



2 **SDMX CANDIDATE TECHNICAL STANDARD**

3 **SDMX-JSON DATA MESSAGE: SYNTAX**
4 **AND DOCUMENTATION**

5 Version 0.8.0

6 April 2014

7 **Status of this Document**

8 *This section describes the status of this document at the time of its publication. Other*
9 *documents may supersede this document.*

10 This is a SDMX Candidate Technical Standard for SDMX-JSON Data Message. It is
11 made available for review by the SDMX user community and the public.

12 The public review period for this document extends until 31st October 2014 in order
13 to allow time for implementation. Please send your comments to the SDMX Technical
14 Working Group (SDMX-TWG) twg@sdmx.org. Each email message should contain only
15 one comment. All feedback is welcome.

16 Publication as a Candidate Technical Standard does not imply endorsement by the
17 SDMX Sponsors. This is a draft document and may be updated, replaced or obsoleted
18 by other documents at any time.

19 This document has been produced by the SDMX-TWG. The authors of this document
20 are the members of the SDMX-TWG. Different parts of this specification have different
21 editors.

22

24 Contents

25	1 Introduction	5
26	2 Field Guide to SDMX-JSON Objects	6
27	2.1 Message	6
28	2.1.1 header	6
29	2.1.2 structure	6
30	2.1.3 dataSets	7
31	2.1.4 errors	7
32	2.1.4.1 code	8
33	2.1.4.2 message	8
34	2.2 header	8
35	2.2.1 id	8
36	2.2.2 test	8
37	2.2.3 prepared	9
38	2.2.4 sender	9
39	2.2.4.1 contact	9
40	2.2.5 receiver	10
41	2.2.6 request	10
42	2.2.6.1 uri	10
43	2.3 structure	10
44	2.3.1 uri	11
45	2.3.2 name	11
46	2.3.3 description	11
47	2.3.4 dimensions	11
48	2.3.5 attributes	12
49	2.3.6 dataSet	12
50	2.3.7 series	12
51	2.3.8 observation	12
52	2.3.8.1 component	13
53	2.3.8.1.1 id	13
54	2.3.8.1.2 name	13
55	2.3.8.1.3 description	13
56	2.3.8.1.4 keyPosition	13
57	2.3.8.1.5 role	14
58	2.3.8.1.6 default	14
59	2.3.8.1.7 values	14

60	2.3.8.1.8	Component value	14
61	2.3.9	Annotations	15
62	2.3.9.1	title	16
63	2.3.9.2	type	16
64	2.3.9.3	uri	16
65	2.3.9.4	text	16
66	2.3.9.5	id	16
67	2.4	dataSets	17
68	2.4.1	action	17
69	2.4.2	reportingBegin	18
70	2.4.3	reportingEnd	18
71	2.4.4	validFrom	18
72	2.4.5	validTo	18
73	2.4.6	publicationYear	18
74	2.4.7	publicationPeriod	18
75	2.4.8	annotations	19
76	2.4.9	attributes	19
77	2.4.10	observations	19
78	2.4.11	series	19
79	2.4.11.1	annotations	20
80	2.4.11.2	attributes	20
81	2.4.11.3	observations	20
82	3	Handling component values	21
83	4	Security Considerations	26
84	5	References	27
85	5.1	Normative References	27
86	Appendix A.	Full Example with Comments	28

1 Introduction

88 Let's first start with a brief introduction of the SDMX information model.

89 In order to make sense of some statistical data, we need to know the concepts associated
90 with them. For example, on its own the figure 1.2953 is pretty meaningless, but if we
91 know that this is an exchange rate for the US dollar against the euro on 23 November
92 2006, it starts making more sense.

93 There are two types of concepts: dimensions and attributes. Dimensions, when combined,
94 allow to uniquely identify statistical data. Attributes on the other hand do not help
95 identifying statistical data, but they add useful information (like the unit of measure or
96 the number of decimals). Dimensions and attributes are known as "components".

97 The measurement of some phenomenon (e.g. the figure 1.2953 mentioned above) is known
98 as an "observation" in SDMX. Observations are grouped together into a "data set".
99 However, there can also be an intermediate grouping. For example, all exchange rates
100 for the US dollar against the euro can be measured on a daily basis and these measures
101 can then be grouped together, in a so-called "time series". Similarly, you can group
102 a collection of observations made at the same point in time, in a "cross-section" (for
103 example, the values of the US dollar, the Japanese yen and the Swiss franc against the
104 euro at a particular date). Of course, these intermediate groupings are entirely optional
105 and you may simply decide to have a flat list of observations in your data set.

106 The SDMX information model is much richer than this limited introduction, however
107 the above should be sufficient to understand the sdmx-json format. For additional
108 information, please refer to the [SDMX documentation](#).

