </str:Category>
</str:CategoryScheme>
```

In order to get Item/Category `ENERGY` we need to request the following resource: `categoryscheme/SDMX/CAT/1.0/ECO_STAT.SECTORAL_STAT.ENERGY` Instead of the identifier of the Item, the full path of identifiers that lead to that Item are required, i.e.: `ECO_STAT` -> `SECTORAL_STAT` -> `ENERGY`.

Similarly, when trying to delete that Item (Category) the same resource must be used. The behaviour of the server, then, should be that the Category and all its children (if any) are deleted. In the above example, deleting the Category `ECO_STAT.SECTORAL_STAT` would result into deleting also `ECO_STAT.SECTORAL_STAT.ENERGY` and `ECO_STAT.SECTORAL_STAT.AGRI_FOREST_FISH`.

When submitting a nested Item Scheme for partial update (using `PUT` or `POST` with `isPartial="true"`) then all root level Categories appearing in the body are going to fully replace existing root level Categories. This means that all Categories comprising any root Category are going to be replaced by the new Categories submitted. Let's assume submitting this partial Category Scheme:

```
<str:CategoryScheme agencyID="SDMX" id="STAT_SUBJECT_MATTER" version="1.0" isPartial="true">
        <com:Name xml:lang="en">SDMX Statistical Subject-Matter Domains</com:Name>
        <str:Category id="ECO_STAT">
                <com:Name xml:lang="en">Economic statistics</com:Name>
                <str:Category id="MACROECO_STAT">
                        <com:Name xml:lang="en">Macroeconomic statistics</com:Name>
                </str:Category>
  </str:Category>
</str:CategoryScheme>
```

When applied to the initial Category Scheme, this update would mean that Category `ECO_STAT` only has one child Category, i.e. the resulting Category Scheme would be:

```
<str:CategoryScheme agencyID="SDMX" id="STAT_SUBJECT_MATTER" version="1.0">
        <com:Name xml:lang="en">SDMX Statistical Subject-Matter Domains</com:Name>
        <str:Category id="DEMO_SOCIAL_STAT">
                <com:Name xml:lang="en">Demographic and social statistics</com:Name>
        </str:Category>
        <str:Category id="ECO_STAT">
                <com:Name xml:lang="en">Economic statistics</com:Name>
                <str:Category id="MACROECO_STAT">
                        <com:Name xml:lang="en">Macroeconomic statistics</com:Name>
                </str:Category>
        </str:Category>
        <str:Category id="ENVIRONMENT_MULTIDOMAIN_STAT">
```

```
                <com:Name xml:lang="en">Environment and multi-domain statistics</com:Name>
        </str:Category>
</str:CategoryScheme>
```

Moreover, if we wanted to add a new Category under `SECTORAL_STAT` in the initial Category Scheme, we would need to submit the whole new tree of `ECO_STAT`, i.e.:

