|  |  |
| --- | --- |
| **STIS**  Statistical Information Systems | Consortium  INTRASOFT INTERNATIONAL S.A.  and  AGILIS S.A. |
|  | isoft_logo_cr agilis-logo_cr |

European Commission – Eurostat/B3

‘SDMX Tools Maintenance’

**SDMX Converter CLI User Manual**

**Version 4.6**

July 2017

Document Service Data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Type of Document** | Project deliverable | | | | |
| **Reference:** | SDMX Converter CLI User Manual | | | | |
| **Issue:** | 1 | **Revision:** | 2 | **Status:** | Draft |
| **Created by:** | Flavian Ioan (FI) | | | **Date:** | 13/03/2017 |
| **Updated by:** | Dragos Balan (DB) | | | **Date:** | 15/03/2017 |
| **Distribution:** | EU-Eurostat, Agilis S.A, Intrasoft International S.A. | | | | |
| For Internal Use Only | | | | | |
| **Reviewed by:** | Dragos Balan (DB) | | | | |
| **Approved by:** | Mario Fendler (MF) | | | | |

Document Change Record

|  |  |  |
| --- | --- | --- |
| Issue/ Revision | Date | Change |
| 0.1 | 13/03/2017 | Initial version of the document. |
| 1.0 | 15/03/2017 | Document reviewed, ready for delivery. |
| 1.1 | 27/03/2017 | Changes applied for SDMXCONV-353-The documentation must indicate which values are valid for the options |
| 1.2 | 17/05/2017 | Changes for SDMXCONV-416 |
| 1.3 | 31/07/2017 | Changes applied for SDMXCONV-487 – create a Converter CLI application and documentation |
| 1.4 | 16/08/2017 | Changes for header writing |
| 1.5 | 22/09/2017 | Changes for SDMXCONV-522 (URL Registry added) |
| 1.6 | 09/11/2017 | Changes for SDMXCONV-516/SDMXCONV 524/SDMXSOURCE-9, SDMXCONV-538, SDMCONV-536, SDMXCONV-540 |
| 1.7 | 14/11/2017 | SDMXCONV-458, mentioning wsdl link for soap registry |
| 1.8 | 28/11/2017 | SDMXCONV-498, Header required by converter for EXCEL to CSV  SDMXCONV-455, csv input cannot have more columns than dsd components |
| 1.9 | 08/12/2017 | SDMXCONV-575 usage of quotes for csv files, SDMXCONV-591: SKIP parameter only for attributes, SDMXCONV-566 Reset and Change Input buttons, SDMXCONV-563 autodetection for struval dsd |
| 2.0 | 28/02/2018 | SDMXCONV-645 Access registries over SSL (HTTPS) |
| 2.1 | 22/03/2018 | SDMXCONV-639 – CSV input file validation |
| 2.2 | 16/04/2018 | SDMXCONV-636 – Enhanced excel parameter support |
| 2.3 | 05/03/2018 | SDMXCONV-660 – Conversion and Validation feature added. |
| 2.4 | 18/07/2018 | Wsdl changes for SDMXCONV-639 – CSV input file validation |
| 2.5 | 11/09/2018 | SDMXCONV-720 – Support ConceptSeparator for MIXED and OBS\_LEVEL position types for Excel  SDMXCONV-744 – Converter need to use the actual not formatted value for ENERGY excel templates |
| 2.6 | 13/09/2018 | SDMXCONV-131 - Option -help not recognized when additional arguments are present  SDMXCONV-132 - Option -version not recognized when additional arguments are present  SDMXCONV-133 - No error message when CLI is run with one single argument |
| 2.7 | 16/01/2019 | SDMXCONV-811 - Sementic error with excel file |
| 2.8 | 21/03/2019 | SDMXCONV-854 - Conversion CSV to SDMX\_CSV: missing delimiter parameter for OUTPUT file  SDMXCONV-759 - inputFileVersion available in CLI  SDMXCONV-740 - EXCEL input file based validation |
| 2.9 | 03/04/2019 | SDMXCONV-876 - Conversion from FLR to CSV not supported |
| 3.0 | 25/04/2019 | SDMXCONV-875 - Difference between struval error codes and the documentation |
|  | 05/08/2019 | SDMXCONV-951 - Parameter ErrorIfEmpty |
| 3.1 | 21/06/2019 | SDMXCONV-796 - LFs and multiple measures  SDMXCONV-797 - Support mapping of a SDMX v2.1 measure dimension’s concepts  SDMXCONV-799 - Modify the GUI & WebApp to support SDMX v2.1 measure dimension concept mappings  SDMXCONV-800 - Transcoding support for SDMX v2.1 measure dimension concepts  SDMXCONV-801 - Single Level CSV Writer support for SDMX v2.1 measure dim. concepts as columns  SDMXCONV-916 - Add support in SdmxSource for explicit measures in structure reference. Delete this link  SDMXCONV-798 - FLR Reader support for SDMX 2.1 measure dim. concepts as columns  SDMXCONV-922 - Missing documentation for "SkipIncompleteKeys"  SDMXCONV-927 - Excel should use actual value by default for numbers |
| 3.2 | 07/08/2019 | SDMXCONV-867 - SDMX CSV internal format |
| 3.3 | 13/09/2019 | SDMXCONV-865 - Batch script example to be provided in the documentation for the CLI |
| 3.4 | 20/09/2019 | SDMXCONV-980 - Sheet with name "Transport" recognized as transcoding sheet |
| 3.5 | 10/12/2019 | SDMXCONV-1025 - Conform CSV Writers with RFC4180 |
| 3.6 | 21/01/2020 | SDMXCONV-914 - STRUVAL output reports to use MSD |
| 3.6 | 21/01/2020 | SDMXCONV-1019 - STRUVAL should be able to output the old validation report and the MSD based one |
| 3.7 | 26/02/2020 | SDMXCONV-996 - make visible formats for conversion configurable in UI |
| 3.7 | 26/02/2020 | SDMXCONV-775 - Converter should be able to output FLR files |
| 3.8 | 06/04/2020 | SDMXCONV-1052 - STRUVAL reports for non UTF-8 encoding files |
| 3.9 | 22/05/2020 | SDMXCONV-1060 - MSD report, include it as an attachment with the old report in the main soap body  SDMXCONV-907 - Possibility to use formulas in external Excel parameter sheet  SDMXCONV-1031 - Energy Phase 4 (MOS%) UAT |
| 4.0 | 20/07/2020 | SDMXCONV-1095 - Converter improvements |
| 4.1 | 15/12/2020 | SDMXCONV-1134 - Issue with SDMX-CSV validation – No error raised if column does not correspond to a concept in the DSD  SDMXCONV-1135 - Issue with CSV validation – not possible to simultaneously impose column order and header names  SDMXCONV-1155 - Timeout uploading files in the webapp |
| 4.2 | 01/02/2021 | SDMXCONV-1135 - Issue with CSV validation – not possible to simultaneously impose column order and header names |
| 4.3 | 26/02/2021 | SDMXCONV-1186 - ErrorIfEmpty to be moved from instance to WS call parameter |
| 4.4 | 30/06/2021 | SDMXCONV-1205 - File without TIME\_PERIOD or OBS\_VALUE considered valid in Struval |
| 4.5 | 01/10/2021 | SDMXCONV-1187 - Async calls in STRUVAL-implementation |
| 4.6 | 20/10/2021 | SDMXCONV-1239 - Excel formula errors not reported by Struval validation |

**Table of contents Page**

1 Introduction 8

2 Arguments and options 9

3 Conversion and validation examples 17

3.1 Conversion examples 17

3.1.1 Conversion from COMPACT\_SDMX to GENERIC\_SDMX 17

Command: 17

CLI parameters: 17

3.1.2 Conversion from GENERIC\_SDMX to MULTI\_LEVEL\_CSV 17

Command: 17

CLI parameters: 17

3.1.3 Conversion from CSV to SDMX\_CSV 18

Command: 18

CLI parameters: 18

3.1.4 Conversion from GENERIC\_SDMX to MULTI\_LEVEL\_CSV 19

Command: 19

CLI parameters: 19

3.1.5 SDMX-Converter mass conversion script 20

3.2 Validation examples 20

3.2.1 A simple validation of an UTILITY\_SDMX 20

Command 20

CLI parameters explanation 20

3.2.2 A validation of a EXCEL file, including a dsd file and input parameters 21

CLI parameters explanation 21

3.2.3 A validation of a SDMX CSV file, including a dsd file and CSV input parameters (delimiter, quote characters, header row) 21

Command 21

CLI parameters explanation 22

3.3 Conversion and Validation example 22

Command 22

CLI parameters explanation 22

3.3.4 A validation of MULTI\_LEVEL\_CSV 23

Command 23

CLI parameters explanation 23

3.3.5 A validation of CSV using Web Service and allow\_additional\_columns 24

CLI parameters explanation 24

3.3.6 A validation of CSV using Web Service and validateCSVheaders 24

CLI parameters explanation 25

Annex A: Structure of a multilevel CSV/FLR 26

1.1 Multilevel files 26

1.2 Sample multilevel CSV file 27

1.3 Sample CSV file with cross sectional measures that appear in one record 27

Annex B: Examples of Excel to SDMX conversion 29

Annex C: How Excel handles decimal numbers 31

1.4 Excel adheres to the IEEE 754 standard 31

1.5 What Excel displays is NOT what is stored internally 31

Annex D: The rounding mechanism implemented in Converter 32

Annex E – CSV Input/Output escaping mechanism 33

Annex F – Converter Web Service and Struval Web Service WSDLs 34

1.6 Converter Web Service: 34

1.7 Struval Web Service: 36

1.8 Metadata Metrics 40

1.9 Converter Transformation Service 41

Annex G: Validations done by STRUVAL 43

1.10 Syntax Validation 43

1.11 Structural Validation 43

1.12 Data Validation 44

Annex H: The structure of a header file 45

Annex I: The structure of a column mapping file 47

1.13 Mapping for CSV 47

1.14 Mapping for FLR 47

1.15 Mapping for Multilevel CSV 48

1.16 Mapping for SDMX CSV 49

1.17 Fixed values example 49

1.18 Mapping for cross sectional files 49

1.19 Mapping for files containing explicit Measures 50

Annex J: The structure of an excel parameter sheet mapping file or sheet 52

Annex K: The structure of a transcoding file 53

Annex L: Conversion limitations 55

1.20 CSV/FLR limitations 55

1.21 Limitations for Message Group messages 56

1.22 Limitations when converting to/from Cross Sectional 57

1.23 Limitations when converting from SDMX 2.1 to other formats 57

1.24 Limitations when converting from Gesmes TS 57

1.25 Detection of input format 58

Annex M: How to configure the excel readers (Excel parameter sheet contents) 59

Annex N: The structure of an excel parameter file 64

ANNEX O: Supported Conversion 65

Annex P – Structure of a SDMX CSV file 68

Annex R: Examples of Validation Requests for Web Service 69

Annex S – Struval Error Messages 71

Annex T – Instructions to Access Registry over HTTPS 77

1.26 Obtain your own certificate 77

1.27 Use an existing certificate 79

Annex U – Reporting Period for SDMX 2.1 Output Formats 80

Annex V: The structure of a transcoding sheet for Excel parsing 81

Annex W: The structure of a properties file 82

Annex X: Internal SDMX CSV format 83

Annex Y: Character Encoding 84

Annex Z: How Conversion/Validation Handles Excel Formulas 85

1.1 Formulas in External Parameter Sheet 85

1.2 Manual Formulas Calculation Detection 85

# Introduction

It is possible for the Converter Web Service to be called using a command line interface (CLI). CLI interface has the following characteristics:

* Offers an alternative way for integrating Converter functionality to other applications. This pertains to applications that are not implemented in Java. Therefore they call CLI commands from the system shell.
* CLI can be utilised to perform conversions or validations in a batch-processing mode without user interaction. It can be coupled with shell scripting of the target platform for tailor-made batch processing.
* CLI requires less system resources, such as processor time and memory, to run than GUIs.
* It is not targeted to human user interaction. The GUI is intended for this purpose.
* CLI accepts the same command line arguments as standalone versions, but calls a Converter Web Service to perform the actual processing.
* When the –wsdl option is present then the CLI uses the web service for conversion/validation, otherwise it uses the Converter API.

# Arguments and options

The following tables show the list of supported arguments and options:

**CLI OPTIONS:**

|  |  |  |
| --- | --- | --- |
| **Option** | **Description** | **Mandatory** |
| **-help** | Provides access to the help menu of CLI, containing the options available and the description for each one. | No |
| **-version** | Provides the current version of the Converter CLI | No |
| **-** **inputFileVersion** | Provides the input file's format and version | No |

**CONVERSION OPTIONS:**

|  |  |  |
| --- | --- | --- |
| **Option** | **Description** | **Mandatory** |
| **-inputFile** | The absolute path of the Input file. | Yes |
| **-outputFile** | The absolute path of the Output file. When not mentioned the output is a file called “output” in the Converter CLI installation folder. The extension of the default file depends on the output format. | No |
| **-to** | The desired output format. One of the following values:  CSV,  GESMES\_TS,  GENERIC\_SDMX,  COMPACT\_SDMX,  UTILITY\_SDMX,  CROSS\_SDMX,  GENERIC\_DATA\_2\_1,  GENERIC\_TS\_DATA\_2\_1,  STRUCTURE\_SPECIFIC\_DATA\_2\_1, STRUCTURE\_SPECIFIC\_TS\_DATA\_2\_1,  EXCEL,  SDMX\_CSV,  MULTI\_LEVEL\_CSV | Yes |
| **-from** | The Input format. One of the following values:  CSV,  GESMES\_TS,  GENERIC\_SDMX,  COMPACT\_SDMX,  UTILITY\_SDMX,  CROSS\_SDMX,  GENERIC\_DATA\_2\_1,  GENERIC\_TS\_DATA\_2\_1,  STRUCTURE\_SPECIFIC\_DATA\_2\_1, STRUCTURE\_SPECIFIC\_TS\_DATA\_2\_1,  EXCEL,  MESSAGE\_GROUP,  SDMX\_CSV,  MULTI\_LEVEL\_CSV,  FLR | Yes |
| **-wsdl** | The location of the web service wsdl. | No |
| **-reg**  **true|false** | Boolean value, default: false.  Use –reg true if Registry should be used for retrieving the structure files (whose identifiers should be provided using the –dsd\_id, -dsd\_version and –dsd\_agency arguments so if –reg true the dsd\_id, dsd\_version and dsd\_agency become mandatory arguments).  If –reg false or –reg argument does not exist at all, the dsd\_file argument becomes mandatory. | No |
| **-dsd\_file** | The absolute path of the DSD file | mandatory if –reg parameter is missing or set to false |
| **-dsd\_id** | The DSD ID to be used in Registry. | No |
| **-dsd\_agency** | The DSD Agency for Registry. | No |
| **-dsd\_version** | The DSD Version for Registry. | No |
| **-df**  **true|false** | Boolean value. Must be true if dataflow information is to be used to retrieve DSD information. Default value: false. | No |
| **-df\_id** | The Dataflow ID for Registry. | No |
| **-df\_version** | The Dataflow Version for Registry. | No |
| **-df\_agency** | The Dataflow Agency for Registry. | No |
| **-header\_file** | The absolute path of the Header file. If the conversion is done with input one of the following formats (EXCEL, CSV, MULTI\_LEVEL\_CSV, SDMX\_CSV) to output one of the following formats (EXCEL, CSV, MULTI\_LEVEL\_CSV, SDMX\_CSV) the header file is not necessary as this formats do not include header information. | No |
| **-date\_format** | The date format to be used for output CSV/FLR. The value of the format can be either SDMX or GESMES. By default the option is SDMX. | No |
| **-level** | Must be a number if the file is a multilevel csv file. Applicable only for CSV/FLR input/output. Default value is 1. | No |
| **-mapping\_file** | The absolute path of the Mapping file, applicable ONLY for CSV/FLR input file. The default mapping is used if no mapping file is provided. | No |
| **-ordered\_input**  **true|false** | Boolean value. True to denote if input is ordered. Applicable for CSV/FLR input file. Default value is false. | No |
| **-trans\_file path** | The absolute path of the Transcoding file, applicable ONLY for CSV input-output file | No |
| **-delimiter** | The CSV delimiter of the input file in double quotes.  This parameter is applicable only to the CSV input type.  Examples:  -delimiter “;” (for semicolon)  -delimiter “ ” (for blank-space)  A special case is the TAB character which should be used like  -delimiter “Tab” .  Default value is (“;”) semicolon. | No |
| **-outputDelimiter** | The CSV delimiter of the output file in double quotes.  This parameter is applicable only to the CSV output type.  Examples:  -delimiter “;” (for semicolon)  -delimiter “ ” (for blank-space)  A special case is the TAB character which should be used like  -delimiter “Tab” .  Default value is (“;”) semicolon. | No |
| **-unescapeInputCSV true|false** | Boolean value. True if the input values must be quoted, false otherwise. Applicable ONLY for CSV input file. Default value: false | No |
| **-escapeOutputCSV** | Applicable ONLY for CSV output file. Accepted values: DEFAULT (escape with double quotes only the values that contain special characters), ESCAPE\_ALL (all values have double quotes), ESCAPE\_NONE (values remain as is, used if all special characters are escaped already).  Default value: DEFAULT. | No |
| **-header\_row** | Value that specifies if the csv file has a header. For csv input the values are: DISREGARD\_COLUMN\_HEADERS, USE\_COLUMN\_HEADERS, NO\_COLUMN\_HEADERS. For csv ouput the values are: NO\_COLUMN\_HEADERS, USE\_COLUMN\_HEADERS.  When USE\_COLUMN\_HEADERS is used for CSV input, then the column names from the first line of csv will be used to detect the DSD concepts. There can be more columns than the number of concepts in DSD, but only those available in DSD will be processed. The rest of the columns will be ignored. | No |
| **-header\_Writing true|false** | Boolean value. If it’s true, when converting /validating SDMX formats, in the current folder a .prop file containing the header information will be saved, that can be later used when converting CSV and EXCEL formats. This option is not available when converting/validating using the web service. Default value : false. . If the conversion is done with input one of the following formats (EXCEL, CSV, MULTI\_LEVEL\_CSV, SDMX\_CSV) to output one of the following formats (EXCEL, CSV, MULTI\_LEVEL\_CSV, SDMX\_CSV) the parameter is not allowed as this formats do not include header information. | No |
| **-nuri** | The namespace URI, applicable ONLY for SDMX Compact, SDMX Utility and SDMX Cross Sectional output files. | No |
| **-nprefix prefix** | The namespace prefix, applicable ONLY for SDMX Compact, SDMX Utility and SDMX Cross Sectional output files. | No |
| **-ges\_technique** | The Gesmes/TS writing technique. The value can be either TIME\_RANGE or SINGLE\_OBS | No |
| **-validation**  **true|false** | Boolean value. Applicable for SDMX and CSV input files. Set true to perform the validation. Default value: false. | No |
| **-mapping\_param\_file** | Applicable for EXCEL input files. Path to a file containing mapping between data sheets and parameter sheets – please see APPENDIX D for examples | No |
| **-parameter\_file** | Applicable for EXCEL input files. Path to an external parameter file – please see APPENDIX D for examples | No |
| **-excel\_template** | The excel template file used for conversion with excel output. | No |
| **-exceltype** | Type of excel either ‘xls’, ‘xlsx’, ‘xlsm’ . | No |
| **-generated\_file\_comment** | The comment to be inserted to the generated file (for SDMX files) | No |
| **-reportingYearStartDay** | Available only for SDMX 2.1 output formats. For using Reporting Period a string of format “-mm-dd”, containing the month and day the user wants to appear in the output file’s REPORTING\_YEAR\_START\_DATE attribute in the DataSet. For example: --01-01 (representing January 1st) or --12-31 (representing the 31st of December). If the parameter is not present then Gregorian time will be used for generating the output file instead Reporting Period. | No |
| **-mapMeasure** | Accepted values are ‘true’ or ‘false’.  When set it to true explicit Measures are expected in the Structure, in the mapped dimensions and in the input data. This feature is supported for FLR input files and CSV output files. | No |
| **-** **errorIfEmpty** | Accepted values are ‘true’ or ‘false’  When set to false and the provided input is not 2.1 Format and the action in the dataset is not equal to ‘DELETE’ then if the input file hasn’t got any data rows the file is considered valid. The default value for this parameter is true and in the case of an empty input file an error is raised. | No |
| **-padding** | Accepted values any character.  Every value has fixed width, padding character is the character with which the unwritten spaces will be filled. | Mandatory only if output format is FLR |