109 Samples, tools and other SDMX-JSON resources are available in the public Github repos-
110 itory <https://github.com/sdmx-twg/sdmx-prototype-json/tree/master/draft-sdmx-json>.

111 Before we start, let's clarify a few more things about this guide:

- 112 • New fields may be introduced in later versions. Therefore consuming applications
113 should tolerate the addition of new fields with ease.
- 114 • The ordering of fields in objects is undefined. The fields may appear in any order
115 and consuming applications should not rely on any specific ordering. It is safe to
116 consider a nulled field and the absence of a field as the same thing.
- 117 • Not all fields appear in all contexts. For example response with error messages may
118 not contain fields for data, dimensions and attributes.

119 2 Field Guide to SDMX-JSON Objects

120 2.1 Message

121 Message is the top level object and it contains the data as well as the metadata needed
122 to interpret those data. Example:

```
123 {  
124   "header": {  
125     # header fields #  
126   },  
127   "structure": {  
128     # structure objects #  
129   },  
130   "dataSets": [  
131     # data set objects #  
132   ],  
133   "errors": [  
134     # Error messages #  
135   ]  
136 }
```

137 2.1.1 header

138 *Object nullable.* *Header* contains basic technical information about the message, such as
139 when it was prepared and how has sent it. Example:

```
140 "header": {  
141   "id": "b1804c51-1ee3-45a9-bb75-795cd4e06489",  
142   "prepared": "2012-05-04T03:30:00"  
143 }
```

144 2.1.2 structure

145 *Object nullable.* *Structure* contains the information needed to interpret the data available
146 in the message, such as the list of concepts used. Example:

```

147 "structure": {
148     "uri": "http://sdw-ws.ecb.europa.eu/dataflow/ECB/EXR/1.0",
149     "dimensions": {
150         # dimensions object #
151     },
152     "attributes": {
153         # attributes object #
154     }
155     "annotations": [
156         # annotation objects #
157     ]
158 }

```

159 2.1.3 dataSets

160 *Array nullable.* *DataSets* field is an array of *DataSet* objects. That's where the data (i.e.:
161 the observations) will be. Example:

```

162 "dataSets": [
163     {
164         "action": "Information",
165         "observations": {
166             # observation objects #
167         }
168     }
169 ]

```

170 In typical cases, the file will contain only one data set. However, in some cases, such as
171 when retrieving, from an SDMX 2.1 web service, what has changed in the data source
172 since in particular point in time, the web service might return more than one data set.

173 2.1.4 errors

174 *Array nullable.* RESTful web services indicates errors using the HTTP status codes. In
175 addition, whenever appropriate, the error messages can also be returned using this error
176 field. Error is an array of error messages. Example:

```

177 "errors": [
178     {
179         "code": 150,
180         "message": "Invalid number of dimensions in the key parameter"
181     }
182 ]

```

183 **2.1.4.1 code**

184 *number*. Provides a code number for the error message. Code numbers are defined in the
185 SDMX 2.1 Web Services Guidelines. Example:

```
186 "code": 150
```

187 **2.1.4.2 message**

188 *string*. Provides the error message. Example:

```
189 "message": "Invalid number of dimensions in the key parameter"
```

190 **2.2 header**

191 Header contains basic information about the message, such as when it was prepared and
192 who has sent it. Example:

```
193 "header": {  
194   "id": "b1804c51-1ee3-45a9-bb75-795cd4e06489",  
195   "prepared": "2013-01-03T12:54:12",  
196   "sender": {  
197     "id": "SDMX"  
198   }  
199 }
```

200 **2.2.1 id**

201 *String*. Unique string that identifies the message for further references. Example:

```
202 "id": "TEC00034"
```

203 **2.2.2 test**

204 *Boolean nullable*. Indicates whether the message is for test purposes or not. False for
205 normal messages. Example:

```
206 "test": false
```


207 **2.2.3 prepared**

208 *String*. A timestamp indicating when the message was prepared. Values must follow the
209 ISO 8601 syntax for combined dates and times, including time zone. Example:

```
210 "prepared": "2012-05-04T03:30:00Z"
```

211 **2.2.4 sender**

212 *Object*. Information about the party that is transmitting the message. Sender contains
213 the following fields:

- 214 • *id* - *String*. A unique identifier of the party.
- 215 • *name* - *String nullable*. A human-readable name of the sender.
- 216 • *contact* - *Array nullable*. A collection of contact details.

