



IMF SDMX CENTRAL

Web Services (API) Guide

Last updated: May 2025

[Abstract](#)

This guide provides information on the web services (API) provided by IMF SDMX

CENTRAL

Structure Web Service

Structural Validation

Dataset Conversion

Data Registration

Table of Contents

1	Scope of this Guide	3
2	REST API Overview	4
2.1	SDMX queries	4
2.2	Elements of a Query	5
2.3	Software Version	6
3	File Formats	7
3.1	CSV Files	7
3.2	JSON Files	7
3.3	XML Files	7
3.4	XLSX Files.....	7
4	Query Structures Web Service.....	12
4.1	Overview	12
4.2	HTTP Headers	12
4.3	Resources	13
4.4	Path Parameters	14
4.5	Request Parameters	14
4.6	Examples.....	17
4.6.1	All concept schemes in SDMX v2.1 format.	17
4.6.2	All structures saved to a file.	17
4.6.3	Any concept with Id OBS_STATUS and all the data structures that reference it.	17
4.6.4	ECOFIN Concept Scheme in SDMX v3.0 format, using Curl, save data to XML file.....	17
5	Programmatic Structural Validation Web Services	18
5.1	Overview	18
5.2	HTTP Headers	18
5.3	Verification Output.....	20
5.3.1	Example of Validation Output for Valid Dataset	20
5.3.2	Example of validation output for invalid dataset.....	21
5.3.3	Example of error output from a server	23
5.3.4	Examples of error types	23
5.4	Examples.....	25
5.4.1	Validate an XML file containing data using Curl.....	25
5.4.2	Validate a CSV file containing data using Curl.....	25
5.4.3	Validate an Excel file containing data using Python.....	26
5.4.4	Validate an XML file containing data using Python	26

6	Programmatic Dataset Conversion Web Services	27
6.1	Overview	27
6.2	HTTP Headers	28
6.3	Examples.....	32
6.3.1	Transform a CSV file containing ECOFIN data into a SDMX-ML 2.1 file.	32
6.3.2	Transform an Excel file containing data in a SDMX-ML 2.1 file using Python.....	32
6.3.3	Transform a SDMX-ML 2.1 file containing data in a CSV file using Python	33
6.3.4	Transform a SDMX-ML 2.1 file containing data in a Fusion Excel file using Python	33
7	Registering data using Web Services	34
7.1	Overview	34
7.2	Introduction.....	34
7.3	SDMX Registration Message	35
7.4	Submit Data Registration Request with Curl.....	36
7.5	Submit Data Registration Request with Python	37

1 Scope of this Guide

This Web Services guide provides instructions on the essential functionalities and usage of the IMF SDMX Central REST API.

This service allows to automate querying, validating, converting datasets, and registering new posting of new data on the National Summary Data Page. The Guide describes how the API can be utilized by running Python code and Bash commands. It is assumed that the reader is familiar with the installation and use of these tools.

Please note that all the functionalities included in this API are accessible through a Graphical User Interface in the [IMF SDMX Central](#) platform. Users of the platform can manually execute these functionalities, with no software installation or technical background required. More detailed instructions on the use of IMF SDMX Central can be found in the IMF SDMX Central User Guide.

More information and guidance on the SDMX framework and available tools can be found in the [official SDMX website](#) and [SDMX.IO](#). Any question or request for assistance related to the content of this guide can be sent [at this mailbox](#).

This guide contains technical documentation and examples for the programmatic execution of the following tasks:

- Querying data structures from IMF SDMX Central Registry to allow users explore available structures and ensure accurate data retrieval and integration.
- Validating data files for SDMX formatting to ensure they conform to required standards and formats, preventing errors, and ensuring data integrity.
- Converting data files into SDMX format enabling compatibility with various SDMX-based tools and systems.
- Registering data in the IMF SDMX Central Registry to ensure that new datasets are properly cataloged and made available for querying and analysis by other users.

2 REST API Overview

IMF SDMX Central provides SDMX web services for querying structures and schemas, including Dataflows, Codelists, Concepts, Data Structures.

The web service entry point is: **<https://sdmxcentral.imf.org/sdmx/v2/>**

It allows external software to connect and communicate using SDMX web services. The Structure REST API follows the SDMX Web Service Guidelines, available at <https://sdmx.org>.

In addition to the standard SDMX specification, IMF SDMX Central supports additional “Accept” header values and query parameters. This document includes both the SDMX-standard and extended query parameters.

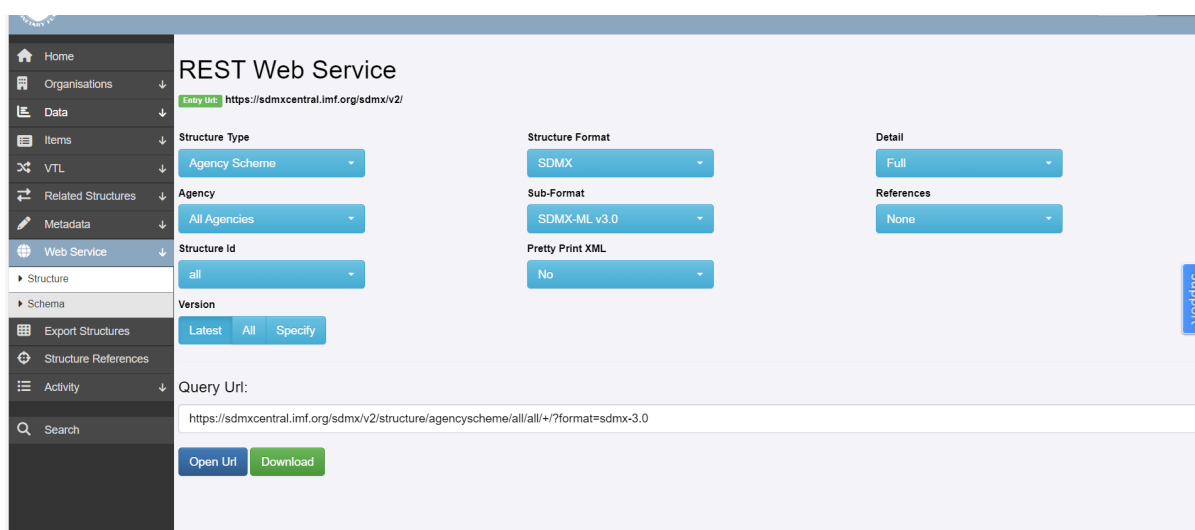
Please note that there currently is no official REST API for data processing services (e.g. validation, conversion, or registration). Instead, FMR specifies its own entry point:

<https://sdmxcentral.imf.org/ws/public/data>.

2.1 SDMX queries

An SDMX API query can be constructed using the information included in this guide and it can be executed by entering it in a browser search bar or making a request through other methods, such as using the Python ‘requests’ library.

Another way to write and execute a query is to use the [‘Web Service’ section of the IMF SDMX Central platform](#).



This functionality allows to build a query for either a **‘Structure’** or a **‘Schema’** using a series of drop-down lists of parameters. In the example above, a request is made to retrieve all available “Agency Schemes”, with the output generated as an **SDMX 3.0 file**. The query, based on selected parameters, is displayed under **‘Query URL’** and it can be manually edited if needed. Pressing **‘Open URL’** or **‘Download’** will result in the query being executed and the output respectively being opened in the browser or downloaded in a file.

In some cases, for example if the goal is to perform a **POST request** (e.g., automatic registration), it is not possible to use the **‘Web Service’** functionality on SDMX Central. As mentioned earlier, requests can be executed outside of SDMX Central using various methods. However, the **SDMX API documentation** recommends **Curl**, a UNIX tool that enables command-line operations without a graphical user interface.