**VALIDATION OPTIONS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Option** | **Description** | | **Mandatory** |
| **-inputFile** | The absolute path of the Input file. Mandatory for Conversion and Validation. | | Yes |
| **-wsdl** | The location of the web service wsdl. | | Yes |
| **-dsd\_file** | The absolute path of the DSD file | | Yes |
| **-validation**  **true|false** | Boolean value. Set true to perform the validation. Default value: false. | | Yes |
| **-from** | The Input format. One of the following values:  CSV,  GESMES\_TS,  GENERIC\_SDMX,  COMPACT\_SDMX,  UTILITY\_SDMX,  CROSS\_SDMX,  GENERIC\_DATA\_2\_1,  GENERIC\_TS\_DATA\_2\_1,  STRUCTURE\_SPECIFIC\_DATA\_2\_1, STRUCTURE\_SPECIFIC\_TS\_DATA\_2\_1,  EXCEL,  MESSAGE\_GROUP,  SDMX\_CSV,  MULTI\_LEVEL\_CSV  If the operation is validation, the parameter is needed to distinct between different csv family formats.   * If it is not used, the validation file is considered of SDMX format; * If the input format is xml, the –from parameter is not taken into consideration; * If it is a format that does not accept validation, an error message is raised; * It is required at this point , when we have a csv family format file, to be either SDMX\_CSV, MULTI\_LEVEL\_CSV or CSV. If the parameter is not added and the input file is csv format, a message will warn the user to add it. | | No |
| **-quote\_cha** | The CSV quote character, applicable ONLY for CSV input file. Can be either a simple quote or a double quote. | | No |
| **-parameter\_file** | Applicable for EXCEL input files. Path to an external parameter file – please see APPENDIX D for examples | | No |
| **-delimiter** | The CSV delimiter of the input file in double quotes.  This parameter is applicable only to the CSV input/output type.  Examples:  -delimiter “;” (for semicolon)  -delimiter “ ” (for blank-space)  A special case is the TAB character which should be used like  -delimiter “Tab” .  Default value is (“;”) semicolon. | | No |
| **-header\_row** | The possible values for the header row parameter are:   * USE\_COLUMN\_HEADERS if in the first row of the output CSV/FLR the field names that correspond to each column should be written or the input CSV/FLR file has a first row containing the column names and it should be used. * NO\_COLUMN\_HEADERS if there is no extra first row in the input CSV/FLR file or no first header row should be written in the output CSV/FLR file * DISREGARD\_COLUMN\_HEADERS if there is a header row in the CSV/FLR but the row should be disregarded and not used in processing the input. | | No |
| **-date\_format** | The date format to be used for output CSV. The value of the format can be either SDMX or GESMES. By default the option is SDMX. | | No |
| **-level** | Must be a number if the file is a multilevel csv file. Applicable only for CSV formats input/output. Default value is 1. | | No |
| **-mapping\_file** | The absolute path of the Mapping file, applicable ONLY for CSV/FLR format input file. The default mapping is used if no mapping file is provided. | | No |
| **-output\_mapping\_file** | The absolute path of the Mapping file, applicable ONLY for CSV format output file. The default mapping is used if no mapping file is provided. | | No |
| **-ordered\_input**  **true|false** | Boolean value. True to denote if input is ordered. Applicable for CSV formats input file. Default value is false. | | No |
| **-trans\_file path** | The absolute path of the Transcoding file, applicable ONLY for CSV input-output file | | No |
| **-unescapeInputCSV true|false** | Boolean value. True if the input values must be quoted, false otherwise. Applicable ONLY for CSV input file. Default value: false | | No |
| **-registry\_url** | The url of the registry to be used (rest endpoint or wsdl link for soap). Default value is provided by the application and is set in configuration file of the application. If the site needs a java certificate to be accessed: Check Annex T – Instructions to Access Registry over Https | | No |
| **-df**  **true|false** | Boolean value. Must be true if dataflow information is to be used to retrieve DSD information. Default value: false. | | No |
| **-maxErrorNumber** | Specifies the maximum number of errors returned by the validation. | | No |
| **-mapMeasure** | Accepted values are ‘true’ or ‘false’.  When set it to true explicit Measures are expected in the Structure, in the mapped dimensions and in the input data. This feature is supported for FLR input files. | | No |
| **-allow\_additional\_columns** | | This parameter concerns all CSV flavours and it is true by default. If it is set to true, then a stricter validation is applied. E.g. a file having columns that do not correspond to anything in the DSD is being rejected. | No |
| **-** **validateCSVheaders** | | This option applies only for simple CSV file validation using mapping. We have an option to validate that each column in the header row matches the concept name mapped to it and it is in the exact same order with the corresponding column of the mapping file. It is available for use only when user selects “Use header” in the Header Row section and it is false by default. | No |
| **-** **errorIfEmpty** | | Accepted values are ‘true’ or ‘false’  When set to false and the provided input is not 2.1 Format and the action in the dataset is not equal to ‘DELETE’ then if the input file hasn’t got any data rows the file is considered valid. The default value for this parameter is true and in the case of an empty input file an error is raised. | No |
| **-expiration** | | Accepted value number. Option to set the time for which the result is kept in the server (in msecs) | No |
| **-timeout** | | Accepted value number. Time interval for which the user will wait to get their validation result in a synchronous manner (in secs) | No |
| **-getStatus** | | Accepted values are true or false.  With this option the user can request the status of a previously performed validation, providing the id of the request. If no Id is provided then a list of request Ids available on the server will be fetched. | No |
| **-uuid** | | Can be used in combination with the ‘getStatus’ option and must be set with the id of the request for which the user will get the status for. | No |
| **-callbackUrl** | | The url to send the asynchronous response of validation, this only applies for validation with Web service. | No |
| **-formulaErrorsReported** | | When set to true all formula errors of an excel input file will be reported. When set to false only formula errors which are recognized from Excel will be reported. | No |

# Conversion and validation examples

## Conversion examples

### Conversion from COMPACT\_SDMX to GENERIC\_SDMX

### Command:

converter-cli.bat -from COMPACT\_SDMX -to GENERIC\_SDMX -inputFile C:\conversion\input.xml -outputFile C:\conversion\result.xml -dsd\_file C:\conversion\UOE\_NON\_FINANCE+ESTAT+0.4.xml -wsdl http://localhost:8080/ConverterWS/ConverterWebService?wsdl

### CLI parameters:

|  |  |
| --- | --- |
| -from COMPACT\_SDMX | the format of the input file is COMPACT |
| -to GENERIC\_SDMX | the format of the output file is GENERIC |
| -inputFile C:\conversion\input.xml | This is the path of the input file ( having the COMPACT format) |
| -outputFile C:\conversion\result.xml | The path of the output file (having the GENERIC format) |
| -dsd\_file C:\conversion\UOE\_NON\_FINANCE+ESTAT+0.4.xml | The file containing the Data structure definition |
| -wsdl http://localhost:8080/ConverterWS/ConverterWebService?wsdl | The location of the wsdl file |

### Conversion from GENERIC\_SDMX to MULTI\_LEVEL\_CSV

### Command:

converter-cli.bat -from GENERIC\_SDMX -to MULTI\_LEVEL\_CSV -inputFile C:\conversion\generic.xml -dsd\_file C:\conversion\ UOE\_NON\_FINANCE+ESTAT+0.4.xml -wsdl http://localhost:8080/ConverterWS/Converter?wsdl -level 3 -outputDelimiter “;” -date\_format SDMX -header\_row true -mapping\_file C:\conversion\mapping3Level.xml

### CLI parameters:

|  |  |
| --- | --- |
| from GENERIC\_SDMX | The SDMX format of the input file |
| to CSV | The expected output format |
| inputFile C:\conversion\generic.xml | The path of the input file |
| dsd\_file C:\conversion\UOE\_NON\_FINANCE+ESTAT+0.4.xml | The path of the Data Structure Definition file |
| wsdl http://localhost:8080/ConverterWS/Converter?wsdl | The URL location of the WSDL file |
| level 3 | The CSV output will be displayed on 3 levels |
| outputDelimiter ; | The delimiter to be used in the output file |
| date\_format SDMX | The format of the dates in the output file |
| header\_row USE\_COLUMN\_HEADERS | The output file should have a header |
| mapping\_file C:\conversion\mapping3Level.xml | The mapping files needed by converter to distribute the data between levels |

### Conversion from CSV to SDMX\_CSV

### Command:

converter-cli.bat -wsdl http://localhost:8080/ConverterWebService?wsdl -to SDMX\_CSV -from CSV -inputFile C:/convFiles/COMEXT\_AGG\_M\_DE\_2017\_0009\_V0001.csv -dsd\_file C:/convFiles/ESTAT+ITGS\_AGG\_A+1.0.xml -df true -df\_agency ESTAT -df\_id ITGS\_AGG\_A -df\_version 1.0 -delimiter , -outputDelimiter ? -header\_row NO\_COLUMN\_HEADERS -outputFile C:/convFiles/854/output.csv

### CLI parameters:

|  |  |
| --- | --- |
| from CSV | The CSV format of the input file |
| to SDMX\_CSV | The expected output format |
| inputFile C:/convFiles/COMEXT\_AGG\_M\_DE\_2017\_0009\_V0001.csv | The path of the input file |
| dsd\_file C:/convFiles/ESTAT+ITGS\_AGG\_A+1.0.xml | The path of the Data Structure Definition file |
| df true | Is a DataFlow |
| -df\_agency ESTAT | The DataFlow agency for Registry |
| df\_id ITGS\_AGG\_A | The Dataflow ID for Registry |
| df\_version 1.0 | The DataFlow version for Registry |
| wsdl http://localhost:8080/ConverterWebService?wsdl | The URL location of the WSDL file |
| delimiter , | The delimiter to be used in the input file |
| outputDelimiter ? | The output delimiter to be used in the output file |
| header\_row NO\_COLUMN\_HEADERS | The output file should not have a header |

### Conversion from GENERIC\_SDMX to MULTI\_LEVEL\_CSV

### Command:

converter-cli.bat -from GENERIC\_SDMX -to MULTI\_LEVEL\_CSV -inputFile C:\conversion\generic.xml -dsd\_file C:\conversion\ UOE\_NON\_FINANCE+ESTAT+0.4.xml -wsdl http://localhost:8080/ConverterWS/Converter?wsdl -level 3 -outputDelimiter “;” -date\_format SDMX -header\_row true –output\_mapping\_file C:\conversion\mapping3Level.xml

### CLI parameters:

|  |  |
| --- | --- |
| from GENERIC\_SDMX | The SDMX format of the input file |
| to CSV | The expected output format |
| inputFile C:\conversion\generic.xml | The path of the input file |
| dsd\_file C:\conversion\UOE\_NON\_FINANCE+ESTAT+0.4.xml | The path of the Data Structure Definition file |
| wsdl http://localhost:8080/ConverterWS/Converter?wsdl | The URL location of the WSDL file |
| level 3 | The CSV output will be displayed on 3 levels |
| outputDelimiter ; | The delimiter to be used in the output file |
| date\_format SDMX | The format of the dates in the output file |
| header\_row USE\_COLUMN\_HEADERS | The output file should have a header |
| output\_mapping\_file C:\conversion\mapping3Level.xml | The mapping files needed by converter to distribute the data between levels |

### SDMX-Converter mass conversion script



The script above is used for processing multiple files in conversion. Download the .zip folder, extract the contents, adjust the path variables included in the batch script and run multiple files conversion. For more information refer to the comments of the batch script, by opening it on edit mode.