217 Example:

```
218 "sender": {  
219   "id": "ECB",  
220   "name": "European Central Bank"  
221   "contact": [  
222     # contact details #  
223   ]  
224 }
```

225 **2.2.4.1 contact**

226 *Array nullable*. Information on how the party can be contacted. Each object in the
227 collection may contain the following field:

- 228 • *name* - *String*. The contact's name.
- 229 • *department* - *String nullable*. The organisational structure for the contact.
- 230 • *role* - *String nullable*. The responsibility of the contact.
- 231 • *telephone* - *Array nullable*. An array of telephone numbers for the contact.
- 232 • *fax* - *Array nullable*. An array of fax numbers for the contact person.
- 233 • *uri* - *Array nullable*. An array of uris. Each uri holds an information URL for the
234 contact.
- 235 • *email* - *Array nullable*. An array of email addresses for the contact person.

236 Example:

```
237 "contact": [  
238   {  
239     "name": "Statistics hotline",  
240     "email": [ "statistics@xyz.org" ]  
241   }  
242 ]
```

243 2.2.5 receiver

244 *Object nullable.* Information about the party that is receiving the message. This can be
245 useful if the WS requires authentication. Receiver contains the same fields as sender (see
246 above):

247 Example:

```
248 "receiver": {  
249   "id": "SDMX"  
250 }
```

251 2.2.6 request

252 *Object nullable.* Information about the client request.

253 Example:

```
254 "request": {  
255   "uri": "https://registry.sdmx.org/ws/rest/data/EXR/D.USD.EUR.SPOO.A"  
256 }
```

257 2.2.6.1 uri

258 *String nullable.* SDMX RESTful API URL requested by the client. Example:

```
259 "uri": "https://registry.sdmx.org/ws/rest/data/EXR/D.USD.EUR.SPOO.A"
```

260 2.3 structure

261 *Object nullable.* Provides the structural metadata necessary to interpret the data contained
262 in the message. It tells you which are the components (dimensions and attributes) used
263 in the message and also describes to which level in the hierarchy (data set, series,
264 observations) these components are attached.

265 Example:

```

266 "structure": {
267     "uri": "http://sdw-ws.ecb.europa.eu/dataflow/ECB/EXR/1.0",
268     "dimensions": {
269         # dimensions object #
270     },
271     "attributes": {
272         # attributes object #
273     },
274     "annotations": {
275         # annotations object #
276     }
277 }

```

278 **2.3.1 uri**

279 *String nullable.* A link to an SDMX 2.1 web service resource where additional information
280 regarding the structure is available (e.g. link to the dataflow). Example:

```

281 "uri": "http://sdw-ws.ecb.europa.eu/dataflow/ECB/EXR/1.0"

```

282 **2.3.2 name**

283 *String nullable.* Data flow name. Example:

```

284 "name": "Sample dataflow"

```

285 **2.3.3 description**

286 *String nullable.* Descriptio of the data flow. Example:

```

287 "description": "Data flow description."

```

288 **2.3.4 dimensions**

289 *Object.* Describes the dimensions used in the message as well as the levels in the hierarchy
290 (data set, series, observations) to which these dimensions are attached. Example:

```

291 "dimensions": {
292     "dataSet": [
293         # Component objects #

```

```
294     ],
295     "series": [
296         # Component objects #
297     ],
298     "observation": [
299         # Component object #
300     ]
301 }
```

302 **2.3.5 attributes**

303 *Object.* Describes the attributes used in the message as well as the levels in the hierarchy
304 (data set, series, observations) to which these attributes are attached. Example:

```
305 "attributes": {
306     "dataSet": [
307         # Component objects #
308     ],
309     "series": [
310         # Component objects #
311     ],
312     "observation": [
313         # Component objects #
314     ]
315 }
```

316 **2.3.6 dataSet**

317 *Array nullable.* Optional array to be provided if components (dimensions or attributes)
318 are attached to the data set level.

319 **2.3.7 series**

320 *Array nullable.* Optional array to be provided if components (dimensions or attributes)
321 are attached to the series level.

322 **2.3.8 observation**

323 *Array nullable.* Optional array to be provided if components (dimensions or attributes)
324 are attached to the observation level.

325 **2.3.8.1 component**

326 A component represents a dimension or an attribute used in the message. It contains
327 basic information about the component (such as its name and id) as well as the list of
328 values used in the message for this particular component. Example:

```
329 {  
330   "id": "FREQ",  
331   "name": "Frequency",  
332   "keyPosition": 0,  
333   "values": [  
334     {  
335       # value object #  
336     }  
337   ]  
338 }
```

339 Each of the components may contain the following fields

340 **2.3.8.1.1 id** *String*. Identifier for the component. Example:

```
341 "id": "FREQ"
```

342 **2.3.8.1.2 name** *String*. Name provides a human-readable name for the component.
343 Example:

```
344 "name": "Frequency"
```

345 **2.3.8.1.3 description** *String nullable*. Provides a description for the component. Ex-
346 ample:

```
347 "description": "The time interval at which observations occur over a given time period."
```