For an example of calling web services via Curl, refer to: [Calling Web Services via Curl - FMR Knowledge Base \(sdmxcloud.org\)](#).

2.2 Elements of a Query

This guide explains how to build a query using the information from the [FMR documentation](#).

The example below shows a generic request written in Python, demonstrating how this guide can help adapt a query to your specific use case.

```
import requests

# Define the URL of the API endpoint
url = 'https://sdmxcentral.imf.org/ws/public/data/validate'

# Define the file path of the input data file
file_path = 'data.xlsx'

# Read the content of the input data file
with open(file_path, 'rb') as file:
    data = file.read()

# Define the headers
headers = {
    'Content-Type': 'application/xml',
    'Accept': 'application/vnd.sdmx.structurespecificdata+xml;version=2.1'
}

# Make the POST request
response = requests.post(url, data=data, headers=headers)

# Write the response to the output file
with open('test_validation.txt', 'wb') as output_file:
    output_file.write(response.content)
```

- **Request's attributes:**
A request requires attributes, highlighted in green, to be executed. One piece of information needed is the **URL** to connect to the required service. The **web service entry point** (see [SDMX queries](#)) is specified in the "url" variable. The example above shows a validation being performed. This means that a file is used as input. The file path is specified in the variable "file_path". Additional attributes that can be requested include the type of request (GET, POST), whether authentication is required (Public, Private), if it accepts files as input and what is the default output format.
- **Headers:**
Headers allow the inclusion of additional parameters in a request, most of which are optional. The "HTTP Headers" section of each chapter provides information on valid header values for specific requests. In the example above, two header elements are included (highlighted in blue), but the number of headers can be adjusted based on the specific requirements of each case.

2.3 Software Version

The content of this guide was tested in a Microsoft Windows 10 Enterprise environment.

Below a list of software used and related version:

- Python: 3.10.9
- Python 'requests' package: 2.28.1
- Curl: 7.88.1

3 File Formats

In this guide, it is relevant to consider the difference between two types of API requests: GET and POST. GET requests are intended to retrieve data from a server. On the other hand, POST requests are used to send data to the server for processing.

Validation, transformation and registrations are POST requests and require input data. A common and effective way to input data is to send a file containing the information to be processed. The SDMX Web Services allow different file formats to be used. This section contains information on the accepted structure of input files.

3.1 CSV Files

Extended instructions on how to structure a CSV file to use in the SDMX Web Services can be found here: [sdmx-csv/data-message/docs/sdmx-csv-field-guide.md at master · sdmx-twg/sdmx-csv · GitHub](#)

An example of a CSV file using the ECOFIN DSD is provided below. It is possible to build a CSV file following the order of the dimensions/measures/attributes that can be seen in SDMX Central. In this case:

```
[DATA_DOMAIN],[REF_AREA],[INDICATOR],[COUNTERPART_AREA],[FREQUENCY],[TIME_PERIOD],[OBS_VALUE],[BASE_PER],[UNIT_MULT],[TIME_FORMAT],[OBS_STATUS]
```

Please note: If this structure of CSV file is used, the system still requires knowing which data structure is being referenced. This means that it will have to be added to the query. See [Example](#).

3.2 JSON Files

Extended instructions on how to structure a JSON file to use in the SDMX Web Services can be found here: [sdmx-json/structure-message at master · sdmx-twg/sdmx-json · GitHub](#)

3.3 XML Files

Extended instructions on how to structure an XML file to use in the SDMX Web Services can be found here: [GitHub - sdmx-twg/sdmx-ml: This repository is used for maintaining the SDMX-ML format specification](#)

3.4 XLSX Files

The format of a data file in Excel is expected to follow the following conventions:

1. Dimension, Attributes and Time Periods appear as header columns:

Demonstration of rule #1 – Dimensions, Attributes, and Time Periods as Column Headers

	A	B	C	D	E	F	G	H	I
1	INDICATOR	FREQ	BASE_PER	UNIT_MULT	TIME_FORMAT	2001	2002	2003	2004
2									

- It is permissible to have multiple worksheets with data. This mechanism can be used to report different frequencies of data per worksheet. The Header section of each worksheet must be consistent in terms of layout, as shown in the image below.

Worksheet 1 with Quarterly Data		Worksheet 2 with Annual Data	
1	Dataflow	IMF:BOP_BPM6(1.0)	
2	DataSet Action	IMF:ECOFIN_DSD(1.0)	
3	DATA_DOMAIN	Information	
4	REF_AREA	BOP6	
5	COUNTERPART_AREA	XX	
6	FREQ	W1	
7	UNIT_MULT	Q	
8	TIME_FORMAT	6	
9	OBS_STATUS	P3M	
10	INDICATOR	A	
11			
12	INDICATOR	1995-Q1	
13	BCA_BP6_EUR		11

1	Dataflow	IMF:BOP_BPM6(1.0)	
2	DataSet Action	IMF:ECOFIN_DSD(1.0)	
3	DATA_DOMAIN	Information	
4	REF_AREA	BOP6	
5	COUNTERPART_AREA	XX	
6	FREQ	W1	
7	UNIT_MULT	A	
8	TIME_FORMAT	6	
9	OBS_STATUS	P3M	
10	INDICATOR	A	
11			
12	INDICATOR		1995
13	BCA_BP6_EUR		99

- Observation Attributes may be reported in the header section, which is used as the default value for all observations.

Demonstration of rule #5 applying a default value for an observation attribute

	A	B	C	D	E	F	G	H	I
1	DATA_DOMAIN	BOP6							
2	REF_AREA	JP							
3	COUNTERPART_AREA	W1							
4	OBS_STATUS	A							
5									
6	INDICATOR	FREQ	BASE_PER	UNIT_MULT	TIME_FORMAT	2001	2002	2003	2004
7									

- The header section should be separated from the data section by a blank row.

A single blank row separates the header and data sections

	A	B	C	D	E	F	G	H	I
1	DATA_DOMAIN	BOP6							
2	REF_AREA	JP							
3	COUNTERPART_AREA	W1							
4									
5	INDICATOR	FREQ	BASE_PER	UNIT_MULT	TIME_FORMAT	2001	2002	2003	2004
6									

- Reported values appear in the data section.

Reported values for Dimensions, Attributes and Time Periods

	A	B	C	D	E	F	G	H	I
1	DATA_DOMAIN	BOP6							
2	REF_AREA	JP							
3	COUNTERPART_AREA	W1							
4	OBS_STATUS	A							
5									
6	INDICATOR	FREQ	BASE_PER	UNIT_MULT	TIME_FORMAT	2001	2002	2003	2004
7	BCA_BP6_XDC	A		0	P1Y	12.2	22.2	32.2	42.2
8	BXCA_BP6_XDC	A		0	P1Y	12.3		32.3	42.3

8. Dimension values are mandatory. If a value is not reported, this will result in an error.

Omitting Dimension value FREQ will result in error

	A	B	C	D	E	F	G	H	I
1	DATA_DOMAIN	BOP6							
2	REF_AREA	JP							
3	COUNTERPART_AREA	W1							
4	OBS_STATUS	A							
5									
6	INDICATOR	FREQ	BASE_PER	UNIT_MULT	TIME_FORMAT	2001	2002	2003	2004
7	BCA_BP6_XDC	A		0	P1Y	12.2	22.2	32.2	42.2
8	BXCA_BP6_XDC					12.3		32.3	42.3

9. Extra rows and columns in the spreadsheet may be added to improve the readability for the user. Blank rows are permitted but with certain restrictions. Blank rows may appear before the header section and between the header section and data section. However, a blank row may not exist within the header section. If a blank row is encountered in the header section, then this is assumed to indicate the end of the header section, and this may cause your spreadsheet to be read incorrectly.
10. Blank columns indicate that no further information should be read from that row. IMF SDMX Central will read from the first column of information in a row until it reaches a blank cell (unless it's a data row). The image below shows a spreadsheet where column H is blank and row 9 is blank. This would mean that in the data section only data for 2001 and 2002 are read (columns F and G). Columns I and J will not be read. However, all the 3 series (rows 7,8 and 10) will be read.