## Validation examples

### A simple validation of an UTILITY\_SDMX

### Command

converter-cli.bat -from UTILITY\_SDMX -inputFile C:\validation\utility.xml -dsd\_file C:\validation\UOE\_NON\_FINANCE+ESTAT+0.4.xml -wsdl http://localhost:8080/ConverterWS/StruvalWebService?wsdl -validation true

### CLI parameters explanation

|  |  |
| --- | --- |
| inputFile C:\validation\utility.xml | The path of the input file (the file to be validated) |
| -from UTILITY\_SDMX | The format of the file to be validated |
| dsd\_file C:\validation\UOE\_NON\_FINANCE+ESTAT+0.4.xml | The Data Structure Definition file |
| wsdl http://localhost:8080/ConverterWS/StruvalWebService?wsdl | The URL location of the wsdl file |
| validation true | Flag to trigger validation |

### A validation of a EXCEL file, including a dsd file and input parameters

converter-cli.bat -validation "true" -dsd\_file C:\ESTAT\_ENERGY\_1.2\_DFs.xml –inputFile C:\ENERGY\_ELECT\_A\_GE\_2016\_0000\_V0001.xlsm -from EXCEL -maxErrorNumber 100 -parameter\_file C:\ENERGY\_ELECT\_SDMXConverterParameters.xlsx

### CLI parameters explanation

|  |  |
| --- | --- |
| validation true | Validation trigger |
| –from EXCEL | The format of the file to be validated |
| inputFile C:\ENERGY\_ELECT\_A\_GE\_2016\_0000\_V0001.xlsm | The path of the file to be validated |
| dsd\_file C:\ESTAT\_ENERGY\_1.2\_DFs.xml | The path of the Data Structure Definition file |
| -parameter\_file C:\ENERGY\_ELECT\_SDMXConverterParameters.xlsx | External Parameter excel file that contains mapping and parameter sheets. If not present look into input excel file. |
| maxErrorNumber 100 | Validation stops after 100 errors |

### A validation of a SDMX CSV file, including a dsd file and CSV input parameters (delimiter, quote characters, header row)

### Command

converter-cli.bat -validation true –from SDMX\_CSV -inputFile C:\validation\file.csv -dsd\_file C:\validation\ESTAT+STS+2.0.xml -wsdl http://localhost:8080/ConverterWS/StruvalWebService?wsdl -delimiter “;” -header\_row true -maxErrorNumber 3

### CLI parameters explanation

|  |  |
| --- | --- |
| validation true | Validation trigger |
| –from SDMX\_CSV | The format of the file to be validated |
| inputFile C: \validation\file.csv | The path of the file to be validated |
| dsd\_file C: \validation\ESTAT+STS+2.0.xml | The path of the Data Structure Definition file |
| wsdl http://localhost:8080/ConverterWS/StruvalWebService?wsdl | The URL location of the wsdl file |
| delimiter ; | The delimiter of the input file |
| header\_row USE\_COLUMN\_HEADERS | The input file has a header |
| maxErrorNumber 3 | Validation stops after 3 errors |

## Conversion and Validation example

### Command

converter-cli.bat -delimiter ";" -validation "true" -dsd\_file ESTAT-NA\_SU-1.2.xml -inputFile NASU\_1719PR\_A\_DE\_2011\_0000\_V0002.xml -outputFile out.csv -from COMPACT\_SDMX -to CSV

### CLI parameters explanation

|  |  |
| --- | --- |
| validation true | Validation trigger |
| –from COMPACT\_SDMX | The format of the file to be validated |
| inputFile NASU\_1719PR\_A\_DE\_2011\_0000\_V0002.xml | The path of the file to be validated |
| dsd\_file  ESTAT-NA\_SU-1.2.xml | The path of the Data Structure Definition file |
| outputDelimiter ; | The delimiter of the output file |
| -to CSV | Conversion to csv format |
| -outputFile | The path for the output file out.csv |

To trigger validation and conversion set -validation to “true” and set parameter –to to the format you wish to convert the input file and define the output file path. If the validation of the input file is not permitted because of the format of the file then validation of the output will be performed. This feature is not possible with the usage of the wsdl.

### A validation of MULTI\_LEVEL\_CSV

### Command

converter-cli.bat -validation true -from MULTI\_LEVEL\_CSV -inputFile C:\validation\multilevel\_csv\_validation\case3.csv -dsd\_file C:\validation\multilevel\_csv\_validation\UOE\_NON\_FINANCE+ESTAT+0.4+Decimal\_Obs\_value.xml -mapping\_file C:\validation\multilevel\_csv\_validation\mapping3Levels.xml -delimiter ; -level 3 -header\_row NO\_COLUMN\_HEADERS -wsdl http://localhost:8080/converter-ws/StruvalWebService?wsdl

### CLI parameters explanation

|  |  |
| --- | --- |
| -inputFile C:\validation\case3.csv | The path of the input file (the file to be validated) |
| -from MULTI\_LEVEL\_CSV | The format of the file to be validated |
| -dsd\_file C:\validation\multilevel\_csv\_validation\  UOE\_NON\_FINANCE+ESTAT+0.4+Decimal\_Obs\_value.xml | The Data Structure Definition file |
| -wsdl http://localhost:8080/ConverterWS/StruvalWebService?wsdl | The URL location of the wsdl file |
| -validation true | Flag to trigger validation |
| -delimiter ; | The delimiter of the input file |
| -header\_row NO\_COLUMN\_HEADERS | The input file has no header |
| -level 3 | The level number of the multi level csv |
| -maxErrorNumber 3 | Validation stops after 3 errors |
| -mapping\_file | The mapping file used in the validation, for a multi level csv the mapping is mandatory |

### A validation of CSV using Web Service and allow\_additional\_columns

converter-cli.bat -validation "true" -allow\_additional\_columns false -inputFile "C:\validation\myCsv.csv" -dsd\_file "C:\validation\myDsd.xml" -dsd\_agency "ESTAT" -dsd\_id "RESPER" -dsd\_version "1.0" -from CSV -delimiter ";" -header\_row USE\_COLUMN\_HEADERS -maxErrorNumber 100 -date\_format "SDMX" -level 1 -ordered\_input "true" -wsdl http://localhost:8080/ConverterWSTomcat/StruvalWebService?wsdl

### CLI parameters explanation

|  |  |
| --- | --- |
| -inputFile C:\validation\myCsv.csv | The path of the input file (the file to be validated) |
| -from CSV | The format of the file to be validated |
| -dsd\_file C:\validation\myDsd.xml | The Data Structure Definition file |
| -wsdl http://localhost:8080/ConverterWS/StruvalWebService?wsdl | The URL location of the wsdl file |
| -validation true | Flag to trigger validation |
| -delimiter ; | The delimiter of the input file |
| -header\_row USE\_COLUMN\_HEADERS | The input file has header row |
| -level 1 | The level number of the csv |
| -maxErrorNumber 100 | Validation stops after 100 errors |
| -allow\_additional\_columns false | This parameter concerns all CSV flavours and it is true by default. If it is set to true, then a stricter validation is applied. E.g. a file having columns that do not correspond to anything in the DSD is being rejected. |

### A validation of CSV using Web Service and validateCSVheaders

converter-cli.bat -validation "true" - validateCSVheaders true -inputFile "C:\validation\myCsv.csv" -dsd\_file "C:\validation\myDsd.xml" -dsd\_agency "ESTAT" -dsd\_id "RESPER" -dsd\_version "1.0" -from CSV -delimiter ";" -header\_row USE\_COLUMN\_HEADERS -maxErrorNumber 100 -date\_format "SDMX" -level 1 -ordered\_input "true" -mapping\_file " C:\validation\mapping.xml" -wsdl http://localhost:8080/ConverterWSTomcat/StruvalWebService?wsdl

### CLI parameters explanation

|  |  |
| --- | --- |
| -inputFile C:\validation\myCsv.csv | The path of the input file (the file to be validated) |
| -from CSV | The format of the file to be validated |
| -dsd\_file C:\validation\myDsd.xml | The Data Structure Definition file |
| -mapping\_file C:\validation\mapping.xml | The column mapping file |
| -wsdl http://localhost:8080/ConverterWS/StruvalWebService?wsdl | The URL location of the wsdl file |
| -validation true | Flag to trigger validation |
| -delimiter ; | The delimiter of the input file |
| -header\_row USE\_COLUMN\_HEADERS | The input file has header row |
| -level 1 | The level number of the csv |
| -maxErrorNumber 100 | Validation stops after 100 errors |
| -validateCSVheaders true | This option applies only for simple CSV file validation using mapping. We have an option to validate that each column in the header row matches the concept name mapped to it and it is in the exact same order with the corresponding column of the mapping file. It is available for use only when user selects “Use header” in the Header Row section and it is false by default. |

# Annex A: Structure of a multilevel CSV/FLR

In this section the structure of a CSV/FLR file that is multilevel will be described.

## Multilevel files

Flat CSV/FLR files imply that they will contain only one level information, i.e. Observation level. Thus, all records are equivalent in terms of structure and semantic. As the sample below all records have the same number of columns (delimiters) with equivalent semantic (per column position), where the first column (with value M is the FREQ - Frequency), the second (values: GR, IT is the REF\_AREA (Reference Area), etc.

M,GR,W,PROD,NS0020,1,2000,200501,1.1,F,A

M,GR,W,PROD,NS0020,1,2000,200502,2.2,F,A

………………………

M,IT,W,PROD,NS0020,1,2000,200501,1.2,F,A

M,IT,W,PROD,NS0020,1,2000,200502,2.5,F,A

The above sample CSV flat file includes all key values (values for all dimensions, including time), as well as the observed value (i.e. 1.2 and 2.5) and its attached attributes (i.e.F,A).   
  
There are cases though, that a CSV file may contain a multi-level representation in order to express metadata that concern a set/group of records. For example:

1,Dataset Attribute1 value,Dataset Attribute2 value

2,GR,W,PROD,NS0020,1,2000,PC,0,2,Elements of the full national

3,M,A,A,P1M

4,200501,1.1,F,A

4,200502,2.2,F,A

4,200503,3.3,F,A

...

3,A,A,A,P1Y

4,2005,123,F,A

4,2006,456,F,A

...

2,GR,W,PROD,NS0021,1,2000,PC,0,2,Elements of the full national

3,M,A,A,P1M

4,200501,1.1,F,A

4,200502,2.2,F,A

4,200503,3.3,F,A

...

3,A,A,A,P1Y

4,2005,123,F,A

4,2006,456,F,A

The prerequisites in the above example are:

* The first column is considered a level identifier, i.e. it defines the level of the record.
* The order of records is significant, since each level is considered a child of the previously higher level encountered within the CSV file.
* The number of columns (delimiters) per level is equal
* The column semantic per level is equivalent

In this case the key values may be split among the different levels.

## Sample multilevel CSV file

In order to convert from or to a multilevel CSV/FLR file the number of the levels and the appropriate mapping file should be provided. Also the argument that defines if the input file is ordered or not should be set to true. There do not exist any constraints and the conversion to/from all SDMX formats can be done.

Bellow is presented an example with a conversion from a multilevel file to a SDMX message. The mapping in Figure 30: Mapping for a multilevel CSV file is applicable for the following csv file, which describes demographic data for the year 2005.

1;RQFI05V1;1;A

2;FI;F;ADJT;0;0;P1Y;PERS

3;2005;35;P

1;RQFI05V1;1;M

2;FI;F;DEATHST;0;0;P1Y;PERS

3;2005;23871;P

1;RQFI05V1;1;Q

2;FI;F;LBIRTHST;0;0;P1Y;PERS

3;2005;28345;P

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Source file | Output file | Command | Description/Details/Comments |
| Example 1 |  |  | converter.bat -header\_file "D:\Test Case 8\header.prop" -dsd\_file "D:\Test Case 8\ESTAT+DEMOGRAPHY+2.1.xml" -mapping\_file "D:\Test Case 8\30.3\_mapping.xml" -ordered\_input true -level 3 -delimiter ; "D:\Test Case 8\IN\_CSV.csv" "D:\Test Case 8\OUT\_Cross.xml" CSV CROSS\_SDMX | A conversion from a CSV multilevel file to a Cross Sectional file with mapping file “30.3\_mapping.xml”, header file “header.prop” and number of levels=3. |

## Sample CSV file with cross sectional measures that appear in one record

When converting from a CSV file using a Cross Sectional DSD, the cross sectional measures that belong to a series might appear in one record rather than in consecutive rows. The following two file structures might appear when converting with a cross sectional DSD.

In the first structure each observation value appears along with the cross sectional measure in one row. Each cross sectional measure appears in the same row in each record (4th column). This has been defined in the mapping procedure where the measure dimension is always mapped to the specific column. In the following sample for the Country Greece (GR) and for female (F) the annual (A) observation values are reported consecutively in rows and correspond to Deaths (DEATHST), Emigrants (EMIGT), Immigrants (IMMIT) and Births (LBIRTHST) respectively.

A;GR;F;DEATHST;2005;23871;P

A;GR;F;EMIGT;2005;6331;P

A;GR;F;IMMIT;2005;10581;P

A;GR;F;LBIRTHST;2005;28345;P

In the second file structure the components that correspond to cross sectional measures do not appear in the CSV, but only their values.

A;GR;F; 23871; 6331;10581;28345;2005;P

In the above record of the CSV file only the observation values that correspond to the cross sectional measure appear. Specifically the value 23871 is the measure of Deaths, the value 6331 of Emigrants, the value of 10581 of Immigrants and the value 28345 of Births. In the mapping procedure, the user should select to map cross sectional measures (please see Figure 23: Map cross sectional measures) and map each measure to a column in the mapping dialog. For this example the measure DEATHST has been mapped to column 4, the measure EMIGT to column 5, the measure IMMIT to column 6 and the measure LBIRTHST to column 7. In that way the user does not have to map the observation value as this will not appear in a specific column in the CSV file but will appear in the column that the respective cross sectional measure has been mapped.

# Annex B: Examples of Excel to SDMX conversion

In this section few examples related to Excel to SDMX conversion are described.

To use a mapping between data sheets and parameter sheets of an Excel file there are several ways. You can provide (in web service and cli) an external file that will look like *MappingXlsParameters.txt,* and matches the names of the data sheets and the names of the parameters sheets, you can attach a sheet inside excel file with the name *Parameter\_mapping* (see Fig.1), or you can attach a parameters mapping sheet inside the external excel file that contains only Parameters sheets and mapping sheet. With the parameter mapping sheet you can provide different parameter sheet for the same Data Sheet. The parameter mapping sheet must contain on the first row the column headers. If a parameter mapping is provided then converter no longer ignores hidden sheets if any, all data sheets that are present in the mapping will be converted.

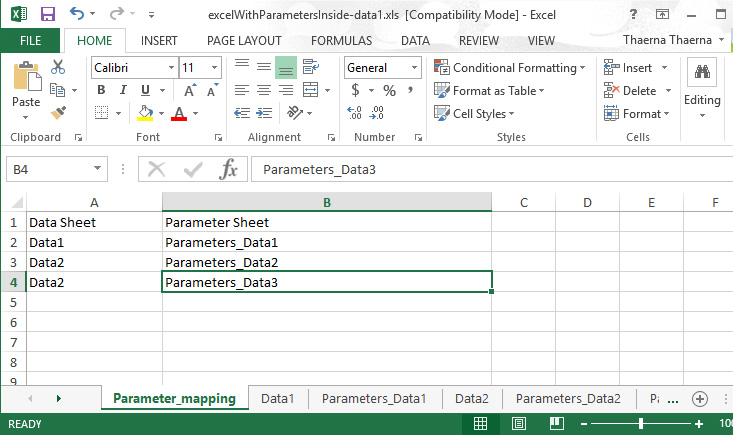


Fig. 1

The attached 0101QY.xlsx will be used with input format Excel. It has two data sheets 0101 QY V and 0101 QY L and only one parameter sheet. In this case there will be no need for an external parameter file and also no mapping parameter is needed as the existing parameter sheet will be used to process all data sheets. But if an external parameter and mapping file is provided then the external file will be used.



ExcelData.xlsx has two data sheets and two parameter sheets.

If no mapping and no external parameter file is provided then the first parameter sheet – Parameter\_Data1 is used. If an external parameter file (e.g. the attached *paramsExternal.xlsx*) is given, but no mapping, then the external parameter file is used.

If a mapping between data sheets and parameter sheets is provided then the parameter sheets are used according to the mapping. In *MappingXlsParameters.txt* Data1 will be processed using description from Parameters\_Data1 and Data2 with Parameters\_Data2.

If no mapping and no external parameter file is provided then the first parameter sheet (in our example Parameter\_Data1) is used. If an external parameter file is given (e.g. the attached *paramsExternal.xlsx*) which contains parameter sheets and a mapping then the external file is used. If the external file is given but it contains only a mapping file then the mapping from the external file is used and searches for the parameter sheets inside the excel. If the external parameter file contains only parameter sheet and no mapping then we search inside the excel input file for a mapping that uses the parameter sheets from the external file.

In general the following priority is used for the mapping of the parameter sheets:

**1st (the highest priority):** read the mapping from external file or UI (mutually exclusive) even if a parameter mapping sheet exists inside excel the mapping from the external file will be used.  
**2nd:** read the mapping from excel (sheet with the name *Parameter\_mapping*).  
**3rd (the lowest priority):** default mapping (the first Parameter sheet will be used for all non-"val" sheets).

# Annex C: How Excel handles decimal numbers

## Excel adheres to the IEEE 754 standard

This means that Excel is not always accurate when performing floating point computations (see [this](http://support.microsoft.com/kb/78113) link from Microsoft).