348 **2.3.8.1.4 keyPosition** *Number nullable*. Indicates the position of the dimension in the
349 key, starting at 0. This field should not be supplied for attributes and it may also be
350 omitted for dimensions. This field could be used to build the “key” parameter string
351 (i.e. D.USD.EUR.SP00.A) for data queries, whenever the order of the dimensions cannot
352 easily be derived from the structural metadata information available in the data message.
353 Example:

```
354 "keyPosition": 0
```

355 **2.3.8.1.5 role** *String nullable*. Defines the component role(s), if any. Roles are rep-
356 resented by the id of a concept defined as [SDMX cross-domain concept](#). Several of
357 the concepts defined as SDMX cross-domain concepts are useful for data visualisation,
358 such as for example, the series title, the unit of measure, the number of decimals to be
359 displayed, the reference area (e.g. when using maps), the period of time to which the
360 measured observation refers, etc. It is recommended to identify any component that can
361 be useful for data visualisation purposes by using the appropriate SDMX cross-domain
362 concept as role. Example:

```
363 "role": "TITLE"
```

364 **2.3.8.1.6 default** *String or Number nullable*. Defines a default value for the component
365 (valid for attributes only!). If no value is provided in the data part of the message then
366 this value applies. Example:

```
367 "default": "A"
```

368 **2.3.8.1.7 values** *Array*. Array of values for the component. Example:

```
369 "values": [  
370   {  
371     "id": "M",  
372     "name": "Monthly"  
373   }  
374 ]
```

375 **2.3.8.1.8 Component value** *Object nullable*. A particular value for a component in a
376 message. Example:

```
377 {  
378   "id": "M",  
379   "name": "Monthly"  
380 }
```

381 **2.3.8.1.8.1 id** *String*. Unique identifier for a value. Example:

```
382 "id": "A"
```

383 **2.3.8.1.8.2 name** *String*. Human-readable name for a value. Example:

```
384 "name": "Missing value; data cannot exist"
```

385 **2.3.8.1.8.3 description** *String nullable*. Description provides a human-readable de-
386 scription of the value. The description is typically longer than the text provided for the
387 name field. Example:

```
388 "description": "Description for missing value."
```

389 **2.3.8.1.8.4 start and end fields** *String nullable*. Start and end are instances of time
390 that define the actual Gregorian calendar period covered by the values for the time
391 dimension. The algorithm for computing start and end fields for any supported reporting
392 period is defined in the SDMX Technical Notes.

393 These fields should be used only when the component value represents one of the values
394 for the time dimension.

395 Values are considered as inclusive both for the start field and the end field. Values must
396 follow the ISO 8601 syntax for combined dates and times, including time zone.

397 Example:

```
398 {  
399     "id": "2010",  
400     "name": "2010",  
401     "start": "2010-01-01T00:00Z",  
402     "end": "2010-12-31T23:59:59Z"  
403 }
```

404 These fields are useful for visualisation tools, when selecting the appropriate point in
405 time for the time axis. Statistical data, can be collected, for example, at the beginning,
406 the middle or the end of the period, or can represent the average of observations through
407 the period. Based on this information and using the start and end fields, it is easy to get
408 or calculate the desired point in time to be used for the time axis.

409 **2.3.9 Annotations**

410 *Array nullable*. Provides a list of annotation objects. Annotations can be attached to
411 data sets, series and observations.

```
412 "annotations": [  
413     {  
414         "title": "Sample annotation",  
415         "uri": "http://sample.org/annotations/74747"  
416     }  
417 ]
```

418 Each annotation object contains the following optional information:

419 **2.3.9.1 title**

420 *string nullable*. Provides a title for the annotation. Example:

421 "title": "Sample annotation"

422 **2.3.9.2 type**

423 *string nullable*. Type is used to distinguish between annotations designed to support
424 various uses. The types are not enumerated, and these can be freely specified by the
425 creator of the annotations. The definitions and use of annotation types should be
426 documented by their creator. Example:

427 "type": "reference"

428 **2.3.9.3 uri**

429 *string nullable*. URI - typically a URL - which points to an external resource which may
430 contain or supplement the annotation. If a specific behavior is desired, an annotation
431 type should be defined which specifies the use of this field more exactly.

432 "uri": "http://sample.org/annotations/74747"

433 **2.3.9.4 text**

434 *string nullable*. Contains the text of the annotation.

435 "text": "Sample annotation text"

436 **2.3.9.5 id**

437 *string nullable*. ID provides a non-standard identification of an annotation. It can be
438 used to disambiguate annotations. Example:

439 "id": "74747"

440 2.4 dataSets

441 An array of data set objects. Example:

```
442 "dataSets": [  
443   {  
444     "action": "Information",  
445     "series": {  
446       # series object #  
447     }  
448   }  
449 ]
```

450 There are between 2 and 3 levels in a data set object, depending on the way the data in
451 the message is organized.