Demonstration of the effect blank rows and columns have on the processing of data

	A	B	C	D	E	F	G	H	I	J
1	DATA_DOMAIN	BOP6								
2	REF_AREA	JP								
3	COUNTERPART_AREA	W1								
4	OBS_STATUS	A								
5										
6	INDICATOR	FREQ	BASE_PER	UNIT_MULT	TIME_FORMAT	2001	2002		2003	2004
7	BCA_BP6_XDC	A		0	P1Y	12.2	22.2		32.2	42.2
8	BXCA_BP6_XDC					12.3			32.3	42.3
9										
10	BEFD_BP6_XDC	A		0	P1Y	12.4	22.4		32.4	42.4

11. It is permissible to have columns in the data section that are not dimensions, attributes or data but may contain additional information for the reader of the spreadsheet. The image below shows a spreadsheet where column F is for additional notes for each data row. The presence of this column will not prevent the data (in columns G to J) from being read even though the data rows themselves do not have a value for the row.

A column (column F) that is not data, dimension or attribute but will not prevent data processing

	A	B	C	D	E	F	G	H	I	J
1	DATA_DOMAIN	BOP6								
2	REF_AREA	JP								
3	COUNTERPART_AREA	W1								
4	OBS_STATUS	A								
5										
6	INDICATOR	FREQ	BASE_PER	UNIT_MULT	TIME_FORMAT	Additional Notes	2001	2002	2003	2004
7	BCA_BP6_XDC	A		0	P1Y		12.2	22.2	32.2	42.2
8	BXCA_BP6_XDC						12.3		32.3	42.3
9	BEFD_BP6_XDC	A		0	P1Y		12.4	22.4	32.4	42.4

12. It is permissible to have entire rows in the data section that are there to indicate what the data represents. This has the restriction that the text in these rows must not be in a column that indicates a Dimension, Attribute or Value. The image below shows row 7 being used to explain what rows 8 and 9 represent. The text for row 7 is in column A, which is now used for additional information.

Addition of rows in the data area to aid readability

	A	B	C	D	E	F	G	H	I	J	K
1	DATA_DOMAIN	BOP6									
2	REF_AREA	JP									
3	COUNTERPART_AREA	W1									
4	OBS_STATUS	A									
5											
6	Comment	INDICATOR	FREQ	BASE_PER	UNIT_MULT	TIME_FORMAT	Additional Notes	2001	2002	2003	2004
7	Balance of Payments, Current Account, Total										
8	Net, National Currency	BCA_BP6_XDC	A		0	P1Y		12.2	22.2	32.2	42.2
9	Credit, National Currency	BXCA_BP6_XDC									
10	Balance of Payments, Exceptional financing										
11	Direct investment, National Currency	BEFD_BP6_XDC	A		0	P1Y		12.4	22.4	32.4	42.4
12											
13											

13. It is permissible to have rows that do not report any observations. In the image above, row 9 reports no values (cells H10, I10, J10 and K10 are all blank). Please note that a file with correct headers but no values in any cell will pass the validation, however it will not convert using the web user interface. It is possible to perform the conversion in this case using the web services.

4 Query Structures Web Service

4.1 Overview

Reference: [Query Structures Web Service - FMR Knowledge Base \(sdmxcloud.org\)](https://sdmxcloud.org/knowledge-base/Query-Structures-Web-Service-FMR-Knowledge-Base) ('Overview' paragraph)

Web Service Entry Point	https://sdmxcentral.imf.org/sdmx/v2/
Access HTTP Method	Public GET
Response Format	Multiple SDMX formats supported. Can be specified in the <i>Accept</i> HTTP Header, or the <i>format</i> parameter of the URL request.
Response Statuses	200 – Success 404 – Structure not found 500 – Server Error
Error Response	SDMX-ML v2.1 Error Response Document

4.2 HTTP Headers

Reference: [Query Structures Web Service - FMR Knowledge Base \(sdmxcloud.org\)](https://sdmxcloud.org/knowledge-base/Query-Structures-Web-Service-FMR-Knowledge-Base) ('Headers' paragraph)

The HTTP headers can be used to specify response format. This can also be defined in the request parameter.

HTTP Header	Purpose	Allowed Values
Accept	To define the response format	<u>SDMX Formats</u> application/vnd.sdmx.structure+edi application/vnd.sdmx.structure+xml;version=1.0 application/vnd.sdmx.structure+xml;version=2.0 application/vnd.sdmx.structure+xml;version=2.1 <u>JSON Format</u> application/vnd.sdmx.json <u>Excel Format</u> application/vnd.xlsx
Accept-Language	This optional header can be used to set the locale to return any multilingual text in (names and descriptions). If the text does not exist in the specified locale, then the default rules will be applied to	<u>Examples:</u> Accept-Language : en (English) Accept-Language : fr (French) Accept-Language : *(all languages – no filter) Accept-Language : all (all languages – no filter)

	find the next best appropriate locale.	
If-Modified-Since	The server sends back the requested resource, with a 200 status, only if it has been last modified after the specified date. If the resource has not been modified since, the response is a 304 without any body.	<p>A date in the expected format of: <Day-Name>, <Day> <Month-Name> <Year> <Hour>:<Minute>:<Second> GMT</p> <p><u>Examples:</u> Fri, 31 Dec 2023 23:59:59 GMT Thu, 8 June 2023 14:00:00 GMT</p>

4.3 Resources

The resource is used to determine which structure type is being queried. The following resources are supported:

- datastructure
- metadatastructure
- categoryscheme
- conceptscheme
- codelist
- hierarchicalcodelist
- organisationscheme
- agencycheme
- dataproviderscheme
- dataconsumerscheme
- organisationunitscheme
- dataflow
- metadataflow
- reportingtaxonomy
- provisionagreement
- structureset
- process
- categorisation
- contentconstraint
- actualconstraint
- allowedconstraint
- attachmentconstraint
- transformationscheme
- rulesetscheme
- userdefinedoperatorscheme
- customtypescheme
- namepersonalisationscheme
- vtlmappingscheme
- structure

4.4 Path Parameters

The path parameters are used to further define the attributes of the request structure(s). All the path parameters are optional. If the path parameters have a default value, it will be used in the absence of the parameter.

Parameter	Purpose	Allowed Values
agencyID	The agency maintaining the artefact to be returned. It is possible to set more than one agency using + as a separator	all – default . any agency Or any string compliant with the SDMX common:NCNameIDType
structureID	The id of the artefact to be returned. It is possible to set more than one id using + as a separator	all – default . all structure ids Or any string compliant with the SDMX common: IDType
version	The version of the structure(s) to be returned	latest – default . latest version all – all versions Or any string compliant with the SDMX common: VersionType
itemID	If the resource is to an item scheme (Codelist, Concept Scheme, Category Scheme), the item inside the scheme can be identified by this parameter	String A string compliant with the SDMX common:NestedNCNameIDType for conceptscheme and agency scheme, SDMX common:IDType for hierarchicalcodelist or with the SDMX common:NestedIDType in all other cases

4.5 Request Parameters

The request parameters are all optional and can be used to define the response detail, format, and any additional structures which reference, or are referenced by those identified in the query path.