## What Excel displays is NOT what is stored internally

Excel stores numbers differently that you may have them formatted on the worksheet. Under normal circumstances, Excel stores numeric values as "Double Precision Floating Point" numbers, or "Doubles" for short. These are 8-byte variables that can store numbers accurate to approximately 15 decimal places. You may have only two decimal places displayed on the worksheet, but the underlying value has the full 15 decimal places. (Paragraph taken from [here](http://www.cpearson.com/excel/rounding.htm))

When a certain value is obtained as the computation of several other cells, due to the floating point arithmetic (see point 1), Excel displays rounded values of what it stores internally. If you export the worksheet to Office 2003 XML format, you will see the real value stored.

Example: Make the following computation in Excel: 4.1 – 4 and copy-paste the result into another worksheet The result will be shown in the formula bar as 0.099999999999 but the cell will contain the value 0.1

Any API reading excel will have access to the internal stored value that’s why we have to apply a rounding algorithm when reading values from Excel.

# Annex D: The rounding mechanism implemented in Converter

As we’ve seen in [Appendix E](#_APPENDIX_E_–), Excel uses the IEEE 754 standard for handling decimal number arithmetic. Due to this standard, for some computations, Excel uses internally approximations of the mathematical results (i.e. for a simple operation like 4.1 - 4 Excel stores internally the result 0.0999999999994). Knowing that this approximation is not at all user-friendly, Excel performs a rounding algorithm to improve the displaying of Numbers (i.e. as expected for the formula 4.1 - 4 the displayed result is 0.1 and this is because of the rounding not due to the correct mathematical computation).

Since Excel exports to any API (including the JAVA APIs) the internal stored values (i.e. the approximations) instead of the displayed values (the rounded ones), SDMX Converter is forced to use a rounding mechanism itself in order to display the same numbers as those shown inside Excel. Tests have shown that the HALF\_EVEN algorithm provides the best results.

[HALF\_EVEN](http://docs.oracle.com/javase/7/docs/api/java/math/RoundingMode.html#HALF_EVEN) is the rounding mode to round towards the "nearest neighbour" unless both neighbours are equidistant, in which case, round towards the even neighbour. It behaves as for [HALF\_UP](http://docs.oracle.com/javase/7/docs/api/java/math/RoundingMode.html#HALF_UP) if the digit to the left of the discarded fraction is odd and it behaves as for [HALF\_DOWN](http://docs.oracle.com/javase/7/docs/api/java/math/RoundingMode.html#HALF_DOWN) if it's even.

This is the rounding mode that statistically minimizes cumulative error when applied repeatedly over a sequence of calculations. It is sometimes known as "Banker's rounding," and is chiefly used in the USA. This rounding mode is analogous to the rounding policy used for float and double arithmetic in Java.

Some rounding examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **Excel** | **Converter** | | |
| Number as provided by Excel | Converted using default rounding precision (= 6) | Converted using rounding precision 0 | Converted using rounding precision 15 |
| 1.2 | 1.2 | 1 | 1.2 |
| 1.49 | 1.49 | 1 | 1.49 |
| 1.5 | 1.5 | 2 | 1.5 |
| 577658.06999999995 | 577658.07 | 577658 | 577658.06999999995 |

As shown in the table above,if user sets the precision too high, converter will display the values as they come from Excel (i.e. if the user sets the precision to 15 then we cannot hide the 0.999-like values which we are trying to avoid through this rounding).

# Annex E – CSV Input/Output escaping mechanism

For CSV Input:

CSV values with quotes, e.g "2010":

* If ‘Has Double Quotes’ is selected (in the GUI or Webapp applications ) or ‘UnescapeCSVInput’ (in Web Service or CLI applications) set to true, it means the CSV has quotes and they will be eliminated. Therefore the output will be 2010.
* If ‘Has Double Quotes’ is not selected (in the GUI or Webapp applications ) or ‘UnescapeCSVInput’ (in Web Service or CLI applications) set to false, it means the CSV has quotes but the user says there are no quotes, therefore an Exception will be thrown, warning the user about this discrepancy: "If the CSV values have quotes, please check 'Has double quotes' or set 'UnescapeCSVInput' to true".

CSV values without quotes, e.g 2010 :

* If ‘Has Double Quotes’ is selected (in the GUI or Webapp applications ) or ‘UnescapeCSVInput’ (in Web Service or CLI applications) set to true, an exception will be thrown warning the user that the CSV values do not include any quotes: "If the CSV values does not have quotes, please uncheck 'Has double quotes' or set 'UnescapeCSVInput' to false".
* If ‘Has Double Quotes’ is not selected (in the GUI or Webapp applications ) or ‘UnescapeCSVInput’ (in Web Service or CLI applications) set to false, it means the csv input is expected not to have quotes and the values will just be passed to the output. In this case, the output will contain the value: 2010.

For CSV Output:

* If ‘Use Double Quotes’ (in the GUI or Webapp applications) is set to “Only when needed” then when a value contains a special character then the value is exported with double quotes. When is set to “Use double quotes” then all values will have double quotes.
* For Web Service and client application we use the parameter EscapeCSVOutput, which can be set with the following values:

1. DEFAULT (escape with double quotes only the values that contain special characters)
2. ESCAPE\_ALL (all values have double quotes)

From v8.3.0 of Converter and on whenever ESCAPE\_NONE is used as an option for EscapeCSVOutput parameter, Converter would automatically translate this to DEFAULT for backwards compatibility reasons.

# Annex F – Converter Web Service and Struval Web Service WSDLs

## Converter Web Service:





## Struval Web Service:

*<?***xml version="1.0" encoding="UTF-8"***?>*<**wsdl:definitions name="StruvalWebServiceService"  
 targetNamespace="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3"  
 xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"  
 xmlns:tns="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3"  
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"  
 xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://www.w3.org/2007/05/addressing/metadata http://www.w3.org/2007/05/addressing/metadata"**>  
 <**wsdl:types**>  
 <**schema xmlns="http://www.w3.org/2001/XMLSchema"**>  
 <**import namespace="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3"  
 schemaLocation="StruvalWebService\_schema3.xsd"**/>  
 </**schema**>  
 </**wsdl:types**>  
 <**wsdl:message name="CustomValidateDataResponse"**>  
 <**wsdl:part name="parameters" element="tns:CustomValidateDataRespMsg"**>  
 </**wsdl:part**>  
 </**wsdl:message**>  
 <**wsdl:message name="ExplainErrorResponse"**>  
 <**wsdl:part name="parameters" element="tns:ExplainErrorRespMsg"**>  
 </**wsdl:part**>  
 </**wsdl:message**>  
 <**wsdl:message name="Exception"**>  
 <**wsdl:part name="Exception" element="tns:Exception"**>  
 </**wsdl:part**>  
 </**wsdl:message**>  
 <**wsdl:message name="ValidateDataResponse"**>  
 <**wsdl:part name="parameters" element="tns:ValidateDataRespMsg"**>  
 </**wsdl:part**>  
 </**wsdl:message**>  
 <**wsdl:message name="ExplainError"**>  
 <**wsdl:part name="parameters" element="tns:ExplainError"**>  
 </**wsdl:part**>  
 </**wsdl:message**>  
 <**wsdl:message name="GetWsVersionResponse"**>  
 <**wsdl:part name="parameters" element="tns:GetWsVersionResponse"**>  
 </**wsdl:part**>  
 </**wsdl:message**>  
 <**wsdl:message name="GetVersionResponse"**>  
 <**wsdl:part name="parameters" element="tns:GetVersionRespMsg"**>  
 </**wsdl:part**>  
 </**wsdl:message**>  
 <**wsdl:message name="CustomValidateData"**>  
 <**wsdl:part name="parameters" element="tns:CustomValidateData"**>  
 </**wsdl:part**>  
 </**wsdl:message**>  
 <**wsdl:message name="ValidateData"**>  
 <**wsdl:part name="parameters" element="tns:ValidateData"**>  
 </**wsdl:part**>  
 </**wsdl:message**>  
 <**wsdl:message name="GetWsVersion"**>  
 <**wsdl:part name="parameters" element="tns:GetWsVersion"**>  
 </**wsdl:part**>  
 </**wsdl:message**>  
 <**wsdl:message name="GetVersion"**>  
 <**wsdl:part name="parameters" element="tns:GetVersion"**>  
 </**wsdl:part**>  
 </**wsdl:message**>  
 <**wsdl:portType name="StruvalDS"**>  
 <**wsdl:operation name="CustomValidateData"**>  
 <**wsdl:input name="CustomValidateData" message="tns:CustomValidateData"  
 wsam:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/CustomValidateDataReqMsg"  
 wsaw:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/CustomValidateDataReqMsg"**>  
 </**wsdl:input**>  
 <**wsdl:output name="CustomValidateDataResponse" message="tns:CustomValidateDataResponse"  
 wsam:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/CustomValidateDataRespMsg"  
 wsaw:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/CustomValidateDataRespMsg"**>  
 </**wsdl:output**>  
 <**wsdl:fault name="Exception" message="tns:Exception"  
 wsam:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/Exception"  
 wsaw:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/Exception"**>  
 </**wsdl:fault**>  
 </**wsdl:operation**>  
 <**wsdl:operation name="ExplainError"**>  
 <**wsdl:input name="ExplainError" message="tns:ExplainError"  
 wsam:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/ExplainErrorReqMsg"  
 wsaw:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/ExplainErrorReqMsg"**>  
 </**wsdl:input**>  
 <**wsdl:output name="ExplainErrorResponse" message="tns:ExplainErrorResponse"  
 wsam:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/ExplainErrorRespMsg"  
 wsaw:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/ExplainErrorRespMsg"**>  
 </**wsdl:output**>  
 <**wsdl:fault name="Exception" message="tns:Exception"  
 wsam:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/Exception"  
 wsaw:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/Exception"**>  
 </**wsdl:fault**>  
 </**wsdl:operation**>  
 <**wsdl:operation name="ValidateData"**>  
 <**wsdl:input name="ValidateData" message="tns:ValidateData"  
 wsam:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/ValidateDataReqMsg"  
 wsaw:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/ValidateDataReqMsg"**>  
 </**wsdl:input**>  
 <**wsdl:output name="ValidateDataResponse" message="tns:ValidateDataResponse" wsam:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/ValidateDataRespMsg"  
 wsaw:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/ValidateDataRespMsg"**>  
 </**wsdl:output**>  
 <**wsdl:fault name="Exception" message="tns:Exception"  
 wsam:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/Exception"  
 wsaw:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/Exception"**>  
 </**wsdl:fault**>  
 </**wsdl:operation**>  
 <**wsdl:operation name="GetVersion"**>  
 <**wsdl:input name="GetVersion" message="tns:GetVersion"  
 wsam:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/GetVersionReqMsg"  
 wsaw:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/GetVersionReqMsg"**>  
 </**wsdl:input**>  
 <**wsdl:output name="GetVersionResponse" message="tns:GetVersionResponse"  
 wsam:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/GetVersionRespMsg"  
 wsaw:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/GetVersionRespMsg"**>  
 </**wsdl:output**>  
 <**wsdl:fault name="Exception" message="tns:Exception"  
 wsam:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/Exception"  
 wsaw:Action="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/Exception"**>  
 </**wsdl:fault**>  
 </**wsdl:operation**>  
 <**wsdl:operation name="GetWsVersion"**>  
 <**wsdl:input name="GetWsVersion" message="tns:GetWsVersion"**>  
 </**wsdl:input**>  
 <**wsdl:output name="GetWsVersionResponse" message="tns:GetWsVersionResponse"**>  
 </**wsdl:output**>  
 <**wsdl:fault name="Exception" message="tns:Exception"**>  
 </**wsdl:fault**>  
 </**wsdl:operation**>  
 </**wsdl:portType**>  
 <**wsdl:binding name="StruvalWebServiceServiceSoapBinding" type="tns:StruvalDS"**>  
 <**soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"**/>  
 <**wsdl:operation name="CustomValidateData"**>  
 <**soap:operation soapAction="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/CustomValidateDataReqMsg"  
 style="document"**/>  
 <**wsdl:input name="CustomValidateData"**>  
 <**soap:body use="literal"**/>  
 </**wsdl:input**>  
 <**wsdl:output name="CustomValidateDataResponse"**>  
 <**soap:body use="literal"**/>  
 </**wsdl:output**>  
 <**wsdl:fault name="Exception"**>  
 <**soap:fault name="Exception" use="literal"**/>  
 </**wsdl:fault**>  
 </**wsdl:operation**>  
 <**wsdl:operation name="ExplainError"**>  
 <**soap:operation soapAction="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/ExplainErrorReqMsg"  
 style="document"**/>  
 <**wsdl:input name="ExplainError"**>  
 <**soap:body use="literal"**/>  
 </**wsdl:input**>  
 <**wsdl:output name="ExplainErrorResponse"**>  
 <**soap:body use="literal"**/>  
 </**wsdl:output**>  
 <**wsdl:fault name="Exception"**>  
 <**soap:fault name="Exception" use="literal"**/>  
 </**wsdl:fault**>  
 </**wsdl:operation**>  
 <**wsdl:operation name="ValidateData"**>  
 <**soap:operation soapAction="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/ValidateDataReqMsg"  
 style="document"**/>  
 <**wsdl:input name="ValidateData"**>  
 <**soap:body use="literal"**/>  
 </**wsdl:input**>  
 <**wsdl:output name="ValidateDataResponse"**>  
 <**soap:body use="literal"**/>  
 </**wsdl:output**>  
 <**wsdl:fault name="Exception"**>  
 <**soap:fault name="Exception" use="literal"**/>  
 </**wsdl:fault**>  
 </**wsdl:operation**>  
 <**wsdl:operation name="GetVersion"**>  
 <**soap:operation soapAction="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3/GetVersionReqMsg"  
 style="document"**/>  
 <**wsdl:input name="GetVersion"**>  
 <**soap:body use="literal"**/>  
 </**wsdl:input**>  
 <**wsdl:output name="GetVersionResponse"**>  
 <**soap:body use="literal"**/>  
 </**wsdl:output**>  
 <**wsdl:fault name="Exception"**>  
 <**soap:fault name="Exception" use="literal"**/>  
 </**wsdl:fault**>  
 </**wsdl:operation**>  
 <**wsdl:operation name="GetWsVersion"**>  
 <**soap:operation soapAction="" style="document"**/>  
 <**wsdl:input name="GetWsVersion"**>  
 <**soap:body use="literal"**/>  
 </**wsdl:input**>  
 <**wsdl:output name="GetWsVersionResponse"**>  
 <**soap:body use="literal"**/>  
 </**wsdl:output**>  
 <**wsdl:fault name="Exception"**>  
 <**soap:fault name="Exception" use="literal"**/>  
 </**wsdl:fault**>  
 </**wsdl:operation**>  
 </**wsdl:binding**>  
 <**wsdl:service name="StruvalWebService"**>  
 <**wsdl:port name="StruvalDSService" binding="tns:StruvalWebServiceServiceSoapBinding"**>  
 <**soap:address location="http://localhost:9090/StruvalWebService"**/>  
 </**wsdl:port**>  
 </**wsdl:service**>  
</**wsdl:definitions**>

## Metadata Metrics



## Converter Transformation Service

<WL5G3N2:definitions xmlns="" xmlns:WL5G3N0="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:WL5G3N1="http://www.w3.org/ns/ws-policy" xmlns:WL5G3N2="http://schemas.xmlsoap.org/wsdl/" xmlns:WL5G3N3="http://xmlns.ec.eu/DataService/Core/TransformationService/V2/" xmlns:WL5G3N4="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" name="TransformationService" targetNamespace="http://xmlns.ec.eu/DataService/Core/TransformationService/V2/">  
 <wsp:UsingPolicy WL5G3N2:Required="true"/>  
 <wsp:Policy WL5G3N0:Id="Mtom.xml">  
 <wsoma:OptimizedMimeSerialization xmlns:wsoma="http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization"/>  
 </wsp:Policy>  
 <WL5G3N1:Policy WL5G3N0:Id="TransformationStreamingPortBinding\_MTOM\_Policy-TransformationStreamingPortBinding\_MTOM\_Policy">  
 <wsoma:OptimizedMimeSerialization xmlns:wsoma="http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization"/>  
 </WL5G3N1:Policy>  
 <WL5G3N2:types>  
 <xsd:schema xmlns:WL5G3N0="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:WL5G3N1="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:WL5G3N2="http://www.w3.org/ns/ws-policy" xmlns:WL5G3N3="http://schemas.xmlsoap.org/wsdl/" xmlns:WL5G3N4="http://xmlns.ec.eu/DataService/Core/TransformationService/V2/" xmlns:WL5G3N5="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://xmlns.ec.eu/DataService/Core/TransformationService/V2/" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wsp1\_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wssutil="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:xsd="http://www.w3.org/2001/XMLSchema">  
 <xsd:import namespace="http://xmlns.ec.eu/DataService/Core/TransformationService/V2/" schemaLocation="http://localhost:8080/converter\_ws/ConverterHTI?xsd=5"/>  
 </xsd:schema>  
 </WL5G3N2:types>  
 <WL5G3N2:message name="TransformReqMsgType">  
 <WL5G3N2:part element="WL5G3N3:TransformReqMsgType" name="parameters"/>  
 </WL5G3N2:message>  
 <WL5G3N2:message name="TransformRespMsgType">  
 <WL5G3N2:part element="WL5G3N3:TransformRespMsgType" name="parameters"/>  
 </WL5G3N2:message>  
 <WL5G3N2:portType name="TransformationService">  
 <WL5G3N2:operation name="transform">  
 <WL5G3N2:input message="WL5G3N3:TransformReqMsgType"/>  
 <WL5G3N2:output message="WL5G3N3:TransformRespMsgType"/>  
 </WL5G3N2:operation>  
 </WL5G3N2:portType>  
 <WL5G3N2:binding name="TransformationStreamingPortBinding" type="WL5G3N3:TransformationService">  
 <WL5G3N1:Policy>  
 <WL5G3N1:PolicyReference URI="#TransformationStreamingPortBinding\_MTOM\_Policy-TransformationStreamingPortBinding\_MTOM\_Policy"/>  
 <WL5G3N1:PolicyReference URI="#Mtom.xml"/>  
 </WL5G3N1:Policy>  
 <WL5G3N4:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>  
 <WL5G3N2:operation name="transform">  
 <WL5G3N4:operation/>  
 <WL5G3N2:input>  
 <WL5G3N4:body use="literal"/>  
 </WL5G3N2:input>  
 <WL5G3N2:output>  
 <WL5G3N4:body use="literal"/>  
 </WL5G3N2:output>  
 </WL5G3N2:operation>  
 </WL5G3N2:binding>  
 <WL5G3N2:service name="TransformationService">  
 <WL5G3N2:port binding="WL5G3N3:TransformationStreamingPortBinding" name="TransformationStreamingPort">  
 <WL5G3N4:address location="http://localhost:8080/converter\_ws/ConverterHTI"/>  
 </WL5G3N2:port>  
 </WL5G3N2:service>  
</WL5G3N2:definitions>

# Annex G: Validations done by STRUVAL

## Syntax Validation

The syntax validation is done in a non-recoverable manner by using a standard XML parser.