452 A data set may contain a flat list of observations. In this scenario, we have 2 levels in
453 the data part of the message: the data set level and the observation level.

454 A data set may also organize observations in logical groups called series. These groups
455 can represent time series or cross-sections. In this scenario, we have 3 levels in the data
456 part of the message: the data set level, the series level and the observation level.

457 Dimensions and attributes may be attached to any of these 3 levels.

458 In case the data set is a flat list of observations, observations will be found directly
459 under a data set object. In case the data set represents time series or cross sections, the
460 observations will be found under the series elements.

461 2.4.1 action

462 *String nullable.* Action provides a list of actions, describing the intention of the data
463 transmission from the sender's side. Default value is Information

- 464 • Append - this is an incremental update for an existing data set or the provision
465 of new data or documentation (attribute values) formerly absent. If any of the
466 supplied data or metadata are already present, it will not replace these data.
- 467 • Replace - data are to be replaced, and may also include additional data to be
468 appended.
- 469 • Delete - data are to be deleted.
- 470 • Information - data are being exchanged for informational purposes only, and not
471 meant to update a system.

472 Example:

```
473 "action": "Information"
```

474 **2.4.2 reportingBegin**

475 *String nullable*. The start of the time period covered by the message. Example:

```
476 "reportingBegin": "2012-05-04"
```

477 **2.4.3 reportingEnd**

478 *String nullable*. The end of the time period covered by the message. Example:

```
479 "reportingEnd": "2012-06-01"
```

480 **2.4.4 validFrom**

481 *String nullable*. The validFrom indicates the inclusive start time indicating the validity
482 of the information in the data.

```
483 "validFrom": "2012-01-01T10:00:00Z"
```

484 **2.4.5 validTo**

485 *String nullable*. The validTo indicates the inclusive end time indicating the validity of
486 the information in the data.

```
487 "validTo": "2013-01-01T10:00:00Z"
```

488 **2.4.6 publicationYear**

489 *String nullable*. The publicationYear holds the ISO 8601 four-digit year.

```
490 "publicationYear": "2005"
```

491 **2.4.7 publicationPeriod**

492 *String nullable*. The publicationPeriod specifies the period of publication of the data in
493 terms of whatever provisioning agreements might be in force (i.e., "2005-Q1" if that is
494 the time of publication for a data set published on a quarterly basis).

```
495 "publicationPeriod": "2005-Q1"
```

496 2.4.8 annotations

497 *Array nullable.* An optional array of annotation indices for the dataset. Indices refer
498 back to the array of annotations in the structure field. Example:

```
499 "annotations": [ 3, 42 ]
```

500 2.4.9 attributes

501 *Array nullable.* Collection of **attributes values** attached to the data set level. This is
502 typically the case when a particular attribute always has the same value for the data
503 available in the data message. In order to avoid repetition, that value can simply be
504 attached at the data set level. Example:

```
505 "attributes": [ 0, null, 0 ]
```

506 2.4.10 observations

507 *Object nullable.* Collection of **observations** directly attached to a data set. This is the
508 case when a data set represents a flat collection of observations. In case the observations
509 are organised into logical groups (time series or cross-sections), use the **series element**
510 instead. Example:

```
511 "observations": {  
512   "0:1:0": [ 105.6, 0, 1 ],  
513   "0:1:1": [ 105,9 ]  
514 }
```

515 2.4.11 series

516 *Object nullable.* A collection of series. Each series object is identified by a key (a string
517 representing the index of each dimension value, separated by a colon) and contains the
518 observation values and associated attributes, when the observations contained in the data
519 set are used into logical groups (time series or cross-sections). This element must **not** be
520 used in case the data set represents a flat list of observations. Example:

```
521 {  
522   "annotations": [],  
523   "attributes": [ 0, 1 ],  
524   "observations": {  
525     "0": [ 105.6, null, null ],
```

```
526     "1": [ 105.9 ],
527     "2": [ 106.9 ],
528     "3": [ 107.3, 0 ]
529   }
530 }
```

531 **2.4.11.1 annotations**

532 *Array nullable.* An optional array of annotation indices for the series. Indices refer back
533 to the array of annotations in the structure field. Example:

```
534 "annotations": [ 3, 42 ]
```

535 **2.4.11.2 attributes**

536 *Array nullable.* Collection of attributes values. Each value is an index to the *values* array
537 in the respective *Attribute* object. Example:

```
538 "attributes": [ 0, 1 ]
```

539 For information on how to handle the attribute values, see the section dedicated to
540 handling component values.

541 **2.4.11.3 observations**

542 *Object nullable.* An object of observation values. Each observation value is an array of
543 one or more values.

```
544 "observations": {
545   "0": [ 105.6, null, null ],
546   "1": [ 105.9 ],
547   "2": [ 106.9 ],
548   "3": [ 107.3, 0 ]
549 }
```

550 The keys in the observation object are the index values of the observation level dimensions.
551 It's one for time series and cross-sections, but there will be more than one when the data
552 set represents a flat list of observations.