Parameter	Purpose	Allowed Values
detail	To define which structures (if any) are output as stubs	full – default . Output full response. allstubs – Output all the structures as stubs. allcompletestubs – include annotations and description. referencestubs – Output the full query result, and any referenced structures are returned as stubs. referencecompletestubs – Complete stubs only on referenced structures. referencepartial – Outputs the full query result and any referenced Codelists, Concept Schemes, Agency Schemes are returned as partial lists based on the Codes, Concepts, and Agencies used by the referencing Provision Agreements, Dataflows, Data Structures, Hierarchical Codelists. Partial Codelists are derived

		<p>from Content Constraints used to define allowable content for data reporting.</p> <p>raw – do not resolve extended codelists.</p> <p><i>Example:</i> detail=allstubs</p>
references	<p>To define if additional structures are returned from the query.</p> <p>The structures can either be ones which reference or are referenced by the structures in the query result.</p> <p>If the query result is for a specific item in an item scheme, then this parameter will identify the references for that item.</p>	<p>none – default. Do not output any additional structures.</p> <p>parents –output structures the reference the structures matching the query.</p> <p>parentsandsiblings – same as parents, but also include all the additional structures referenced by the parents.</p> <p>children – the structures referenced by the structures in the query result.</p> <p>descendants – children and their children (up to any level).</p> <p>all – return all.</p> <p>ancestors – return ancestors.</p> <p>In addition, a concrete type of resource may be used, for example: datastructure</p> <p><i>Example:</i> references=datastructure</p>
partial	<p>If set to true creates partial Codelists in the response based on IMF SDMX Central Content Constraints defining allowable content.</p> <p>The pre-requisite is that the query must be for a single constrainable structure (Provision Agreements, Dataflow, or Data Structure) and include references.</p>	true/false
format	Can be used to define the response format (as an alternative to the HTTP Accept Header).	<p>sdmx (latest version)</p> <p>sdmx-3.0</p> <p>sdmx-2.1</p> <p>sdmx-2.0</p> <p>sdmx-1.0</p> <p>sdmx-edi</p> <p>sdmx-edi-lenient</p> <p>sdmx-json</p> <p>xlsx</p> <p><i>Example:</i> format=sdmx-edi</p>
locale	This optional parameter can be used to set the locale to return any multilingual text in (names and descriptions). If the	Any locale

	text does not exist in the specified locale, then the default rules will be applied to find the next best appropriate locale.	<i>Example:</i> locale=fr
saveAs	<p>If provided the HTTP Header 'Content-Disposition' will be set to attachment with the filename being set to the value provided.</p> <p>This will result in the response being saved to a file.</p> <p>The file extension is not required as IMF SDMX Central will determine the extension based on the response format.</p>	<p>String</p> <p><i>Example:</i> saveAs=myDownload</p>
prettyPrint	If the you are requesting XML, and you would like the response XML to be formatted, then you can pass true	<p>String</p> <p>prettyPrint=true</p>
validFrom	For structures with a defined validFrom or validTo value, returns only those structures which have a validFrom value before the specified date	<p>Single time period. Conforms to the ISO-8601 standard but SDMX date time formats may also be used</p> <p><i>Example:</i> validFrom=1960-12-31</p>
validTo	For structures with a defined validFrom or validTo value, returns only those structures which have a validTo value after the specified date	<p>Single time period. Conforms to the ISO-8601 standard but SDMX date time formats may also be used.</p> <p><i>Example:</i> validTo=1960-12-31</p>
validOn	Returns structures where the items listed are applicable for the specified date. This parameter is only applicable for those structures which support Item Validity.	<p>Single time period. Conforms to the ISO-8601 standard but SDMX date time formats may also be used.</p> <p><i>Example:</i> validOn=1960-12-31</p>
labels	<p><u>For sdmx-csv format only:</u> For each element of the CSV, will return either the ID only or the ID and Name separated by a colon</p>	<p>Id – default</p> <p>Both</p> <p><i>Example:</i> labels=both</p>
timeFormat	<p><u>For sdmx-csv format only:</u> Normalized TIME_PERIOD values are converted to the most granular ISO 8601 representation taking into account the highest frequency of the data in the message</p>	<p>original – default</p> <p>normalized</p> <p><i>Example:</i> timeFormat=normalized</p>

4.6 Examples

As querying structures is a GET request, these examples can be copy pasted in a browser search tab and executed by pressing enter.

The URL used to get structures/schemas is built using the items mentioned in the previous paragraphs following this blueprint:

```
https://sdmxcentral.imf.org/sdmx/v2/structure/resource/agencyID/resourceID/version/itemID/?parameter1=x&parameter2=y
```

4.6.1 All concept schemes in SDMX v2.1 format.

<https://sdmxcentral.imf.org/sdmx/v2/structure/conceptscheme/all/all/+/?format=sdmx-2.1>

4.6.2 All structures saved to a file.

<https://sdmxcentral.imf.org/sdmx/v2/structure/structure/all/all/+/?format=sdmx-2.1&saveAs=fullexport>

4.6.3 Any concept with Id OBS_STATUS and all the data structures that reference it.

https://sdmxcentral.imf.org/sdmx/v2/structure/conceptscheme/all/all/all/OBS_STATUS?references=datastucture

4.6.4 ECOFIN Concept Scheme in SDMX v3.0 format, using Curl, save data to XML file.

curl -X GET

https://sdmxcentral.imf.org/sdmx/v2/structure/conceptscheme/IMF/ECOFIN_CONCEPTS/+/?format=sdmx-3.0 -o test_download_curl_1.xml

5 Programmatic Structural Validation Web Services

5.1 Overview

Reference: [Validation Web Service - FMR Knowledge Base \(sdmxcloud.org\)](https://sdmxcloud.org/knowledge-base/validation-web-service/)

The validation web service performs structural validation against a specified DSD. It consumes a dataset (both SDMX and non-SDMX formats are supported) and returns a JSON response identifying details about the dataset, including if there are any validation errors.

URL Entry Point	https://sdmxcentral.imf.org/ws/public/data/validate
Access	Public (default). Configurable to Private
HTTP Method	POST
Accepts	CSV, XLSX, SDMX-ML, SDMX-EDI (any format for which there is a Data Reader)
Compression	Zip files supported, if loading from URL gzip responses supported
Content Type	1. multipart/form-data (if attaching file) – the attached file must be in field name of uploadFile 2. application/text or application/xml (if submitting data in the body of the POST)
Response Format	application/json
Response Statuses	200 - Validation could be performed 400 - Validation could not be performed (either an unreadable dataset, or unresolvable reference to a required structure) 401 - Unauthorized (if access has been restricted) 500 - Server Error

5.2 HTTP Headers

HTTP Header	Purpose	Allowed Values
Data-Format	Used to inform the server when the data is in CSV format. See examples for more details on performing validation using a CSV file.	csv;delimiter=[delimiter] Where [delimiter] is either: <ul style="list-style-type: none">• comma• tab• semicolon• space
Sender-Id	The SenderId is included in the verification report. If not provided, the SenderId will be taken from the header of the dataset. If the dataset does not contain a SenderId (for example a non-SDMX format) then the verification report will contain the SenderId of IMF SDMX Central.	The following characters are allowed: A-z, a-z 0-9 \$, _, -, @, \