## Structural Validation

The following recoverable checks are performed against the SDMX message:

|  |  |
| --- | --- |
| **Error description** | Example |
| **Illegal text** | </message:Header>  **BLAH**  <message:DataSet > |
| **illegal attribute** | <message:DataSet **dummy="yes"** |
| **illegal element** | **<Unexpected a="b"></Unexpected>** |
| **series under series** | <Series ADJUSTMENT="W" FREQ="Q" REF\_AREA="IT" STS\_INDICATOR="PROD" STS\_INSTITUTION="1" STS\_ACTIVITY="NS0020" STS\_BASE\_YEAR="2000" TIME\_FORMAT="P1M">   **<Series** |
| **obs under obs** | <Obs TIME\_PERIOD="2005-08" OBS\_VALUE="1.53" OBS\_CONF="F" OBS\_STATUS="A" >  <Obs TIME\_PERIOD="2005-12" OBS\_VALUE="1.54" OBS\_CONF="F" OBS\_STATUS="A" > |
| **group under series (2.1 specific)** | </Series>  <Group xsi:type="ns |
| **Non-flat with flat contents** | <message:Structure structureID="ESTAT\_STS\_2\_2" namespace="urn:sdmx:org.sdmx.infomodel.datastructure.DataStructure=ESTAT:STS(2.2):ObsLevelDim:TIME\_PERIOD" **dimensionAtObservation="TIME\_PERIOD"**>  …  <Obs ADJUSTMENT="W" FREQ="A" REF\_AREA="IT" STS\_INDICATOR="PROD" STS\_INSTITUTION="1" STS\_ACTIVITY="NS0040" STS\_BASE\_YEAR="2000" TIME\_FORMAT="P1Y" TIME\_PERIOD="2004-02" OBS\_VALUE="1.2004" OBS\_CONF="F" OBS\_STATUS="A" /> |
| **Flat file with non-flat contents** | <message:Structure structureID="ESTAT\_STS\_2\_2" namespace="urn:sdmx:org.sdmx.infomodel.datastructure.DataStructure=ESTAT:STS(2.2):ObsLevelDim:AllDimensions" dimensionAtObservation="AllDimensions"> ...  <Series ADJUSTMENT="W" FREQ="A" REF\_AREA="IT" STS\_INDICATOR="PROD" STS\_INSTITUTION="1" STS\_ACTIVITY="NS0040" STS\_BASE\_YEAR="2000" TIME\_FORMAT="P1Y"> |
| **Invalid XS attachment level. SDMX v2.0 Cross Sectional Specific** | <ns1:Section FREQ="A" |

Table 3 - Structural Validations

## Data Validation

|  |  |
| --- | --- |
| **Error description** | **Example** |
| **invalid code** | <Series ADJUSTMENT="W" FREQ="A" REF\_AREA="IT" STS\_INDICATOR="PROD" STS\_INSTITUTION="1" STS\_ACTIVITY="NS0040" STS\_BASE\_YEAR="2000" **TIME\_FORMAT="P1A"**> |
| **missing dimension at series** | <Series ~~ADJUSTMENT="W"~~ REF\_AREA="LU" …> |
| **missing mandatory attribute** | <Obs TIME\_PERIOD="2005-08" OBS\_VALUE="1.53" OBS\_CONF="F" /> |
| **invalid text format** | <Obs TIME\_PERIOD="2004-02" OBS\_VALUE="FOO\_BAR" OBS\_CONF="F" OBS\_STATUS="A" /> |
| **minValue** | Value should not be below minValue which is declared inside structure file |
| **maxValue** | Value should not exceed the specified value in this attribute |
| **dim at obs at series/wrong dim at obs** | <message:Structure structureID="ESTAT\_STS\_2\_2" namespace="urn:sdmx:org.sdmx.infomodel.datastructure.DataStructure=ESTAT:STS(2.2):ObsLevelDim:TIME\_PERIOD" **dimensionAtObservation="TIME\_PERIOD"**> … **<Series** ADJUSTMENT="W" **TIME\_PERIOD="2004-02"** REF\_AREA="IT" STS\_INDICATOR="PROD" STS\_INSTITUTION="1" STS\_ACTIVITY="NS0040" STS\_BASE\_YEAR="2000" TIME\_FORMAT="P1Y">  **<Obs FREQ="A"** |
| **wrong attribute attachment level** | <Group xsi:type="ns1:Sibling" REF\_AREA="LU" ADJUSTMENT="W" STS\_INDICATOR="PROD" STS\_ACTIVITY="NS0050" STS\_INSTITUTION="1" STS\_BASE\_YEAR="2000" ... **TIME\_FORMAT="P1Y"**/> |
| **dimension at dataset level** | <message:DataSet **ADJUSTMENT="W"** |
| **invalid time period value** | <Obs TIME\_PERIOD="2004-**Q5**" |
| **missing dimension at group** | <Group xsi:type="ns1:Sibling" ~~REF\_AREA="IT"~~ ADJUSTMENT="W" |
| **duplicate observations** | <Obs TIME\_PERIOD="2004-02" OBS\_VALUE="1.2004" OBS\_CONF="F" OBS\_STATUS="A" /> **<Obs TIME\_PERIOD="2004-02" OBS\_VALUE="14" OBS\_CONF="F" OBS\_STATUS="A" />** |

Table 4 - Data validations

# Annex H: The structure of a header file

The full description of the header file can be found in the [SDMX 2.0 section 03A](https://sdmx.org/wp-content/uploads/SDMX_2_0_SECTION_03A_SDMX_ML.pdf) online documentation.

header.id=JD014

header.test=true

header.truncated=false

header.name=Trans46302

header.prepared=2001-03-11T09:30:47-05:00

header.senderid=BIS

header.sendername=Bank for International Settlements

header.sendercontactname=G.B.Smith

header.sendercontactdepartment=Statistics

header.sendercontactrole=Statistician

header.sendercontacttelephone=210 2222222

header.sendercontactfax=210 00010999

header.sendercontactx400=

header.sendercontacturi=www.sdmx.org

header.sendercontactemail=gbsmith@sdmx.com

header.receiverid=ECB

header.receivername=European Central Bank

header.receivercontactname=John John

header.receivercontactdepartment=Statistics

header.receivercontactrole=Jack Over

header.receivercontacttelephone=210 1234567

header.receivercontactfax=210 3810999

header.receivercontactx400=x400

header.receivercontacturi=www.sdmx.org

header.receivercontactemail=lala@sdmx.com

header.datasetagency=BIS

header.datasetid=BIS\_JD\_237

header.datasetaction=Append

header.extracted=2001-03-11T09:30:47-05:00

header.reportingbegin=2000-01-01T00:00:00

header.reportingend=2006-01-01T00:00:00

header.reportingbegin=2000-12-01T00:00:00

header.source=StatApp

header.lang=en

header.keyfamilyref=JD

header.keyfamilyagency=ESTAT

# Annex I: The structure of a column mapping file

## Mapping for CSV

This is how a file that maps input / output for CSV looks like:

<?xml version="1.0" encoding="UTF-8"?>

<Mapping>

<Concept name="FREQ" value="1" level="1" fixed="false"/>

<Concept name="JD\_TYPE" value="2" level="1" fixed="false"/>

<Concept name="JD\_CATEGORY" value="3" level="1" fixed="false"/>

<Concept name="VIS\_CTY" value="4" level="1" fixed="false"/>

<Concept name="TIME" value="5" level="1" fixed="false"/>

<Concept name="OBS\_VALUE" value="6" level="1" fixed="false"/>

<Concept name="OBS\_CONF" value="7" level="1" fixed="false"/>

<Concept name="OBS\_STATUS" value="8" level="1" fixed="false"/>

<Concept name="OBS\_PRE\_BREAK" value="9" level="1" fixed="false"/>

</Mapping>

The name of the concept contains one of the dimensions, attributes, measures, the value contains either the column number or the fixed value, the fixed flag is true/false in accordance with the value attribute and the level is the an integer value representing the level on which the value can be found (in the case of input CSV values) or the level on which the value will be displayed.

## Mapping for FLR

This is how a file that maps input for Flr looks like:

<?xml version="1.0" encoding="UTF-8"?>

<Mapping>

<Concept name="FREQ" value="1-3"/>

<Concept name="JD\_TYPE" value="4-4"/>

<Concept name="JD\_CATEGORY" value="5-5"/>

<Concept name="VIS\_CTY" value="6-7"/>

<Concept name="TIME" value="8-14"/>

<Concept name="OBS\_VALUE" value="15-16+17-19"/>

<Concept name="OBS\_CONF" value="20-20"/>

<Concept name="OBS\_STATUS" value="21-21"/>

<Concept name="OBS\_PRE\_BREAK" value="22-22"/>

<Concept name="AVAILABILITY" value="" level="1" fixed="true" />

<Concept name="DECIMALS" value="1" level="" fixed="true" />

<Concept name="BIS\_UNIT" value="" level="" fixed="true" />

<Concept name="UNIT\_MULT" value="" level="" fixed="false" />

<Concept name="TIME\_FORMAT" value="" level="" fixed="false" />

<Concept name="COLLECTION" value="" level="" fixed="false" />

</Mapping>

The name of the concept contains one of the dimensions, attributes, measures, the value contains either the start/end character or the fixed value, the fixed flag is true/false in accordance with the value attribute and the level is the an integer value representing the level on which the value can be found or the level on which the value will be displayed.

In the mapping for an Flr output file “AUTO” is supported as value. To enable “AUTO” functionality for a component in the mapping file, the user should set fixed=”false” and value=”AUTO”. If auto is selected means that length will be extracted from the input file for this component.

## Mapping for Multilevel CSV

Rules for creating a mapping file for multiple level CSV When creating a mapping file, the user must follow all the rules from section 3.2 Rules for creating a multi-level CSV file plus the following:

· The mapping should contain all the dimensions, observations and attributes (attached to Dataset, Group, Series, Observation) of the DSD1.  And all of them are expected to have a value provided.

· No default mapping can be provided for multilevel CSV. The number of levels inside the mapping file must be the same as the number of levels declared in number of levels of CSV/FLR parameter.

· Ignoring a component is not currently permitted in multilevel CSV files and all components must be provided a value/column.

· Different from the flat input CSV case, for multiple level CSV the supported number of columns per concept is only one and the same format can be used both for output and input. The concatenator “+” is not supported for multiple columns associated with the same concept. See examples with supported values below:

If fixed=”false” then for CSV there can only be an integer value as a String, meaning it takes the value from the respective column:

 <Concept name="COLLECTION" value="2" level="3" fixed="false" />

And for FLR an integer or a range, described using “-“:

<Concept name="COLLECTION" value="2" level="3" fixed="false" />

COLLECTION takes the value from position 2 (a character only)

<Concept name="COLLECTION" value="2-4" level="3" fixed="false" />

COLLECTION value will be taken from position 2 to 4.

The following is not supported by multilevel CSV:  concatenate two or more fields use the “+” character (plus symbol) supported only for flat input CSV (number of levels = 1)

<?xml version="1.0" encoding="UTF-8"?>

<Mapping>

<Concept name="TAB\_NUM" value="1" level="1" fixed="false"/>

<Concept name="REV\_NUM" value="2" level="1" fixed="false"/>

<Concept name="FREQ" value="3" level="1" fixed="false"/>

<Concept name="COUNTRY" value="4" level="2" fixed="false"/>

<Concept name="SEX" value="5" level="2" fixed="false"/>

<Concept name="DEMO" value="6" level="2" fixed="false"/>

<Concept name="UNIT\_MULT" value="7" level="2" fixed="false"/>

<Concept name="DECI" value="8" level="2" fixed="false"/>

<Concept name="TIME\_FORMAT" value="9" level="2" fixed="false"/>

<Concept name="UNIT" value="10" level="2" fixed="false"/>

<Concept name="TIME" value="11" level="3" fixed="false"/>

<Concept name="OBS\_VALUE" value="12" level="3" fixed="false"/>

<Concept name="OBS\_STATUS" value="13" level="3" fixed="false"/>

</Mapping>

## Mapping for SDMX CSV

SDMX CSV format does not use a mapping. The order from the DSD  is expected to be followed but grouped by type as the following: *dimensions, time, observation value, observation level attributes, other attributes.*

## Fixed values example

For both CSV and FLR mapping files, the user can set a fixed value for a component. When trying to convert a message of CSV or FLR format to an SDMX-ML some columns may not be present because the data is implied. For example the column for the ‘REF\_COUNTRY’ concept may be omitted. In that case user can set a fixed value in the mapping dialog for the absent column.

Another sample of an XML mapping file, applicable for an FLR input file is shown below:

<?xml version="1.0" encoding="UTF-8"?>

<Mapping>

<Concept name="FREQ" value="ANNUAL" level="" fixed="true"/>

<Concept name="JD\_TYPE" value="1-1" level="" fixed="false"/>

<Concept name="JD\_CATEGORY" value="2-2" level="" fixed="false"/>

<Concept name="VIS\_CTY" value="3-4" level="" fixed="false"/>

<Concept name="TIME" value="5-11" level="" fixed="false"/>

<Concept name="OBS\_VALUE" value="12-13+14-16" level="" fixed="false"/>

<Concept name="OBS\_CONF" value="17-17" level="" fixed="false"/>

<Concept name="OBS\_STATUS" value="18-18" level="" fixed="false"/>

<Concept name="OBS\_PRE\_BREAK" value="19-19" level="" fixed="false"/>

</Mapping>

\* FREQ correlates with the value of column 1

\* OBS\_VALUE correlates with the value of column 13, concatenated with the value of column 14, concatenated with the values of columns 15,16,17

A user can also specify a fixed value in case he/she wants to replace an existing one and cannot use the transcoding procedure. For example a user may want to replace an existing value for an observation attribute which might not have a codelist thus cannot be transcoded

## Mapping for cross sectional files

When converting from a CSV file using a Cross Sectional DSD, the cross sectional measures that belong to a series might appear in one record rather than in consecutive rows. The following two file structures might appear when converting with a cross sectional DSD:

1. In the first structure each observation value appears along with the cross sectional measure in one row. Each cross sectional measure appears in the same row in each record (4th column).

A;GR;F;DEATHST;2005;23871;P

A;GR;F;EMIGT;2005;6331;P

A;GR;F;IMMIT;2005;10581;P

A;GR;F;LBIRTHST;2005;28345;P

For this kind of files, the measure dimension is always mapped to the specific column. In the above sample for the Country Greece (GR) and for female (F) the annual (A) observation values are reported consecutively in rows and correspond to Deaths (DEATHST), Emigrants (EMIGT), Immigrants (IMMIT) and Births (LBIRTHST) respectively.