553 The first value in the observation array is the observation value. The data type for
554 observation value is *Number*. Data type for a reported missing observation value is a *null*.

555 Elements after the observation value are values for the observation level attributes.

3 Handling component values

557 Let's say that the following message needs to be processed:

```
{
  "header": {
    "id": "62b5f19d-f1c9-495d-8446-a3661ed24753",
    "prepared": "2012-11-29T08:40:26Z",
    "sender": {
      "id": "ECB",
      "name": "European Central Bank"
    }
  },
  "structure": {
    "id": "ECB_EXR_WEB",
    "uri": "http://sdw-ws.ecb.europa.eu/dataflow/ECB/EXR/1.0",
    "dimensions": {
      "dataSet": [
        {
          "id": "FREQ",
          "name": "Frequency",
          "keyPosition": 0,
          "values": [
            {
              "id": "D",
              "name": "Daily"
            }
          ]
        }
      ]
    },
    {
      "id": "CURRENCY_DENOM",
      "name": "Currency denominator",
      "keyPosition": 2,
      "values": [
        {
          "id": "EUR",
          "name": "Euro"
        }
      ]
    }
  ]
}
```

```

    },
    {
      "id": "EXR_TYPE",
      "name": "Exchange rate type",
      "keyPosition": 3,
      "values": [
        {
          "id": "SPO0",
          "name": "Spot rate"
        }
      ]
    },
    {
      "id": "EXR_SUFFIX",
      "name": "Series variation - EXR context",
      "keyPosition": 4,
      "values": [
        {
          "id": "A",
          "name": "Average or standardised measure"
        }
      ]
    }
  ],
  "series": [
    {
      "id": "CURRENCY",
      "name": "Currency",
      "keyPosition": 1,
      "values": [
        {
          "id": "NZD",
          "name": "New Zealand dollar"
        }, {
          "id": "RUB",
          "name": "Russian rouble"
        }
      ]
    }
  ],
  "observation": [
    {
      "id": "TIME_PERIOD",
      "name": "Time period or range",

```

```

        "values": [
            {
                "id": "2013-01-18",
                "name": "2013-01-18",
                "start": "2013-01-18T00:00:00Z",
                "end": "2013-01-18T23:59:59Z"
            }, {
                "id": "2013-01-21",
                "name": "2013-01-21",
                "start": "2013-01-21T00:00:00Z",
                "end": "2013-01-21T23:59:59Z"
            }
        ]
    },
    ],
    "attributes": {
        "dataSet": [],
        "series": [
            {
                "id": "TITLE",
                "name": "Series title",
                "values": [
                    {
                        "name": "New zealand dollar (NZD)"
                    }, {
                        "name": "Russian rouble (RUB)"
                    }
                ]
            }
        ]
    },
    ],
    "observation": [
        {
            "id": "OBS_STATUS",
            "name": "Observation status",
            "values": [
                {
                    "id": "A",
                    "name": "Normal value"
                }
            ]
        }
    ]
},

```

```

    "dataSets": [
      {
        "action": "Information",
        "series": {
          "0": {
            "attributes": [0],
            "observations": {
              "0": [1.5931, 0],
              "1": [1.5925, 0]
            }
          },
          "1": {
            "attributes": [1],
            "observations": {
              "0": [40.3426, 0],
              "1": [40.3000, 0]
            }
          }
        }
      }
    ]
  }
}

```

558 There is one data set in the message, and it contains two series.

```

"0": {
  "attributes": [0],
  "observations": {
    "0": [1.5931, 0],
    "1": [1.5925, 0]
  }
},
"1": {
  "attributes": [1],
  "observations": {
    "0": [40.3426, 0],
    "1": [40.3000, 0]
  }
}

```

559 The structure.dimensions field tells us that, out of the 6 dimensions, 4 have the same
 560 value for the 2 series and are therefore attached to the data set level.

561 We see that, for the first series, we get the value 0:

562 "0": { ... }

563 From the structure information, we know that CURRENCY is the series dimension.

```
"series": [  
  {  
    "id": "CURRENCY",  
    "name": "Currency",  
    "keyPosition": 1,  
    "values": [  
      {  
        "id": "NZD",  
        "name": "New Zealand dollar"  
      }, {  
        "id": "RUB",  
        "name": "Russian rouble"  
      }  
    ]  
  }  
]
```

564 The value 0 identified previously is the index of the item in the collection of values for
565 this component. In this case, the dimension value is therefore “New Zealand dollar”.