Structure	<p>Provides the structure to verification the data against.</p> <p>This is optional as this information may be present in the header of the DataSet. If provided this value will override the value in the dataset (if present).</p>	Valid SDMX URN for Provision Agreement, Dataflow, or Data Structure Definition.
Inc-Metrics	<p>Optional. Includes metrics on the verification.</p> <p>This will add extra detail to the verification report</p>	Boolean (true/false)
Inc-Valid	<p>Optional. Instructs the service to include a dataset with all the valid series and observations in the response.</p> <p>As the result will contain a separate file for the dataset, the response format will be set to either multipart/mixed message with a boundary per file, or if the Zip header is set to true, the output will be a single zip file.</p> <p>The file is called ValidData with the file extension based on the output format.</p>	Boolean (true/false)
Inc-Invalid	<p>Optional. Instructs the service to include a dataset with all the invalid series and observations in the response.</p> <p>As the result will contain a separate file for the dataset, the response format will be set to either multipart/mixed message with a boundary per file, or if the Zip header is set to true, the output will be a single zip file.</p> <p>The file is called InvalidData with the file extension based on the output format.</p>	Boolean (true/false)
Accept	<p>Optional. Instructs the service which data output format to output the valid or invalid datasets in.</p> <p>This Header is only used if Inc-Valid or Inc-Invalid are set to true.</p>	See Accept formats for REST Data Query
Zip	Optional. Compresses the output as a zip file. If used in conjunction with Inc-	Boolean (true/false)

	Valid or Inc-Invalid the zip will contain multiple files.	
--	--	--

5.3 Verification Output

The verification output contains both human readable error descriptions, as well as machine processible locations of the errors within the dataset. The location in the dataset is described as a key or observation locator in the format; A:UK:M:2008 – where each component relates to the Dimension value, separated by a colon. If the error position is observation, the last part of the key is the observation's time period.

5.3.1 Example of Validation Output for Valid Dataset

```
{
  "Meta": {
    "RequestTime": 1564410081711,
    "Duration": 43
  },
  "FileFormat": "Structure Specific (Compact) v2.1",
  "Prepared": "2019-07-29T10:23:01",
  "SenderId": "FR_DEMO",
  "DataSetId": null,
  "Status": "Complete",
  "Errors": false,
  "Datasets": [
    {
      "DSD":
"urn:sdmx:org.sdmx.infomodel.datastructure.DataStructure=OECD:HIGH_AGLINK_2011(1.0)
",
      "Dataflow":
"urn:sdmx:org.sdmx.infomodel.datastructure.Dataflow=OECD:AGRIC_OUTLOOK_2011_2020(1.0)",
      "DataProvider":
"urn:sdmx:org.sdmx.infomodel.base.DataProvider=METATECH:DATA_PROVIDERS(1.0).METATECH",
      "ProvisionAgreement":
"urn:sdmx:org.sdmx.infomodel.registry.ProvisionAgreement=OECD:OECD_AGRIC_OUTLOOK(1.0)",
      "KeysCount": 2,
      "ObsCount": 62,
      "GroupsCount": 0,
      "Errors": false
      "ReportedPeriods": {
        "A": {
          "Name": "Annual",
          "StartPeriod": "1990",
          "EndPeriod": "2020"
        }
      },
    },
  ],
  "PreventsConversion": false,
  "PreventsPublication": false
}
```

5.3.2 Example of validation output for invalid dataset

```
{
  "Meta": {
    "RequestTime": 1564401209760,
    "Duration": 34
  },
  "InvalidData": {
    "Datasets": [
      {
        "Structure":
"urn:sdmx:org.sdmx.infomodel.registry.ProvisionAgreement=OECD:OECD_AGRIC_OUTLOOK(1.0)",
        "Series": 2,
        "Observations": 61,
        "Groups": 0
      }
    ]
  },
  "ValidData": {
    "Datasets": [
      {
        "Structure":
"urn:sdmx:org.sdmx.infomodel.registry.ProvisionAgreement=OECD:OECD_AGRIC_OUTLOOK(1.0)",
        "Series": 2,
        "Observations": 32,
        "Groups": 0
      }
    ]
  },
  "FileFormat": "Structure Specific (Compact) v2.1",
  "Prepared": "2019-07-29T10:23:01",
  "SenderId": "FR_DEMO",
  "DataSetId": null,
  "Status": "Complete",
  "Errors": true,
  "Datasets": [
    {
      "DSD":
"urn:sdmx:org.sdmx.infomodel.datastructure.DataStructure=OECD:HIGL_AGLINK_2011(1.0)",
      "Dataflow":
"urn:sdmx:org.sdmx.infomodel.datastructure.Dataflow=OECD:AGRIC_OUTLOOK_2011_2020(1.0)",
      "DataProvider":
"urn:sdmx:org.sdmx.infomodel.base.DataProvider=METATECH:DATA_PROVIDERS(1.0).METATECH",
      "ProvisionAgreement":
"urn:sdmx:org.sdmx.infomodel.registry.ProvisionAgreement=OECD:OECD_AGRIC_OUTLOOK(1.0)",
      "KeysCount": 3,
      "ObsCount": 93,
      "GroupsCount": 0,
      "ReportedPeriods": {
        "A": {
          "Name": "Annual",
          "StartPeriod": "1990",
          "EndPeriod": "2020"
        }
      }
    },
    {
      "Errors": true,
      "ValidationReport": [
        {
          "Type": "Constraint",
          "Errors": [
            {

```

```

        "Message": "Disallowed Dimension Value: REF_AREA=AFR",
        "Dataset": 0,
        "ComponentId": " REF_AREA ",
        "ReportedValue": "AFR",
        "Position": "Series",
        "Keys": ["AFR:BT:AA"]
    }
]
},
{
    "Type": "Representation",
    "Errors": [
        {
            "Message": "Dimension 'VARIABLE' is reporting value 'AA' which is not
a valid representation in referenced Codelist
'OECD:CL_HIGH_AGLINK 2011 VARIABLE(1.0) '",
            "Dataset": 0,
            "Position": "Series",
            "ComponentId": " VARIABLE",
            "ReportedValue": "AA",
            "Keys": ["AFR:BT:AA"]
        },
        {
            "Message": "Error in Primary Measure 'OBS_VALUE': Reported value 'XXX'
is not of expected type 'Double'",
            "Dataset": 0,
            "ComponentId": " OBS_VALUE",
            "ReportedValue": "XXX",
            "Position": "Observation",
            "Keys": ["AFR:BT:IM:2010"]
        }
    ]
},
{
    "Type": "FormatSpecific",
    "Errors": [
        {
            "Message": "Unexpected attribute 'ASD' for element
'StructureSpecificData/DataSet/Series/Obs'",
            "Dataset": 0,
            "Position": "Dataset"
        }
    ]
}
]
}
},
{
    "PreventsConversion": false,
    "PreventsPublication": true
}
}

```

Note the first three elements 'Meta', 'InvalidData', 'ValidData', are present in the report if Inc-Metrics is set to true. Inc-valid and Inc-Invalid set to true enables the report to know the metrics for the invalid and valid data.

The Error Position is either set to Dataset, Series, Observation, or Group.

PreventsConversion and PreventsPublication is an indication on the severity of the error. These settings on which errors prevent conversion and publication can be set in the Fusion Registry by the administrator of the system.

5.3.3 Example of error output from a server

An example error output from a server, which makes the request un-processible, is shown below:

```
{"Error": "Unrecognised file format, contents of file are: this is a bad format"}
```

5.3.4 Examples of error types

In the validation message received after performing the automatic validation on some data, two elements' "Errors" may be present. The first error message is a Boolean, reporting if errors were or were not found in the data. If this flag is true, then a second error message is added containing "ValidationReport", which in turns contains more details about the errors that were found. Below there is a non-exhaustive list of potential errors.