```
<str:CategoryScheme agencyID="SDMX" id="STAT_SUBJECT_MATTER" version="1.0">
        <com:Name xml:lang="en">SDMX Statistical Subject-Matter Domains</com:Name>
        <str:Category id="ECO_STAT">
                <com:Name xml:lang="en">Economic statistics</com:Name>
                <str:Category id="MACROECO_STAT">
                        <com:Name xml:lang="en">Macroeconomic statistics</com:Name>
                </str:Category>
                <str:Category id="SECTORAL_STAT">
                        <com:Name xml:lang="en">Sectoral statistics</com:Name>
                        <str:Category id="AGRI_FOREST_FISH">
                                <com:Name xml:lang="en">Agriculture, forestry,
fisheries</com:Name>
                        </str:Category>
      <str:Category id="NEW_SECTORAL_CATEGORY">
        <com:Name xml:lang="en">New</com:Name>
      </str:Category>
                        <str:Category id="ENERGY">
                                <com:Name xml:lang="en">Energy</com:Name>
                        </str:Category>
                </str:Category>
                <str:Category id="GOV_FINANCE_PUBLIC_SECTOR">
                        <com:Name xml:lang="en">Government finance, fiscal and public sector
        statistics</com:Name>
                </str:Category>
        </str:Category>
</str:CategoryScheme>
```

Summary of HTTP response codes

Method	Exists	Is Final	Is Referenced	Refs exist/ provided	Response Code
POST	F	-	-	T	`201` (successful) or `207` (partially successful)
POST	F	-	-	F	`409` failed references
PUT/POST	T	F	F	T	`200` (successful) or `207` (partially successful)
PUT/POST	T	F	F	F	`409`
PUT/POST	T	F	T	T	`200` if update is possible - `409` if references would break
PUT/POST	T	T	-	-	`409` (in case of structural changes)

PUT/POST	-	-	-	-	`422` see section Server
DELETE	F	-	-	-	`404`
DELETE	T	F	F	I	`200`
DELETE	T	T	I	I	`409`
DELETE	T	F	T	I	`409`

T: True, F: False, I: Irrelevant, -: Not applicable

Examples

Difference between replace and partially update

To explain the updating semantics, the difference between using replacing and partially updating the same SDMX Artefact shall be illustrated in the following example. For the sake of simplicity, a Codelist will be utilised, i.e. let's assume:

```
<str:Codelist agencyID="SDMX" id="CL_DECIMALS" version="1.0">
   <com:Name>Code list for Decimals (DECIMALS)</com:Name>
   <com:Description xml:lang="en">It provides a list of values showing the
     number of decimal digits used in the data.</com:Description>
   <str:Code id="0">
      <com:Name>Zero</com:Name>
   </str:Code>
   <str:Code id="1">
      <com:Name>One</com:Name>
   </str:Code>
   <str:Code id="2">
      <com:Name>Two</com:Name>
   </str:Code>
</str:Codelist>
```

Let's assume that we `PUT` the following Codelist, under `/codelist/SDMX/CL_DECIMALS/1.0`, or `POST` it under `/codelist/`:

```
<str:Codelist agencyID="SDMX" id="CL_DECIMALS" version="1.0">
   <com:Name>Code list for Decimals (DECIMALS)</com:Name>
   <com:Description xml:lang="en">It provides a list of values showing the
     number of decimal digits used in the data.</com:Description>
   <str:Code id="0">
      <com:Name>No decimal</com:Name>
   </str:Code>
   <str:Code id="1">
      <com:Name>One</com:Name>
   </str:Code>
 </str:Codelist>
```

This will result into replacing the original Codelist with the one submitted. Hence, in the new Codelist only two Codes will exist; the first one (Code `0`) with an updated name.

If, instead, we added the flag `isPartial="true"` in the above Codelist, i.e.:

```
<str:Codelist agencyID="SDMX" id="CL_DECIMALS" version="1.0" isPartial="true">
   <com:Name>Code list for Decimals (DECIMALS)</com:Name>
   <com:Description xml:lang="en">It provides a list of values showing the
     number of decimal digits used in the data.</com:Description>
   <str:Code id="0">
      <com:Name>No decimal</com:Name>
   </str:Code>
   <str:Code id="1">
      <com:Name>One</com:Name>
   </str:Code>
 </str:Codelist>
```

Then the result would be a bit different. The new Codelist will still have three Codes, but the first one (Code `0`) would have an updated name, i.e.:

```
<str:Codelist agencyID="SDMX" id="CL_DECIMALS" version="1.0">
   <com:Name>Code list for Decimals (DECIMALS)</com:Name>
   <com:Description xml:lang="en">It provides a list of values showing the
     number of decimal digits used in the data.</com:Description>
   <str:Code id="0">
      <com:Name>No decimal</com:Name>
   </str:Code>
   <str:Code id="1">
      <com:Name>One</com:Name>
   </str:Code>
   <str:Code id="2">
      <com:Name>Two</com:Name>
   </str:Code>
</str:Codelist>
```