566 The same logic applies when mapping attributes.

567 **4 Security Considerations**

568 This document defines a response format for SDMX RESTful Web Services in JSON
569 format and it raises no new security considerations. SDXM Web Services Guidelines
570 includes the security considerations associated with its usage.

571 **5 References**

572 **5.1 Normative References**

573 [iso8601] *ISO 8601:2004 Data elements and interchange formats — Information inter-*
574 *change — Representation of dates and times*, ISO, 2004

575 [json] *The JSON Data Interchange Format*, Standard ECMA-404, ECMA International,
576 October 2013

577 [sdmxim21] *Section 2: Information Model: UML Conceptual Design*, SDMX Standards
578 2.1, SDMX, April 2011

579 [sdmxws21] *Section 7: Guidelines for the Use of Web Services*, SDMX Standards 2.1,
580 SDMX, April 2011

Appendix A. Full Example with Comments

```

{
  "header": {

    # dynamically generated GUI
    "id": "62b5f19d-f1c9-495d-8446-a3661ed24753",

    # extraction time from db (=now in SQL query), include timezone!
    "prepared": "2012-11-29T08:40:26Z",

    # optional with default false
    "test": false,

    "sender": {
      "id": "ECB",
      "name": "European Central Bank",
      "contact": [
        {
          "name": "Statistics hotline",
          "department": "Statistics Department",
          "role": "helpdesk",
          "telephone": ["+00-00-99999"],
          "fax": ["+00-00-88888"],
          "uri": ["http://www.xyz.org"],
          "email": ["statistics@xyz.org"]
        }
      ]
    },

    # receiver is optional, info from user record if authenticated
    "receiver": {
      "id": "SDMX",
      "name": "SDMX",
      "contact": [
        {
          "name": "name",
          "department": "department",

```

```

        "role": "role",
        "telephone": ["telephone"],
        "fax": ["fax"],
        "uri": ["uri"],
        "email": ["sdmx@xyz.org"]
    }
]
},

"request": {
    # include complete URL as used by the client
    "uri": "http://www.myorg.org/ws/data/ECB_ICP1/M.PT+FI.N.000000+071100.4.INX?
startPeriod=2009-01&dimensionAtObservation=AllDimensions"
}
},
"errors": [
    {
        "code": 123,
        "message": "Invalid number of dimensions in parameter key"
    }
],
"structure": {
    # resolvable uri to dataflow
    "uri": "http://sdw-ws.ecb.europa.eu/dataflow/ECB/EXR/1.0",

    "name": "dataflow name",
    "description": "dataflow description",
    "dimensions": {

        # dataSet is used only if grouping of dimensions with single values
        "dataSet": [
            {
                "id": "FREQ",
                "name": "Frequency",
                "description": "Description for the dimension",

                # 0-based position of dimension in key in user request url
                "keyPosition": 0,

                # restricted list of dimension and attribute roles (time, frequency,
                # geo, unit, scalefactor, referenceperiod, ...)
                "role": "frequency",

                "values": [

```

```

    {
      "id": "D",
      "name": "Daily"
    }
  ],
}, {
  "id": "CURRENCY_DENOM",
  "name": "Currency denominator",
  "description": "Description for the dimension",
  "keyPosition": 3,
  "values": [
    {
      "id": "EUR",
      "name": "Euro"
    }
  ]
}, {
  "id": "EXR_TYPE",
  "name": "Exchange rate type",
  "description": "Description for the dimension",
  "keyPosition": 4,
  "values": [
    {
      "id": "SP00",
      "name": "Spot rate"
    }
  ]
}, {
  "id": "EXR_SUFFIX",
  "name": "Series variation - EXR context",
  "description": "Description for the dimension",
  "keyPosition": 5,
  "values": [
    {
      "id": "A",
      "name": "Average or standardised measure for given frequency"
    }
  ]
}
],

# only if dimensionAtObservation <> allDimensions
"series": [
  {

```

```

    "id": "CURRENCY",
    "name": "Currency",
    "description": "Description for the dimension",
    "keyPosition": 2,
    "role": "unit",
    "values": [
      {
        "id": "NZD",
        "name": "New Zealand dollar"
      }, {
        "id": "RUB",
        "name": "Russian rouble",
      }
    ]
  }
],

```

only for dimensions used at observation level

```

"observation": [
  {
    "id": "TIME_PERIOD",
    "name": "Time period or range",
    "description": "Description for the dimension",
    "role": "time",
    "values": [
      {
        "id": "2013-01-18",
        "name": "2013-01-18",
        "start": "2013-01-18T00:00:00Z",
        "end": "2013-01-18T23:59:59Z"
      },
      {
        "id": "2013-01-21",
        "name": "2013-01-21",
        "start": "2013-01-21T00:00:00Z",
        "end": "2013-01-21T23:59:59Z"
      }
    ]
  }
]
},
"attributes": {