- **Duplicate Observations**

When conflicting observations are inputted for a given indicator and time period, an error is raised. In this case, for indicator NGDPVA_ISIC4_A_XDC, time period 1997, two different observations are supplied: 50 and 100.

```
Errors":true,
"ValidationReport":[
  {
    "Type":"Duplicate",
    "Errors":[
      {
        "ErrorCode":"REG-201-230",
        "Message":"Duplicate value reported NAG:BS:NGDPVA_ISIC4_A_XDC:_Z:A:1997.
Reported values are: '50' and '100'",
        "Dataset":0,
        "ComponentId":"OBS_VALUE",
        "ReportedValue":"100",
        "Position":"Observation",
        "Keys":["NAG:BS:NGDPVA_ISIC4_A_XDC:_Z:A:1997"]
      }
    ]
  }
]
```

- **Semantically compliant**

This is a check related to the format of the file used for the validation. In this case an Excel file is used, and one of the requirements is that time periods should be in ascending chronological order. In this case this criterion was not satisfied:

```
Errors":true,
"ValidationReport":[
  {
    "Type":"FormatSpecific",
    "Errors":[
      {
        "ErrorCode":"-",
        "Message":"Worksheet 'data' contains Annual data which is not in the expected
chronological order",
        "Dataset":0,
        "Position":"Dataset",
      }
    ]
  }
]
```


- **Mandatory components**

In the SDMX framework, certain information must be provided. One example of this are Dimensions (as opposed to Attributes that are not mandatory), specified in a given DSD. In this example, value for a dimension was not reported:

```
Errors":true,
"ValidationReport":[
  {
    "Type":"Structure",
    "Errors":[
      {
        "ErrorCode":"REG-201-186",
        "Missing value for Dimension INDICATOR",
        "Dataset":0,
        "ComponentId":"INDICATOR",
        "Position":"Series",
      }
    ]
  }
]
```

- **Obs Status**

This check ensures that the OBS_STATUS provided is compliant with the OBS_STATUS codelist and that it is used coherently with the data input of the validation. In this case one data series has OBS_STATUS=="M" which means 'Missing' and thus an empty value is expected. However, that is not the case, and we find observations associated with this data series:

```
Errors":true,
"ValidationReport":[
  {
    "Type":" ObsStatusValidator",
    "Errors":[
      {
        "ErrorCode":"REG-201-130",
        "Observation value has been specified but OBS_STATUS enforces that this is
not valid. OBS_STATUS: M",
        "Dataset":0,
        "Position":"Observation",
      }
    ]
  }
]
```

- **Multiple Errors**

In this example multiple checks fail. This is how they appear in the validation report:

```
"Errors":true,
"ValidationReport":[
{
  "Type":"Structure",
  "Errors":[
    {
      "ErrorCode":"REG-201-186",
      "Message":"Missing value for Dimension INDICATOR",
      "Dataset":0,
      "ComponentId":"INDICATOR",
      "Position":"Series",
      "Keys":["NAG:BS:_Z:A"]}},
    {
      "Type":"Duplicate",
      "Errors":[
        {
          "ErrorCode":"REG-201-230",
          "Message":"Duplicate value reported
NAG:BS:NGDPVA_ISIC4_G_XDC:_Z:A:1997. Reported values are: '50' and '100'",
          "Dataset":0,
          "ComponentId":"OBS_VALUE",
          "ReportedValue":"100",
          "Position":"Observation",
          "Keys":["NAG:BS:NGDPVA_ISIC4_G_XDC:_Z:A:1997"]}
        ]
      }
    ]
  }
}]}
```

5.4 Examples

As validating files is a POST method, these examples have to be executed using Curl or another comparable method.

5.4.1 Validate an XML file containing data using Curl

```
curl -X POST --header "Content-Type:application/xml" --header "Accept:
application/vnd.sdmx.structurespecificdata+xml;version=2.1" --data-binary @data.xml
"https://sdmxcentral.imf.org/ws/public/data/validate" > test_validation.txt
```

5.4.2 Validate a CSV file containing data using Curl

```
curl -X POST --data-binary @data.csv --header
"Accept:application/vnd.sdmx.structurespecificdata+xml;version=2.1" --header "Data-
Format:csv;delimiter=comma" --header
"Structure:urn:sdmx:org.sdmx.infomodel.datastructure.DataStructure=IMF:ECOFIN_DSD(1.0)" --header
"Content-Type:application/text" "https://sdmxcentral.imf.org/ws/public/data/validate" > test_validation.txt
```

5.4.3 Validate an Excel file containing data using Python

```
import requests

# Define the URL of the API endpoint
url = 'https://sdmxcentral.imf.org/ws/public/data/validate'

# Define the file path of the input file
file_path = 'data.xlsx'

# Read the content of the input file
with open(file_path, 'rb') as file:
    data = file.read()

# Define the headers
headers = {
    'Content-Type': 'application/xml',
    'Accept': 'application/vnd.sdmx.structurespecificdata+xml;version=2.1'
}

# Make the POST request
response = requests.post(url, data=data, headers=headers)

# Write the response to the output file
with open('test_validation.txt', 'wb') as output_file:
    output_file.write(response.content)
```

5.4.4 Validate an XML file containing data using Python

```
import requests

# Define the URL of the API endpoint
url = 'https://sdmxcentral.imf.org/ws/public/data/validate'

# Define the file path of the input file
file_path = 'data.xml'

# Read the content of the input file
with open(file_path, 'rb') as file:
    data = file.read()

# Define the headers
headers = {
    'Content-Type': 'application/xml',
    'Accept': 'application/vnd.sdmx.structurespecificdata+xml;version=2.1'
}

# Make the POST request
response = requests.post(url, data=data, headers=headers)

# Write the response to the output file
with open('test_validation.txt', 'wb') as output_file:
    output_file.write(response.content)
```

6 Programmatic Dataset Conversion Web Services

6.1 Overview

Reference: [Data Transformation Web Service - FMR Knowledge Base \(sdmxcloud.org\)](https://sdmxcloud.org/)

The Convert web service can be used to convert a dataset from one accepted file format to another. To use this web service, POST the file to the URL Entry Point.

URL Entry Point	https://sdmxcentral.imf.org/ws/public/data/transform
Access	Public
HTTP Method	POST
Accepts	CSV, XLSX, SDMX-ML, SDMX-EDI (any format for which there is a Data Reader)
Content Type	<ol style="list-style-type: none">1. multipart/form-data (if attaching file) – the attached file must be in field name of uploadFile2. application/text or application/xml (if submitting data in the body of the POST)
Response Format	SDMX Structure Specific v2.1
Response Statuses	200 - Transformation performed 400 - Transformation could not be performed (either an unreadable dataset, or resolvable reference to a required structure) 401 - Unauthorized (if access has been restricted) 500 - Server Error

6.2 HTTP Headers

The following headers are all optional and can be used to provide more details on how to perform the conversion.

HTTP Header	Purpose	Allowed Values
Accept	<p>The data transmission format to convert the dataset to.</p> <p>Note: From FMR 11.5.0 the format (if not specified) defaults to the input format. Previous versions defaulted to SDMX Structure Specific 2.1</p>	<p>SDMX Formats</p> <ul style="list-style-type: none"> • application/vnd.sdmx.data+csv;version=2.0.0;labels=[id name both];timeFormat=[original normalized];keys=[none obs series both] • application/vnd.xlsx • application/vnd.sdmx.genericdata+xml;version=2.1 • application/vnd.sdmx.structurespecificdata+xml;version=2.1 • application/vnd.sdmx.generictimeseriesdata+xml;version=2.1 • application/vnd.sdmx.structurespecifictimeseriesdata+xml;version=2.1 • application/vnd.sdmx.data+json;version=1.0.0 • application/vnd.sdmx.data+csv;version=1.0.0;labels=[id both];timeFormat=[original normalized] • application/vnd.sdmx.data+edi <p>SDMX Formats to be supported in future FMR releases</p> <ul style="list-style-type: none"> • application/vnd.sdmx.data+json;version=2.0.0 • application/vnd.sdmx.data+xml;version=3.0.0 <p>Note that the Fusion Excel data transmission format is supported as the input, but not output of a transformation.</p>
Data-Format	Used to inform the server when the data is in CSV format.	<p>csv;delimiter=[delimiter]</p> <p>Where [delimiter] is either:</p> <ul style="list-style-type: none"> • comma • tab • semicolon • space
Structure	<p>(optional) Provides the structure to validate the data against.</p> <p>This is optional as this information may be present in the header of the DataSet. If provided this value will</p>	Valid SDMX URN for Provision Agreement, Dataflow, or Data Structure Definition