1. In the second file structure the components that correspond to cross sectional measures do not appear in the CSV, but only their values.

A;GR;F; 23871; 6331;10581;28345;2005;P

In the above record of the CSV file only the observation values that correspond to the cross sectional measures appear. Specifically the value 23871 is the measure of Deaths, the value 6331 of Emigrants, the value of 10581 of Immigrants and the value 28345 of Births. In the CSV input parameters screen the user should select to map cross sectional measures and map each measure to a column in the mapping screen. For this example the measure DEATHST has been mapped to column 4, the measure EMIGT to column 5, the measure IMMIT to column 6 and the measure LBIRTHST to column 7. In that way the user does not have to map the observation value as this will not appear in a specific column in the CSV file but will appear in the column that the respective cross sectional measure has been mapped

## Mapping for files containing explicit Measures

When converting from a FLR file to CSV using a DSD that contains explicit measures, those measures might appear in one record rather than in consecutive rows. The following two file structures might appear when converting with a DSD that contains explicit measures:

In the first structure each observation value appears along with the measure in one row. Each explicit measure appears in the same row in each record (2nd column).

HHSEQNUM;MEASURE;OBS\_VALUE

03;NOWKREAS;3

03;PROXY;01

For this kind of files, the measure dimension is always mapped to the specific column.

In the second file structure the components that correspond to explicit measures do not appear in the CSV, but only their values. To achieve the following output the user must provide a mapping file for the explicit measures that will replace the MEASURE dimension and have to set the parameter UseExplicitMeasures=true or check the corresponding checkbox for the desktop application or the web application.

HHSEQNUM;NOWKREAS;PROXY;

03;3;01

In the above record of the CSV file only the observation values that correspond to the explicit measures appear. The user does not have to map the observation value as this will not appear in a specific column in the CSV file but will appear in the column that the respective explicit measure has been mapped.

# Annex J: The structure of an excel parameter sheet mapping file or sheet

The following example shows a mapping between value-sheets (Data1, Data2, and Data3) and parameters sheets (Parameters\_Data1, Parameters\_Data2) in an external txt file:

Data1=Parameters\_Data1

Data2=Parameters\_Data2

Data3=Parameters\_Data2

The following example shows a mapping between value-sheets and the parameter sheets inside the excel data file as a sheet with the name Parameter\_mapping:

|  |  |
| --- | --- |
| Data Sheet | Parameter Sheet |
| Data1 | Parameters\_Data1 |
| Data2 | Parameters\_Data2 |

The headers “Data Sheet” and “Parameter Sheet” must exist at the first row of the sheet.

The external txt mapping file can only be used by webservice as txt. In all modules you can include the mapping file inside the excel parameters file.

# Annex K: The structure of a transcoding file

The following file is an example of a transcoding file:

<?xml version="1.0" encoding="UTF-8"?>

<mes:Structure xmlns:mes="http://www.SDMX.org/resources/SDMXML/schemas/v2\_0/message" xmlns:str="http://www.SDMX.org/resources/SDMXML/schemas/v2\_0/structure">

<mes:Header>

<mes:ID>IDREF1</mes:ID>

<mes:Test>false</mes:Test>

<mes:Prepared>2016-08-10T14:38:11.402+03:00</mes:Prepared>

<mes:Sender id="Unknown"/>

<mes:Receiver id="Unknown"/>

</mes:Header>

<mes:StructureSets>

<str:StructureSet agencyID="DummyID"

id="Id\_for\_StructureSet" urn="urn:sdmx:org.sdmx.infomodel.mapping.StructureSet=DummyID:Id\_for\_StructureSet(1.0)" version="1.0">

<str:Name xml:lang="en">Name of StructureSet</str:Name>

<str:CodelistMap id="FREQ">

<str:Name xml:lang="en">Name of CodelistMap</str:Name>

<str:CodelistRef> <str:URN>urn:sdmx:org.sdmx.infomodel.codelist.Codelist=ESTAT:CL\_FREQ(1.0)</str:URN>

<str:AgencyID>ESTAT</str:AgencyID>

<str:CodelistID>CL\_FREQ</str:CodelistID>

<str:Version>1.0</str:Version>

</str:CodelistRef>

<str:TargetCodelistRef>

<str:URN>urn:sdmx:org.sdmx.infomodel.codelist.Codelist=ESTAT:CL\_FREQ(1.0)</str:URN>

<str:AgencyID>ESTAT</str:AgencyID>

<str:CodelistID>CL\_FREQ</str:CodelistID>

<str:Version>1.0</str:Version>

</str:TargetCodelistRef>

<str:CodeMap>

<str:MapCodeRef>annuel</str:MapCodeRef>

<str:MapTargetCodeRef>A</str:MapTargetCodeRef>

</str:CodeMap>

<str:CodeMap>

<str:MapCodeRef>semestriel</str:MapCodeRef>

<str:MapTargetCodeRef>S</str:MapTargetCodeRef>

</str:CodeMap>

<str:CodeMap>

<str:MapCodeRef>trimestiel</str:MapCodeRef>

<str:MapTargetCodeRef>Q</str:MapTargetCodeRef>

</str:CodeMap>

</str:CodelistMap>

</str:StructureSet>

</mes:StructureSets>

</mes:Structure>

# Annex L: Conversion limitations

## CSV/FLR limitations

In order for a flat CSV or FLR file to be converted successfully some limitations exist. Each line in such a file represents one record. This implies that no new line characters should be present inside the data in that record. In cases where the text contains new line characters, the user should supress all new line characters.

For flat CSV files also the attributes at a higher level than Observation are processed.

The user has to repeat the respective attribute on every related observation if not the first encountered value is used and the rest are ignored. In case the values of the attributes other than the Observation level are not the same for the same group or time series then a warning is printed in the log. The conversion will be successful using the first encountered value.

In the mapping file all the dimensions, attributes, primary measure, time reference etc must be provided according to the DSD.

For multilevel files and those containing groups is mandatory to have the data ordered - rows for the same series or group are consecutive (Input Ordered). Input has to be ordered also when converting to cross sectional format.

Also for **multilevel** the following rules apply:

* The lower the level number, the higher is the priority of that level in the CSV file.
* The first column is considered a level identifier, meaning it defines the level of the record. The implementation expects it at the beginning of each row both for CSV and FLR
* As soon as level n is reached in the file, the previous information for levels higher than n are no longer useful.
* For each level the expected order is Group, Dataset, Series, Time and Observation. There can be more on the same level (i.e. Dataset and Series attributes on the same level) but having Observation on level 1 and Dataset attributes on level 3 might lead to an erroneous result. Therefore, observation and observation attributes are expected on the last levels.
* Each level must have at least one value. Providing 4 levels but having an empty row for level 3 is incorrect.
* Multilevel CSV does not support a header row (both for input and for output). Meaning for multilevel case “Header row” is false.
* For mapping both CSV, FLR as input or output, only one column position is accepted. E.g “2+3” as for flat files is not supported.
* In the input file the number of columns (delimiters) per level is equal. Meaning that a for the same level, the same number must be provided. E.g. all rows starting with 2 (for level 2 records) will have the same number of columns (hence delimiters)
* The column semantic per level is equivalent. Related to the mapping and to affirmation above it is expected that the csv file (equally input or output) at the same level, the same column holds the same semantic. E.g. below, the second column for third level will represent the TIME dimension value:

1;RQFI05V1;1;A

2;FI;F;ADJT;0;0;P1Y;PERS

3;**2005**;35;P

3;**2006**;55;P

3;**2007**;38;P

* The order of the levels for each group should be correct, in the sense that each level is considered a child of the previous higher level encountered in the file. The example below is wrong. 3;2000-01;1.113;;A;; is assumed to belong to group 1;Cbecause level 2 is placed in a wrong location:

1;C

3;2000-01;1.113;;A;;

2;ANN;Amounts;M

3;2000-04;5.2;;A;;

* The group values should be unique: The same group values should not appear later in the file as it will be considered a different group (key). The example below Converter will consider a new group when encountering twice the same information:

1;P

2;MON;Amounts;M

1;P

2;MON;Amounts;M

3;2000-01;3.14;;A;;

3;2000-01;1.113;;A;;

2;ANN;Amounts;M

3;2000-04;5.2;;A;;

3;2000-09;5.2;;A;;

1;A

2;MON;Amounts;M

3;2000-04;5.55;;A;;

1;P

2;MON;Amounts;M

3;2000-02;4.54;;A;;

Last two affirmations are related to the input order. For multilevel files and those containing groups is mandatory to have the data ordered - rows for the same series or group are consecutive (Input Ordered). Input must be ordered also when converting to cross sectional format.

In other words, the order of records is expected to be consistent, in the sense that each level is considered a child of the previously higher lever encountered in the file, also the same group should not appear later in the file as it will be considered a different group (key).

## Limitations for Message Group messages

The message group is not an input format (as Generic or Compact are) but just a hint to Converter that the input file should be treated differently. The Message Group allows more than one data message to be included in a single transmission. Currently, Converter supports Message Group as input sdmx-ml file for all four different representations of reporting datasets: Generic data message, Compact data Message, Utility data Message and Cross Sectional Data Message. The only limitation the user has when converting a message group file is that the input file must have only ONE dataset reported in the message

## Limitations when converting to/from Cross Sectional

The selected DSD for these files must support cross-sectional messages i.e. having defined cross-sectional measures and cross-sectional attachment levels for its components (see SDMX standard [www.sdmx.org](http://www.sdmx.org))

A special case is the conversion from Cross Sectional to SDMX 2.1 messages where the provided data structure has to be valid with Time dimension or cross sectional measures.

## Limitations when converting from SDMX 2.1 to other formats

With exception of SDMX V2.1 formats no other format supports multiple datasets in a single data message. Consequently Converter only supports conversions of SDMX V2.1 files having a single dataset per message. Also conversion from SDMX 2.1 to SDMX 2.0 formats is possible only when the DSD is compatible with both formats. The compatibility with V2.0 formats is checked in SDMX Source API before conversion begins.

## Limitations when converting from Gesmes TS

In GESMES/TS (SDMX-EDI) there are 2 kinds of observation attributes:

* the observation flags which are part of the ARR segments
* the normal attributes that are included in the attribute section.

The observation flags are fixed. They are the observation status, observation confidentiality and observation pre-break. In SDMX-ML the concept of "observation flags" does not exist. In order to map between SDMX-ML DSD observation attributes and SDMX-EDI observation flags the OBS\_CONF or CONF\_STATUS are considered to be the concept for “observation confidentiality”. Also OBS\_PRE\_BREAK or PRE\_BREAK\_VALUE are considered to be the concept for "observation pre-break". **These attribute names are hard-coded in SDMX Converter** and, at conversion time, checked which one exists in DSD to be used in the output. If any of these names was not found in DSD the OBS\_CONF/OBS\_PRE\_BREAK pair is used.

**Having said that, the following limitations may occur when converting from Gesmes TS:**

* If in the future other attributes names will be used in DSD for observation flags in GESMES/TS the Converter needs to be updated as the above names are hard coded.
* Parsing of ARR segment in the input Gesmes file is dependent by the value of the frequency code. As the frequency code lists could evolve over time Converter needs to be updated in order to support new codes.
* Currently Converter supports A, H, Q, M, W, B or D as values for frequency in the Gesmes file. B and D are interpreted in the same way.

**Sibling group wildcard**

In GESMES/TS (SDMX-EDI) there is the possibility to use wildcard for dimensions in a sibling group (e.g. **ARR+4+:XX:ZZ:CC'** - this is a sibling group with FREQ wildcarded). Cross sectional 2.0 (CROSS\_SDMX) does not support groups. So, to be able to make the conversion to cross sectional as well, it will be translated to the corresponding seriesfor each FREQ code from the codelist instead being converted to a sibling group.

This means that for **ARR+4+:XX:ZZ:CC'** input**,** instead of converting to a sibling group as below:

<bis:SiblingGroup JD\_TYPE="XX" JD\_CATEGORY="XX" VIS\_CTY="CC"/>

The conversion will be made to series like this (below is a COMPACT\_SDMX example):

<bis:Series FREQ="A" JD\_TYPE="XX" JD\_CATEGORY="XX" VIS\_CTY="CC"/>

<bis:Series FREQ="D" JD\_TYPE="XX" JD\_CATEGORY="XX" VIS\_CTY="CC"/> <bis:Series FREQ="H" JD\_TYPE="XX" JD\_CATEGORY="XX" VIS\_CTY="CC"/>

<bis:Series FREQ="M" JD\_TYPE="XX" JD\_CATEGORY="XX" VIS\_CTY="CC"/> <bis:Series FREQ="Q" JD\_TYPE="XX" JD\_CATEGORY="XX" VIS\_CTY="CC"/>

<bis:Series FREQ="W" JD\_TYPE="XX" JD\_CATEGORY="XX" VIS\_CTY="CC"/>

## Detection of input format

Converter desktop application is only able to detect the SDMX 2.0, SDMX 2.1and Excel formats. In case the format cannot be detected the user will get a notification message.

# Annex M: How to configure the excel readers (Excel parameter sheet contents)

Parameter File is used as layout description of the Excel input file. If the external parameter file is provided and mapping parameter is not provided then the external parameter file will be used for all data sheets even if the Excel file contains other parameter sheets.

The external parameter file must follow the rules below:

* All the contained data must be separated by space. Only the value after FIX position type can have a value containing space character.
* The parameter file must contain on the first row the column headers.
* The Element is the name of the dimension or attribute exactly as it is named in the DSD file.
* The Type can only be DIM (dimension) or ATT (attribute).
* **PosType** is CELL, ROW, COLUMN, FIX, SKIP, OBS\_LEVEL or MIXED.
* **DataStart** is the position where the actual data starts.
* **DataEnd** is the position where the actual data ends. If DataEnd is specified, the converter will read until that limit and ignore the maximum of empty rows/column. To have multiple pairs of DataStart/End for one DataSheet then use parameter mapping sheet inside Excel file and map different parameter sheets for the same DataSheet.
* Either **NumColumns**, the number of columns containing data, or **MaxEmptyRows**, the maximum row allowed to be empty before finishing reading data, must be provided. Valid values are integers greater than zero.
* DataStart, DataEnd and NumColumns (or MaxEmptyRow) can be defined to the left of the Elements or at the end of the file.
* **MaxEmptyColumns**, the maximum number of empty cells allowed in the current row before start reading the next row. If not present the default is 1000.
* **TrancodingSheet**, specify the name of the sheet containing a mapping between the couple text/dimension and the SDMX value. For the structure of the transcoding sheet see [appendix V](#_Annex_V:_The)
* **The Transcoding Sheet name must start with “Trans” to avoid parsing it as data Sheet for example:**

|  |  |
| --- | --- |
| TranscodingSheet | Trans\_1 |

A transcoding sheet can be present inside external parameter sheet or inside input file, but always must be declared inside parameter sheet. If transcoding sheet exists inside both files the below priorities are followed.

**1st Priority:** Parameters external + transcoding external

**2nd Priority:** Parameters external + transcoding internal

**3rd Priority:** Parameters Internal + transcoding internal

* **SkipRows**, this parameter should be used to skip one or several rows. This parameter contains the number of the row(s) to be skipped. For multiple rows to be skipped you can put number of rows in a list with commas inside the same cell:

|  |  |
| --- | --- |
| SkipRows | 24,25,26 |

or you can put the parameter multiple times inside the parameter sheet:

|  |  |
| --- | --- |
| SkipRows | 24 |
| SkipRows | 25 |

* **SkipObservationWithValue**, skip observation with certain value, converter will skip and ignore this observation.
* **SkipIncompleteKeys**, skip observations with empty or wrong dimension or concept values when value is set to true. Default value is set to true.
* **ConceptSeparator**, character that separates multiple concepts defined in the same cell. In the parameter sheet you can define the position inside the cell that indicates the value for this concept with the slash character:

|  |  |  |  |
| --- | --- | --- | --- |
| REF\_AREA | DIM | CELL | B1/2 |

And inside data Sheet the converter will search the cell B1 and the first position separated with the character defined by the ConceptSeparator. For example if the concept separator is “;” and in Data Sheet the B1 cell has the value DE;PROD then the REF\_AREA for the above example will be assigned the value PROD. ConceptSeparator can be defined for Position Types Cell, Rows and Columns, OBS\_LEVEL and MIXED.