```

only for attributes returned at dataset level

```

"dataSet": [],

# only for attributes returned at series level
"series": [
  {
    "id": "ID",
    "name": "Attribute name",
    "description": "Description for the attribute",
    "role": null,
    "default": null,

    # inclusion of attachment level and its format to be decided
    # e.g. "attachment": [ true, true, true, true, true, true, false ],

    "values": [
      {
        # id property is optional to allow for uncoded attributes
        "id": null,
        "name": "New Zealand dollar (NZD)"
      },
      {
        "id": null,
        "name": "Russian rouble (RUB)"
      }
    ]
  }
],
"observation": [
  {
    "id": "OBS_STATUS",
    "name": "Observation status",
    "description": "Description for the attribute",
    "role": null,

    # optional
    "default": "A",

    "values": [
      # a null attribute can be used to shorten the message by
      # using 0 index later in message
      null,

      {
        "id": "A",

```



```

        "name": "Normal value",
        "description": "Normal value"
    }
  ]
}
],
},
"annotations": [
  {
    "title": "AnnotationTitle provides a title for the annotation.",
    "type": "AnnotationType is used to distinguish between annotations
designed to support various uses.",
    "uri": "http://www.myorg.org/ws/uri/for/this/annotation",
    "text": "AnnotationText holds a language-specific string containing
the text of the annotation.",
    "id": "The id attribute provides a non-standard identification of an
annotation. It can be used to disambiguate annotations."
  }
]
},
"dataSets": [
  {
    "action": "Information",

    # optional first time period in returned message
    "reportingBegin": "2012-05-04",

    # optional last time period in returned message
    "reportingEnd": "2012-06-01",

    # optional only for version history
    "validFrom": "2012-01-01T10:00:00Z",

    # optional only for version history
    "validTo": "2013-01-01T10:00:00Z",

    # optional only for publication release calendars
    "publicationYear": "2005",

    # optional only for publication release calendars
    "publicationPeriod": "2005-Q1",

    # optional as per annotations
    "annotations": [0],
  }
]
}

```

```

# optional as per attributes at dataset level
"attributes": [0],

# 1st alternative (only if series level
# (dimensionAtObservation <> allDimensions))

"series": {
  "0": {
    "annotations": [],
    "attributes": [0],
    "observations": {
      "0": [1.5931, 0],
      "1": [1.5925, 0]
    }
  },
  "1": {
    "annotations": [ 34 ],
    "attributes": [1],
    "observations": {
      "0": [40.3426, 0],
      "1": [40.3000, 0]
    }
  }
},
{
  "action": "Information",

# 2nd alternative (only if no series level
# (dimensionAtObservation == allDimensions))

"observations": {
  "0:0": [1.5931, 0],
  "0:1": [1.5925, 0],
  "1:0": [40.3426, 0],
  "1:1": [40.3000, 0]
},

# In case that the server does not group dimensions
# with single values at dataset level

{

```

```

"action": "Information",

# 1st alternative (only if series level
# (dimensionAtObservation <> allDimensions))

"series": {
  "0:0:0:0;0": {
    "attributes": [0],
    "observations": {
      "0": [1.5931, 0],
      "1": [1.5925, 0]
    }
  },
  "0:0:0:0;1": {
    "attributes": [1],
    "observations": {
      "0": [40.3426, 0],
      "1": [40.3000, 0]
    }
  }
}
},
{
  "action": "Information",

# 2nd alternative (only if no series level
# (dimensionAtObservation == allDimensions))

"observations": {
  "0:0:0:0:0:0": [1.5931, 0],
  "0:0:0:0:0:1": [1.5925, 0],
  "0:0:0:0:1:0": [40.3426, 0],
  "0:0:0:0:1:1": [40.3000, 0]
}
},

# In case the client is using the detail parameter
# and the server supports it

{
  "action": "Information",

# Detail parameter: serieskeyonly. No observation values,
# attributes or annotations.

```

```

    "observations": {
      "0:0": [],
      "0:1": [],
      "1:0": [],
      "1:1": []
    }
  ],
  {
    "action": "Information",

    # Detail parameter: dataonly. No attributes or annotations.

    "observations": {
      "0:0": [1.5931],
      "0:1": [1.5925],
      "1:0": [40.3426],
      "1:1": [40.3000]
    }
  ],
  {
    "action": "Information",

    # Detail parameter: nodata. No observation values
    # just attributes and annotations.

    "observations": {
      "0:0": [0],
      "0:1": [0],
      "1:0": [0],
      "1:1": [0]
    }
  }
]
}

```