	override the value in the dataset (if present).	
Receiver-Id (Since v9.8)	<p>The ReceiverId may be included in the validation report.</p> <p>If not provided, the ReceiverId will be taken from the header of the dataset if it is present.</p> <p>If the dataset does not contain a ReceiverId (for example a non-SDMX format) then the validation report will not contain a ReceiverId in the header.</p>	The following characters are allowed: A-z, a-z 0-9 \$, _, -, @, \
Dataset-Idx	If the loaded file contains multiple datasets, this argument can be used to indicate which dataset is transformed. If this argument is not present then all datasets will be in the output file (if the file formats permits multiple datasets).	Zero indexed integer, example: 0
Dataset-Id (Since v9.8)	An optional parameter which allows the user to specify the value of the DataSetID generated in the validation.	<p>The following characters are allowed: A-z, a-z 0-9 \$, _, -, @, \ Specific variables permit the insertion of Data Structure / Data Flow values. These values are:</p> <p> \${DATFLOW_ID} \${DATFLOW_ACY} \${DATFLOW_VER} \${DSD_ID} \${DSD_ACY} \${DSD_VER} </p> <p>Note that dots in the version number will be replaced with the _ character, since dots are not permitted in the ID.</p>
Dataset-Action (Since v9.8.1)	An optional parameter which allows the user to specify the value of the DataSetAction generated in the validation report. If this parameter is not specified, the default value will be used.	<p>May be one of the following:</p> <ul style="list-style-type: none"> • Append • Replace • Merge • FullReplace • Delete • Information
Map-Structure (Since v9.2.13)	An optional parameter to inform the Fusion Registry to transform the structure of the dataset to conform to another Data Structure Definition.	Valid SDMX URN for Dataflow or Data Structure Definition.

	<p>The value provided can be a URN of a Dataflow or Data Structure Definition to map the incoming data to. A Structure Map must exist in the Fusion Registry which maps between the incoming Data Structure/Dataflow and Mapped Data Structure/Dataflow.</p> <p>Alternatively the URN may be the URN of the Data Structure Map to use for the mapping (since v9.4.4)</p>	
Inc-Unmapped (Since v9.6.5)	<p>If the Map-Structure Header is used, then the inclusion of Inc-Unmapped will output a second dataset, if there are unmapped series. The additional dataset contains the data that could not be mapped due to missing mapping rules, or ambiguous outputs.</p> <p>The format of the additional dataset is the same format as the output dataset.</p> <p>As the result may contain a separate file, the response format is either set to multipart/mixed message with a boundary per file, or if the Zip header is set to true, the output will be a single zip file. The file names are 'out' and 'unmapped' with the file extension based on the output format.</p>	Boolean (true/false)
Inc-Unmapped Report (Since v11.5.0)	<p>If the Map-Structure Header is used, then the inclusion of Inc-UnmappedReport may output another file, if there are unmapped series. The additional file contains a report on the information that could not be mapped due to missing mapping rules, or ambiguous outputs.</p> <p>The format of this report consists of JSON elements:</p> <ul style="list-style-type: none"> • The StructureMap used in the mapping • The Source Structure URN • The Target Structure URN 	Boolean (true/false)

	<ul style="list-style-type: none"> • The Result <p>The result consists of an Input and an Output which details what the input managed to map to. The output also contains an Array called "MissingDimensions" which lists the ID of the missing dimensions.</p>	
Inc-Metrics (Since v9.6.5)	<p>Includes metrics on the transformation.</p> <p>The result will contain a separate file, either as a multipart/mixed message with a boundary per file, or if the Zip header is set to true, the output will be a single zip file.</p>	Boolean (true/false)
Fail-On-Error (Since v9.5.0)	An optional parameter to tell the transformation process to fail if an error is detected in the dataset.	Boolean (true/false)
Zip (Since v9.6.5)	Compresses the output as a zip file. This if used in conjunction with Inc-Metrics or Inc-Unmapped the zip will contain multiple files.	Boolean (true/false)
Duplicate-Behaviour (Since v11.1.6)	Specify the behaviour to perform when duplicate observations are encountered. Either the duplicates can be preserved or either the first or last value can be used.	<p>May be one of the following:</p> <ul style="list-style-type: none"> • useFirst • useLast • preserve
Skip-Validation (Since v11.5.1)	Allows the validation process to be skipped when transforming a file. Useful when the input file is well understood or large. Default is false.	Boolean (true/false)

6.3 Examples

As converting files is a POST method, these examples must be executed using Curl or another comparable method.

6.3.1 Transform a CSV file containing ECOFIN data into a SDMX-ML 2.1 file.

```
$ curl -X POST --data-binary @data.csv --header  
"Accept:application/vnd.sdmx.structurespecificdata+xml;version=2.1" --header "Data-  
Format:csv;delimiter=comma" --header  
"Structure:urn:sdmx:org.sdmx.infomodel.datastructure.DataStructure=IMF:ECOFIN_DSD(1.0)" --header  
"Content-Type:application/text" "https://sdmxcentral.imf.org/ws/public/data/transform" >  
transformation.xml
```

6.3.2 Transform an Excel file containing data in a SDMX-ML 2.1 file using Python.

```
import requests  
  
# Define the URL of the API endpoint  
url = 'https://sdmxcentral.imf.org/ws/public/data/transform'  
  
# Define the file path of the input file  
file_path = 'data.xlsx'  
  
# Read the content of the input file  
with open(file_path, 'rb') as file:  
    data = file.read()  
  
# Define the headers  
headers = {  
    'Accept': 'application/vnd.sdmx.structurespecificdata+xml;version=2.1',  
    'Content-Type': 'application/text'  
}  
  
# Make the POST request  
response = requests.post(url, data=data, headers=headers)  
  
# Write the response to the output XML file  
with open('test_output.xml', 'wb') as output_file:  
    output_file.write(response.content)
```

6.3.3 Transform a SDMX-ML 2.1 file containing data in a CSV file using Python

```
import requests

# Define the URL of the API endpoint
url = 'https://sdmxcentral.imf.org/ws/public/data/transform'

# Define the file path of the input file
file_path = 'data.xml'

# Read the content of the input file
with open(file_path, 'rb') as file:
    data = file.read()

# Define the headers
headers = {
    'Accept': 'application/vnd.sdmx.data+csv;version=2.0.0',
    'Content-Type': 'application/text'
}

# Make the POST request
response = requests.post(url, data=data, headers=headers)

# Write the response to the output CSV file
with open('test_output.csv', 'wb') as output_file:
    output_file.write(response.content)
```

Notice how the 'Accept' header must change in order to specify a new desired output. See [HTTP Headers](#) for more details.