* **MissingObservationCharacter**, if set at an observation cell converter treats the observation as missing. If DefaultValue is set this value will be assigned.
* **roundToFit,** If this parameter is present must have value true or false. After an observation value is rounded according to parameter *Rounding Precision*, the value is checked again and if the number of digits that remain are more from the ***maxLength*** attribute of the DSD, then the value is rounded again (with Round Half algorithm) to the number of digits declared in the DSD.
* **formatValues,** Excel stores numbers differently that you may have them formatted on the worksheet. When the actual value of cells is needed for the output we need to set this parameter on the parameter sheet with the value *actualValue.* The **default value** is *customFormat* which means for the custom formatted cells the output will be the value displayed by cell including the formatting but for numbers the actual value is computed except the cells that have a custom format defined by the user. If only the displayed values needed then set the parameter with the value *asDisplayed*. If this parameter is not set or another invalid value is selected then the *customFormat* behavior is used.

*In general, for numeric elements (as defined in the DSD), the leading or trailing whitespaces from the values are trimmed.*

As *custom format* is recognized every format that is defined by the user. The following table indicates the formats that are not considered as custom formats, everything else is considered custom:

|  |  |
| --- | --- |
| 0, | "General" |
| 1, | "0" |
| 2, | "0.00" |
| 3, | "#,##0" |
| 4, | "#,##0.00" |
| 5, | "$#,##0\_);($#,##0)" |
| 6, | "$#,##0\_);[Red]($#,##0)" |
| 7, | "$#,##0.00);($#,##0.00)" |
| 8, | "$#,##0.00\_);[Red]($#,##0.00)" |
| 9, | "0%" |
| 0xa, | "0.00%" |
| 0xb, | "0.00E+00" |
| 0xc, | "# ?/?" |
| 0xd, | "# /" |
| 0xe, | "m/d/yy" |
| 0xf, | "d-mmm-yy" |
| 0x10, | "d-mmm" |
| 0x11, | "mmm-yy" |
| 0x12, | "h:mm AM/PM" |
| 0x13, | "h:mm:ss AM/PM" |
| 0x14, | "h:mm" |
| 0x15, | "h:mm:ss" |
| 0x16, | "m/d/yy h:mm" |
|  | // 0x17 - 0x24 reserved for international and undocumented 0x25, "#,##0\_);(#,##0)" |
| 0x26, | "#,##0\_);[Red](#,##0)" |
| 0x27, | "#,##0.00\_);(#,##0.00)" |
| 0x28, | "#,##0.00\_);[Red](#,##0.00)" |
| 0x29, | "(**#,##0);(**(#,##0);(\* \"-\");(@\_)" |
| 0x2a, | "($**#,##0);($**(#,##0);($\* \"-\");(@\_)" |
| 0x2b, | "(**#,##0.00);(**(#,##0.00);(\*\"-\"??);(@\_)" |
| 0x2c, | "($**#,##0.00);($**(#,##0.00);($\*\"-\"??);(@\_)" |
| 0x2d, | "mm:ss" |
| 0x2e, | "[h]:mm:ss" |
| 0x2f, | "mm:ss.0" |
| 0x30, | "##0.0E+0" |
| 0x31, | "@" - This is text format. |
| 0x31 | "text" - Alias for "@" |

* When **MIXED** is used as PosType not all the combinations are allowed. For example
* VALUATION DIM MIXED COLUMNS 5 ROW 20 is a valid description if COLUMN 5 is found empty ROW 20 will be used.
* The same for VALUATION DIM MIXED CELL B5 FIX 23 if CELL B5 is empty the FIX value ‘23’ will be used.
* Unaccepted combinations are between FIX, SKIP, OBS\_VALUE. e.g. MIXED OBS\_VALUE 1 FIX 25 will fail.
* When OBS\_LEVEL appears the column corresponding to the Observation values is expected to exist even if OBS\_LEVEL is part of MIXED position type.
* e.g. For the declaration below, if H14 cell is found non empty the converter application will skip one column to the right which represents the observation level attributes that can be empty or non empty.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **OBS\_STATS** | **ATT** | **MIXED** | **CELL** | **H14** | **OBS\_LEVL** | **1** |

* For MIXED the values are checked for each Observation. So for the example below for each Observation the observation level is checked and it the cell at that corresponding position is empty the value from the cell H14 will be used.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **OBS\_STATS** | **ATT** | **MIXED** | **OBS\_LEVEL** | **1** | **CELL** | **H14** |

* If an Observation value is left empty intentionally, the data sender might to transmit further attributes (e.g. OBS\_STATUS), giving reasoning on the fact that the observation is blank. For this a new parameter is added called “DefaultValue”. This parameter is optional. If present and all the mandatory dimensions can be resolved when parsing the empty observation cell, the Converter will create an observation with the observation value equals with the value specified in the DefaultValue parameter (this could be either a 0, a " " or a "NaN" for example, the converter will only put the value specified in the output file). This parameter can be defined to the left of the Elements in the second group of parameters (DataStart, MaxEmptyRows).

**DefaultValue “NaN”**

* For all cells storing NUMBERs, Excel uses internally a rounding mechanism in order to avoid displaying floating point approximations induced by the IEEE 754 standard (see [appendix E](#_APPENDIX_E_–) and [appendix F](#_APPENDIX_F_–) for more details). In order to control the rounding performed by Converter for Excel decimal numbers, a new parameter (RoundingPrecision) has been introduced and will be applied for converting the values from all cells storing NUMBERs as well as from cells having a formula which evaluates to a NUMBER.

The RoundingPrecision parameter is optional, if not present Converter uses 6 as default precision number. The parameter value should be an integer between 0 and 15. If it is 0 the numbers will be rounded without decimals. If the parameter value is greater than 15 then the value will be ignored and 15 will be used. If the parameter value is a negative integer then the default value (6) is used.

**RoundingPrecision 4**

The same rules apply to Excel Parameter Sheets with the difference that each piece of information is stored in its own cell instead of being separated by spa

# Annex N: The structure of an excel parameter file

The following example is a potential parameter sheet inside the excel:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Element | Type | PosType | Position |  | DataStart | D24 |
| REF\_AREA | DIM | CELL | B1/1 |  | MaxNumOfEmptyColums | 1 |
| STS\_INDICATOR | DIM | CELL | B1/2 |  | ConceptSeparator | ; |
| STS\_INSTITUTION | DIM | FIX | 1 |  | NumColumns | 6 |
| STS\_BASE\_YEAR | DIM | CELL | B7 |  | TranscodingSheet | Trans\_1 |
| FREQ | DIM | CELL | I14 |  | SkipRows | 31 |
| STS\_ACTIVITY | DIM | COLUMN | B/1 |  | SkipObservationWithValue | 0 |
| TIME\_PERIOD | DIM | ROW | 23 |  | DataEnd | I31 |
| OBS\_STATUS | ATT | CELL | D2/1 |  |  |  |
| OBS\_CONF | ATT | CELL | D2/2 |  |  |  |
| ADJUSTMENT | DIM | MIXED | COLUMN | A | ROW | 21 |
| OBS\_ATT | ATT | FIX | P |  |  |  |
| TITLE | ATT | SKIP |  |  |  |  |
| UNIT | ATT | SKIP |  |  |  |  |
| UNIT\_MULT | ATT | SKIP |  |  |  |  |
| DECIMALS | ATT | SKIP |  |  |  |  |
| COLLECTION | ATT | SKIP |  |  |  |  |
| AVAILABILITY | ATT | SKIP |  |  |  |  |
| UNIT\_INDEX\_BASE | ATT | SKIP |  |  |  |  |
| TITLE\_COMPL | ATT | SKIP |  |  |  |  |
| NAT\_TITLE | ATT | SKIP |  |  |  |  |
| DOM\_SER\_IDS | ATT | SKIP |  |  |  |  |
| COMPILATION | ATT | SKIP |  |  |  |  |
| BREAKS | ATT | SKIP |  |  |  |  |
| COVERAGE | ATT | SKIP |  |  |  |  |
| SOURCE\_PUB | ATT | SKIP |  |  |  |  |
| SOURCE\_AGENCY | ATT | SKIP |  |  |  |  |
| OBS\_COM | ATT | SKIP |  |  |  |  |
| TIME\_FORMAT | ATT | FIX | P1Y |  |  |  |
| OBS\_PRE\_BREAK | ATT | SKIP |  |  |  |  |

# ANNEX O: Supported Conversion

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input**  **Output** | **CSV** | **SDMX**  **CSV** | **GESMESTS** | **SDMX GENERIC** | **SDMX COMPACT** | **SMDX UTILITY** | **SDMX CROSS SECTIONAL** | **EXCEL** | **SDMX 2.1 (\*\*\*\*\*)** | **Message Group (\*\*\*\*\*\*)** | **FLR** |
| CSV (\*\*) | NO | YES | YES | YES | YES | YES | YES(\*) | YES | YES | YES | YES |
| SDMX CSV | YES | NO | YES | YES | YES | YES | YES | YES | YES | YES | YES |
| GESMES/TS | YES | YES | NO | YES | YES | YES | YES(\*) | YES | YES | NO | NO |
| SDMX GENERIC | YES | YES | YES | NO | YES | YES | YES(\*) | YES | YES | NO | YES |
| SDMX COMPACT | YES | YES | YES | YES | NO | YES | YES(\*) | YES | YES | NO | YES |
| SMDX UTILITY | YES | YES | YES | YES | YES | NO | YES(\*) | YES | YES | NO | YES |
| SDMX CROSS-SECTIONAL | YES  (\*) | YES | YES  (\*) | YES  (\*) | YES  (\*) | YES  (\*) | NO | NO | YES | NO | YES |
| SDMX 2.1 | YES | YES | YES | YES | YES | YES | YES  (\*\*\*\*) | YES | NO | NO | YES |
| EXCEL | YES | YES | YES | YES | YES | YES | NO | NO | YES | NO | YES |
| FLR | YES | YES | YES | NO | YES | YES | NO | YES | YES | YES | NO |

(\*) The selected DSD for these files must support cross-sectional messages i.e. having defined cross-sectional measures and cross-sectional attachment levels for its components (see SDMX standard [www.sdmx.org](http://www.sdmx.org/))

(\*\*)When the output format (CSV) is a flat file the attributes, attached in higher level than observations in the input message are omitted.

(\*\*\*\*)The provided DSD has to be valid with Time dimension or cross sectional measures.

(\*\*\*\*\*)With exception of SDMX V2.1 formats no other format supports multiple datasets in a single data message. Consequently Converter only supports conversions of SDMX V2.1 files having a single dataset per message. Also conversion from SDMX 2.1 to SDMX 2.0 formats is possible only when the DSD is compatible with both formats. The compatibility with V2.0 formats is checked in SDMX Source API before conversion begins.

(\*\*\*\*\*\*)The Message Group input file should have only one dataset reported in the message.

# Annex P – Structure of a SDMX CSV file

For SDMX\_CSV file header row is mandatory and the order of concepts and attributes is expected to be the one from the Data Structure but grouped by type as the following: *dimensions, time, observation value, observation level attributes, other attributes.*. The first column is the Dataflow Id (Agency:Id:Version). See an example below:

DATAFLOW;TABLE\_IDENTIFIER;FREQ;REF\_AREA;REF\_SECTOR;EDU\_TYPE;ISC11P\_LEVEL;ISC11P\_CAT;ISC11P\_SUB;GRADE;FIELD;INTENSITY;COUNTRY\_ORIGIN;COUNTRY\_CITIZENSHIP;SEX;AGE;STAT\_UNIT;UNIT\_MEASURE;TIME\_PERIOD;OBS\_VALUE;OBS\_STATUS;COMMENT\_OBS;TIME\_PER\_COLLECT;ORIGIN\_CRITERION;REF\_YEAR\_AGES;REF\_PER\_START;REF\_PER\_END;COMPILING\_ORG;DECIMALS;UNIT\_MULT

ESTAT:SSTSCONS\_PROD\_DT\_A(1.0);PERS1;A;MT;INST\_PRIV\_GOV;\_T;L5T8;C4;\_T;\_T;\_T;PT;W0;W0;\_T;\_T;STU;PER;2013;NaN;;;2014-07-01;\_Z;2013-01-01;2012-09-22;2013-06-15;\_T;0;0

ESTAT:SSTSCONS\_PROD\_DT\_A(1.0);PERS2;A;MT;INST\_PRIV;\_T;L2;\_T;\_T;\_T;\_T;\_T;W0;W0;\_T;\_T;TEACH;PER;2013;NaN;;;2014-07-01;\_Z;2013-01-01;2012-09-22;2013-06-15;\_T;0;0

ESTAT:SSTSCONS\_PROD\_DT\_A(1.0);PERS4;A;MT;INST\_T;\_T;L1;\_T;\_T;\_T;\_T;FT;W0;W0;M;\_T;AIDE;PER;2013;NaN;;;2014-07-01;\_Z;2013-01-01;2012-09-22;2013-06-15;\_T;0;0

ESTAT:SSTSCONS\_PROD\_DT\_A(1.0);PERS1;A;MT;INST\_PRIV\_IND;\_T;L5T8;C4;\_T;\_T;\_T;FT;W0;W0;\_T;\_T;STU;PER;2013;NaN;;;2014-07-01;\_Z;2013-01-01;2012-09-22;2013-06-15;\_T;0;0

ESTAT:SSTSCONS\_PROD\_DT\_A(1.0);PERS1;A;MT;INST\_PRIV;\_T;L4;C5;\_T;\_T;\_T;\_T;W0;W0;\_T;\_T;STU;PER;2013;NaN;;;2014-07-01;\_Z;2013-01-01;2012-09-22;2013-06-15;\_T;0;0

ESTAT:SSTSCONS\_PROD\_DT\_A(1.0);PERS2;A;MT;INST\_PRIV;\_T;L4;C4;\_T;\_T;\_T;PT;W0;W0;M;\_T;TEACH;PER;2013;NaN;;;2014-07-01;\_Z;2013-01-01;2012-09-22;2013-06-15;\_T;0;0

ESTAT:SSTSCONS\_PROD\_DT\_A(1.0);PERS2;A;MT;INST\_PUB;\_T;L5T8;\_T;\_T;\_T;\_T;FT;W0;W0;F;\_T;TEACH;PER;2013;NaN;;;2014-07-01;\_Z;2013-01-01;2012-09-22;2013-06-15;\_T;0;0

ESTAT:SSTSCONS\_PROD\_DT\_A(1.0);PERS4;A;MT;INST\_T;\_T;L0;C2;\_T;\_T;\_T;FT;W0;W0;\_T;\_T;MAN;PER;2013;NaN;;;2014-07-01;\_Z;2013-01-01;2012-09-22;2013-06-15;\_T;0;0

ESTAT:SSTSCONS\_PROD\_DT\_A(1.0);PERS3;A;MT;INST\_T;\_T;L0;\_T;\_T;\_T;\_T;\_T;W0;W0;\_T;\_U;TEACH;PER;2013;NaN;;;2014-07-01;\_Z;2013-01-01;2012-09-22;2013-06-15;\_T;0;0

ESTAT:SSTSCONS\_PROD\_DT\_A(1.0);PERS3;A;MT;INST\_T;\_T;L3;\_T;\_T;\_T;\_T;\_T;W0;W0;F;Y\_GE65;TEACH;PER;2013;NaN;;;2014-07-01;\_Z;2013-01-01;2012-09-22;2013-06-15;\_T;0;0

ESTAT:SSTSCONS\_PROD\_DT\_A(1.0);PERS1;A;MT;INST\_PUB;\_T;L3;C4;\_T;\_T;\_T;\_T;W0;W0;\_T;\_T;STU;PER;2013;NaN;;;2014-07-01;\_Z;2013-01-01;2012-09-22;2013-06-15;\_T;0;0

# Annex R: Examples of Validation Requests for Web Service

**sample soap request with structure in registry**

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v3="http://xmlns.ec.eu/BusinessActivityService/StruvalBAS/V3" xmlns:v31="http://xmlns.ec.eu/BusinessObjects/StruvalDS/V3/">

<soapenv:Header/>

<soapenv:Body>

<v3:ValidateData>

<Input type=”SDMX\_ML”>

<V31:InputFile>cid:input</V31:InputFile>

</Input>

<StructureConfig>

<v31:Registry url="http://registry.org/rest" agency="ESTAT" id="UOE\_NON\_FINANCE" version="1.0"/>