6.3.4 Transform a SDMX-ML 2.1 file containing data in a Fusion Excel file using Python

```
import requests

# Define the URL of the API endpoint
url = 'https://sdmxcentral.imf.org/ws/public/data/transform'

# Define the file path of the input file
file_path = 'data.xml'

# Read the content of the input file
with open(file_path, 'rb') as file:
    data = file.read()

# Define the headers
headers = {
    'Accept': 'application/vnd.xlsx',
    'Content-Type': 'application/text'
}

# Make the POST request
response = requests.post(url, data=data, headers=headers)

# Write the response to the output file
with open('test_output.xlsx', 'wb') as output_file:
    output_file.write(response.content)
```

7 Registering data using Web Services

7.1 Overview

Reference: [Data Registration - Fusion Registry Wiki \(sdmxcloud.org\)](https://sdmxcloud.org/wiki/Data%20Registration)

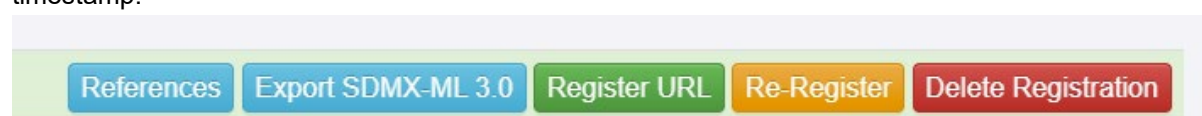
URL Entry Point	https://sdmxcentral.imf.org/ws/secure/sdmxapi/rest
Access	Private (admin/agency/data provider for the Provision Agreement)
HTTP Method	POST
Accepts	SDMX-ML Submit Registration Request
Compression	gzip
Content Type	xml
Response Format	xml
Response Statuses	200 - registration success 400 - Bad URN syntax 401 - Unauthorized (if access has been restricted) 403 - No results (no data) 500 - Server Error

7.2 Introduction

Data can be registered using the web user interface at the following link: [IMF SDMX Central](#)

This procedure allows to link a dataflow and a data provider to an URL pointing to a data file in SDMX-ML format.

In order to perform a registration, it is necessary to login. After this step, the button 'Register URL' will be available, and it will allow to perform a registration by inputting Dataflow, Data Provider and URL. It is also possible to select an existing registration and 'Re-Register' it, to update the 'Last Updated' timestamp.



To perform a registration using the Web Services, a POST request is used to send a SDMX_ML file containing a registration message. This file does not contain the data we want to register, but rather the information that we would otherwise input manually using the user interface, including the URL that points to the actual data.

In the paragraphs below it is possible to find the registration message XML file that can be edited and used to execute an automatic registration, and step by step instructions on how to do it using either Curl or Python.

7.3 SDMX Registration Message

The XML file below is an example of a SDMX message used to perform a registration through Web Services:

```
<?xml version="1.0" encoding="UTF-8"?>
<mes:RegistryInterface xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xml="http://www.w3.org/XML/1998/namespace"
xmlns:mes="http://www.sdmx.org/resources/sdmxml/schemas/v2_1/message"
xmlns:reg="http://www.sdmx.org/resources/sdmxml/schemas/v2_1/registry"
xmlns:com="http://www.sdmx.org/resources/sdmxml/schemas/v2_1/common"
xsi:schemaLocation="http://www.sdmx.org/resources/sdmxml/schemas/v2_1/message
https://registry.sdmx.org/schemas/v2_1/SDMXMessage.xsd">
  <mes:Header>
    <mes:ID>IREF981386</mes:ID>
    <mes:Test>false</mes:Test>
    <mes:Prepared>2020-02-20T11:28:51Z</mes:Prepared>
    <mes:Sender id="ZZZ"/>
    <mes:Receiver id="not_supplied"/>
  </mes:Header>
  <mes:SubmitRegistrationsRequest>
    <reg:RegistrationRequest action="Replace">
      <reg:Registration lastUpdated="2019-03-21T12:58:17" indexReportingPeriod="false" indexDataSet="false"
indexAttributes="false" id="1167332d21d37d8589879ce5ff7d35ee" indexTimeSeries="false"
id="B1CFA93A355AD234D318B80DEDE2ADCB" >
        <reg:ProvisionAgreement>
          <Ref package="registry" agencyID="IMF" id="ZZFR_TEST" version="1.0" class="ProvisionAgreement"/>
        </reg:ProvisionAgreement>
        <reg:Datasource>
          <reg:SimpleDataSource>https://si3.bcentral.cl/SieteSdmx/SieteSdmx.ashx?SDMXCategory=SDMX_BOP6</reg:
SimpleDataSource>
        </reg:Datasource>
      </reg:Registration>
    </reg:RegistrationRequest>
  </mes:SubmitRegistrationsRequest>
</mes:RegistryInterface>
```

In the SDMX message above, the values highlighted are the parameters that should be edited to perform a registration:

- **(Sender) id**: CL_ORGANISATION code of the institution submitting the registration;
- **(Receiver) id**: CL_ORGANISATION code for IMF ('1C0')
- **agencyID**: The agency associated with a Provision Agreement;
- **(Provision Agreement) id**: The ID of the Provision Agreement;
- **(Registration) id**: this value will be stored in SDMX Central only if it is the first registration for the Dataflow/Country. If it is an attempt to update an existing registration, the value is ignored, and the previously existing ID is kept;
- **SimpleDataSource**: It can be a URL or a file.

Please note that the information contained in the Header ('<mes>:Header') is ignored when performing the registration.

7.4 Submit Data Registration Request with Curl

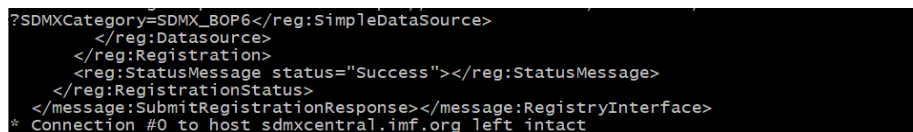
Submitting a registration requires to be authenticated. This can be done directly at the time of execution using Curl. The steps below outline how to perform an automatic registration:

1. Save the SDMX message from [SDMX Registration Message](#) in an XML file and edit it as needed
2. Open an environment where you can use Curl
3. Make sure the current working directory is the one where your XML registration request is saved
4. Copy and paste the following command:

```
#!/bin/bash
URL=https://sdmxcentral.imf.org/ws/secure/sdmxapi/rest
AUTH="username:password"
curl -X POST -H "Content-Type: application/xml" -u ${AUTH} -d @filename.xml -v ${URL}
```

Where 'URL' is the Entry Point (see [Overview](#)), **AUTH** should be edited to contain the user's username and password and **filename.xml** should be replaced with the name of the file referenced at point 1 of the list above.

After executing this command, a response message will be displayed. If the registration was successful it should look like the screenshot below:



```
?SDMXCategory=SDMX_BOP6</reg:SimpleDataSource>
</reg:DataSource>
</reg:Registration>
<reg:StatusMessage status="Success"></reg:StatusMessage>
</reg:RegistrationStatus>
</message:SubmitRegistrationResponse></message:RegistryInterface>
* Connection #0 to host sdmxcentral.imf.org left intact
```

<reg:StatusMessage status="Success"> suggests that the registration was carried out successfully. The result should be visible from IMF SDMX Central.

7.5 Submit Data Registration Request with Python

The code below can be used to submit an automatic registration using Python and the 'requests' package. Highlighted in yellow, the fields that should be edited according to the specific case: username, password and the name of the file containing the SDMX message shown in [SDMX Registration Message](#).

```
import requests

# Define the authentication credentials required to access the API
AUTH = ('username', 'password')
# Define the URL of the API endpoint you want to send the POST request to
URL = 'https://sdmxcentral.imf.org/ws/secure/sdmxapi/rest'

# Set the name of the XML file
xml_file = "filename.xml"

# Define the headers for the request
headers = {'Content-Type': 'application/xml'}
# Open the XML file and read its content
with open(xml_file, 'r') as file:
    xml = file.read()

# Make a POST request to the specified URL, passing the XML data and headers, and providing authentication
response = requests.post(URL, data=xml, headers=headers, auth=AUTH)

# Check if the request was successful (status code 200)
if response.status_code == 200:
    # Print the response content
    print(response.text)
else:
    # Print the error message if the request was not successful
    print("Error:", response.status_code, response.text)
```