</StructureConfig>

<MaximumErrorNumber>1</MaximumErrorNumber>

</v3:ValidateData>

</soapenv:Body>

</soapenv:Envelope>

**sample soap request with structure in file and minumum configuration**

<soapenv:Body>

<v3:ValidateData>

<Input>

<V31:InputFile>cid:input</V31:InputFile>

</Input>

<StructureConfig>

<V31:Structure>

<V31:StructureFile><cid:structure></V31:StructureFile>

</V31:Structure>

</StructureConfig>

<MaximumErrorNumber>1</MaximumErrorNumber>

</v3:ValidateData>

</soapenv:Body>

**sample soap request with structure in file and all non-mandatory attributes used**

<soapenv:Body>

<v3:ValidateData>

<Input>

<V31:InputFile>cid:input</V31:InputFile>

</Input>

<StructureConfig>

<V31:Structure agency="ESTAT" id="UOE\_NON\_FINANCE" version="1.0">

<V31:StructureFile><cid:structure></V31:StructureFile>

</V31:Structure>

</StructureConfig>

<MaximumErrorNumber>10</MaximumErrorNumber>

</v3:ValidateData>

</soapenv:Body>

**sample Soap request for SDMX CSV validation**

<soapenv:Body>

<v3:ValidateData>

<Input type=”SDMX\_CSV”>

<v31:InputFile>cid:input</v31:InputFile>

<v31:InputConfig>

<v31:Config type="parameters">

<v31:configFile> cid:properties <v31:configFile/>

</v31:Config>

</v31:InputConfig>

</Input>

<StructureConfig>

<v31:Structure agency="ESTAT" id="UOE\_NON\_FINANCE" version="1.0">

<v31:StructureFile>cid:structure </v31:StructureFile>

<Structure>

</StructureConfig>

<MaximumErrorNumber>1</MaximumErrorNumber>

</v3:ValidateData>

</soapenv:Body>

**sample Soap request for MULTI\_LEVEL\_CSV validation with transcoding**

<soapenv:Body>

<v3:ValidateData>

<Input>

< v31:InputFile type=” MULTI\_LEVEL\_CSV”>cid:input</ v31:InputFile>

< v31:InputConfig>

< v31:Config type="parameters">

< v31:configFile>cid:properties </v31:configFile/>

</ v31:Config>

< v31:Config type=”mapping”>

< v31:configFile>cid:mapping</v31:configFile>

</ v31:Config>

< v31:Config type=”transcoding”>

< v31:configFile>cid:transcodin</v31:configFile>

</ v31:Config>

</ v31:InputConfig>

</Input>

<StructureConfig>

< v31:Structure agency="ESTAT" id="UOE\_NON\_FINANCE" version="1.0">

< v31:StructureFile>cid:structure</ v31:StructureFile>

</ v31:Structure>

</StructureConfig>

<MaximumErrorNumber>1</MaximumErrorNumber>

</v3:ValidateData>

</soapenv:Body>

# Annex S – Struval Error Messages

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Error Code** | **Message Code** | **MessageText** | **Error Details** | **Comments** |
| 500 |  | Internal server error. Validation service not available. | The STRUVAL service is not able to process the inputs due to an internal server error. |  |
| 140 |  | <Message from XML Parser> | The SDMX-ML file is not a well-formed XML file. It may contain such errors as invalid characters, tags that are not closed or are closed out of order. Well formedness of an XML file can be checked using different tools, such as the advanced text editors or online. |  |
| 150 | 003 | The dataset contains a series with a missing concept {0} | The data file contains series having dimensions or attributes which are not defined in DSD. |  |
| 150 | 004 | The DSD {0} used does not define a time dimension, required for the time series data. | When building a time-series dataset, one must use a DSD that includes a time dimension. |  |
| 150 | 005 | The dataset includes primary measure {0}, not expected by the DSD. | When building a time-series dataset, one must use a DSD that has a primary measure. |  |
| 150 | 904-1 | Series key {0} is not defined in DSD (unexpected size). | Dataset contains series keys with unexpected size. | Replaced by other validations (like missing concept 031 or some other code depending on the specific case because the size of the series keys may be higher or lower than expected size) |
| 150 | 904-2 | Series key {0} is not defined in DSD (incorrect codes). | Dataset contains series keys which unexpected size. | Replaced by 029 and the error is found when the code of the concept is incorrect. |
| 150 | 007 | The dataset contains a concept {0} that is not defined in DSD. | All concepts used in a dataset must be defined in a DSD. |  |
| 150 | 008 | Attribute {0} defined as mandatory in DSD is missing from the dataset. | The dataset contains a mandatory series level attribute which is not present in the data file. |  |
| 150 | 009 | Series attribute {0} is not defined in DSD. | The encountered attribute at the series level in data file does not exist in Data Structure. |  |
| 150 | 010 | Attribute {0} defined as mandatory in DSD is missing from the group. | The dataset contains a mandatory group level attribute which is not present in the data file. |  |
| 150 | 011 | Attribute {0} is assigned to the incorrect group. | The encountered attribute at the dataset level in data file does not exist in Data Structure. |  |
| 150 | 012 | Attribute {0} defined as mandatory in DSD is missing from the observation. | The dataset contains a mandatory observation level attribute which is not present in the data file. |  |
| 150 | 013 | Attribute {0} is not defined in DSD for observation. | The encountered observation attribute is not defined by Data Structure. |  |
| 150 | 014 | Dataset group {0} is not defined in the DSD. | Dataset contains group keys which unexpected size. |  |
| 150 | 015 | Dataset group {0} is not defined in the DSD. | Data Structure Definition does not define a Group with id '%s' |  |
| 150 | 016 | The mandatory concept {0} in DSD is currently missing from the group. | The dataset contains a group which does not have all mandatory concepts defined in the DSD. |  |
| 150 | 017 | Concept {0} is assigned to the incorrect group. | The encountered group in the dataset contains a concept which is not defined in the DSD. | Replaced by 011 and “Xml unexpected attribute”[[1]](#footnote-1) |
| 150 | 018 | XML error - The dataset contains an invalid node. | Appears when an unexpected node exists in the dataset file. |  |
| 150 | 019 | Error never occurs | A flat dataset is a dataset that consists of observations, which are not grouped in series. Each observation gives values for all dimensions ("all dimensions are under observation"). It is an error to have a series element in a flat dataset. |  |
| 150 | 020 | Error never occurs | Non flat dataset expected, (dimension At Observation is not 'All Dimensions') but flat dataset encountered. Flat datasets contain only observation elements |  |
| 150 | 021 | XML error - Unexpected text ''{0}'' found at node ''{1}'' | Unexpected text is found as children of one sdmx node which does not contain text. Sdmx node names are kept in an internal structure and has the names like: Header, Series, Obs, Group etc. This error message appears when the dataset contains as children of these elements. |  |
| 150 | 022 | XML error - Dataset header fails to reference a provision agreement, dataflow, or DSD. | Dataset header fails to reference a provision agreement, a dataflow, or a data structure definition |  |
| 150 | 023 | XML error - Dataset does not contain a header. | Dataset does not contain a header |  |
| 150 | 024 | XML error - Dataset structure reference incomplete. | The message appear if the referenced structure is incomplete, ie the agencyId, id or maintainable ParentId are missing or empty. | Replaced by 022 validation (which was not implemented from the beginning) |
| 150 | 025 | XML error - Invalid DSD reference. | Dataset structure reference invalid, could not process reference, no Ref node or URN node found |  |
| 150 | 026 | Attribute {0} is not defined in DSD. | An attribute at dataset level is present in data file but it is not defined in the DSD. | Replaced by 009 and 011 (depending on the case) and “Xml undefined attribute” [[2]](#footnote-2) |
| 150 | 027 | Expected component {0} must be an attribute, but is {1}. | Another component appears as a dataset attribute in data file. |  |
| 150 | 028 | Attribute {0} incorrectly attached to {2} instead of to {1}. | The dataset has an attribute with different attachment level. | Replaced by the errors 009 or 011. |
| 150 | 029 | {0} ''{1}'' is reporting value ''{2}'' which is not a valid representation in referenced codelist ''{3}''. | An attribute at dataset, series or observation level has a value which is not valid in the referenced codelist. |  |
| 150 | 030 | {0} {1} is reporting invalid value {3} which is not of expected type {2}. | Appears when reported value of a concept is unexpected. |  |
| 150 | 031 | Component {0} in group {1} not defined in DSD {2}. | The dataset contains groups which contains components that are not defined as group components in the DSD. | Replaced by 011 and “Xml unexpected attribute”[[3]](#footnote-3) |
| 150 | 032 | Observation missing time dimension for time series data. | Observation missing time dimension for time series data. |  |
| 150 | 033 | Observations not allowed for this dataset. | Appears if there is a constraint on the dataset which does not allow observations. |  |
| 150 | 034 | Observation time ''{0}'' is before the expected reporting period start date "{1}". | Appears if there is a constraint which specify report start date and the observation time is before this date. |  |
| 150 | 035 | Observation Time ''{0}'' is after the expected reporting period end date "{1}". | Appears if there is a constraint which specify report end date and the observation time is after this date. |  |
| 150 | 036 | Series not allowed in this dataset. | Appears if there is a constraint on the dataset which does not allow series. |  |
| 150 | 037 | Series key {0} not allowed. | Appears if the dimension is not allowed in the key due to an existing constraint. |  |
| 150 | 038 | Illegal Series key {0} contains invalid value "{1}" not defined in DSD for {2} {3}. | Appears when the series key contains some value which is disalowed by constraints in DSD. |  |
| 150 | 039 | Duplicate observation found: {0} | Appears when more than one observation are found in one series. |  |
| 150 | 040 | Data validation failed : {0} | It appears when a custom validation rule does not pass.  Data validation failed : Failure("checkDataSetAttributeValue") | Error never occurs |
| 150 | 041 | Cross-sectional component {0} is incorrectly attached to {2} instead of to {1}. | The cross-sectional component is attached to a wrong level. |  |
| 150 | 042 | Invalid date format "{0}". | Appears if the TIME\_PERIOD attribute does not match the TIME\_FORMAT. | Validation left for a future release. (see SDMXCONV-310). |
| 150 | 043 | Structure type wrongly references {1} instead of {0}. | If the dataset header contains a URN reference to other artefact then expected. |  |
| 100 | 044 | The dataset references dataflow “{0}” which could not be resolved. | The structure file supplied to the STRUVAL service call does not contain a dataflow (identified by agency, name, and version) that is referenced from the dataset. |  |
| 100 | 045 | The dataset references DSD “{0}” which could not be resolved. | The structure file supplied to the STRUVAL service call does not contain a DSD (identified by agency, name, and version) that is referenced from the dataset. |  |
| 501 | 046 | Component attribute {0} with parent {1} not supported. | When an XML attribute is in the wrong element. |  |
| 501 | 047 | Cannot read dataset for structure of type: '{0}' | If the dataset has a structure reference which is neither DSD or Dataflow | The message is not testable in Converter applications because we don't support structures other than DSD and Dataflows. This error message appears only if struval is used as API. |
| 501 | 048 | The DSD {0} is missing a time dimension. | DSDs that STRUVAL can process must contain a time dimension. | The error message 501 - 048 - "The DSD {0} is missing a time dimension" is in conflict with 150 - 004 - "The DSD {0} used does not define a time dimension, required for the time series data". They both refer to the same thing, the first was created by Intrasoft ( hence the 4 digits message code) in order to cover some bugs in SdmxSource ( see [SDMXSOURCE-10](https://webgate.ec.europa.eu/CITnet/jira/browse/SDMXSOURCE-10)) whereas the 004 is thrown by Sdmx Source (so it's standard). Starting with Converter 6.9.1 the first one (9946) is no longer used. |
| 501 | 049 | Cannot validate the header of format {0}. | Appears when the struval tries to validate a header but the given dataset file is not detected as been one of the following formats: COMPACT\_2\_0, GENERIC\_2\_0, CROSS\_SECTIONAL\_2\_0, UTILITY\_2\_0, GENERIC\_2\_1, GENERIC\_2\_1\_XS,COMPACT\_2\_1 or COMPACT\_2\_1\_XS. |  |
| 150 | 050 | Property not found {0} | Appears when the validation fails, because of missing input or structure file |  |
| 140 | 051 | Configuration Error {0} | Appears when Excel Data Reader was not configured correctly |  |
| 140 | 052 | Excel data reader error {0} | Appears when Reading the excel file was not possible |  |
| 140 | 053 | Invalid Parameters detected {0} | Appears when misconfiguration exists inside Parameter Sheet or Mapping Sheet |  |
| 150 | 054 | Error While Processing XML : {0} | Appears when XML structure validation fails |  |

# Annex T – Instructions to Access Registry over HTTPS

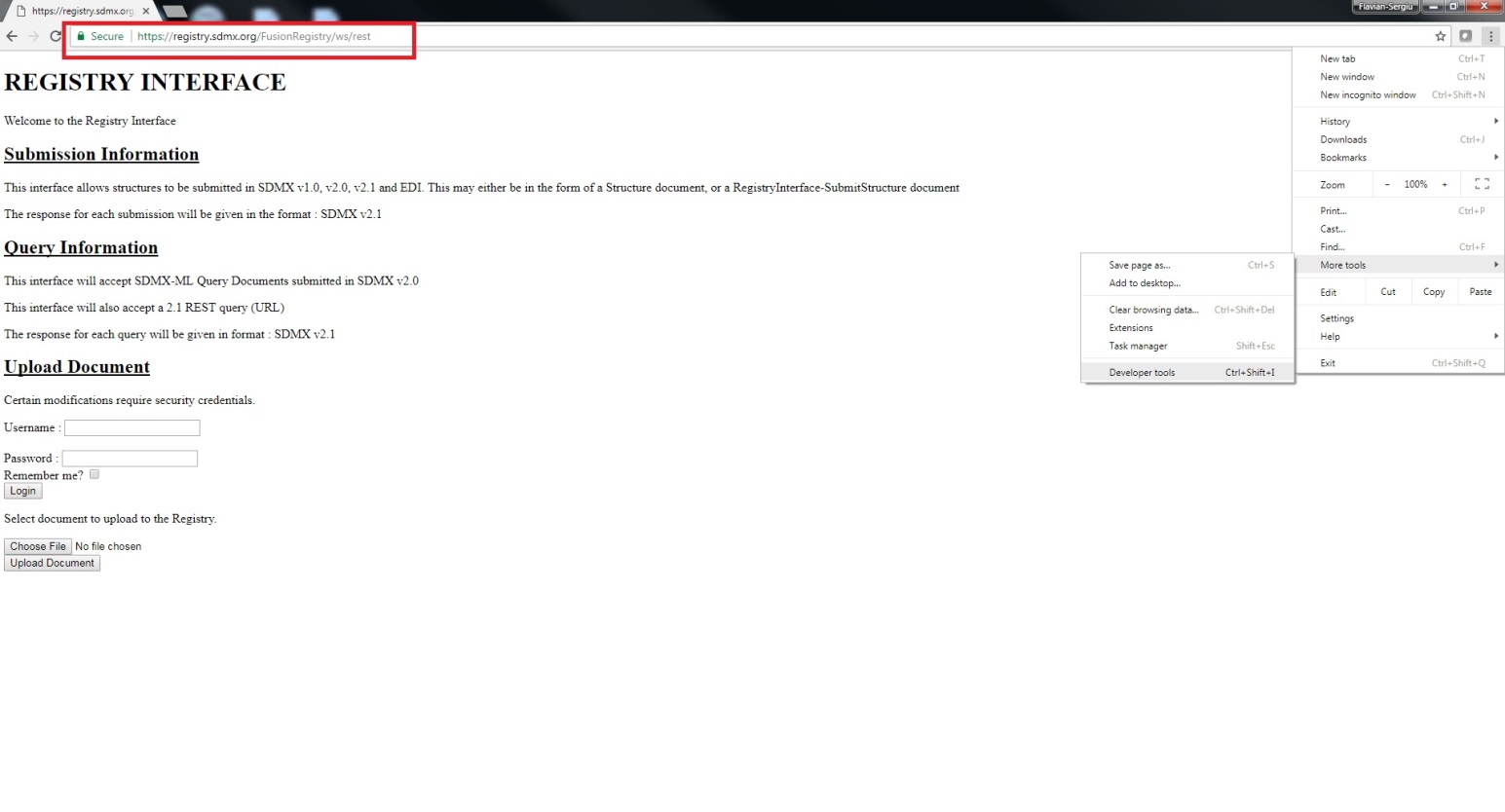
Being able to use the registry from some sites for rest/wsdl requests from Java they needed to have a certificate added in java trust store. Trying to access it from SDMX Converter web service for example triggers an error like (in logs): *javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested targett sun.security.ssl.Alerts.getSSLException..*

The error appears because whenever Java attempts to connect to another application over SSL (e.g.: HTTPS, IMAPS, LDAPS), it will only be able to connect to that application if it can trust it. The way trust is handled in the Java world is that you have a keystore (typically $JAVA\_HOME/lib/security/cacerts), also known as the truststore. This contains a list of all known Certificate Authority (CA) certificates, and Java will only trust certificates that are signed by one of those CAs or public certificates that exist within that keystore.

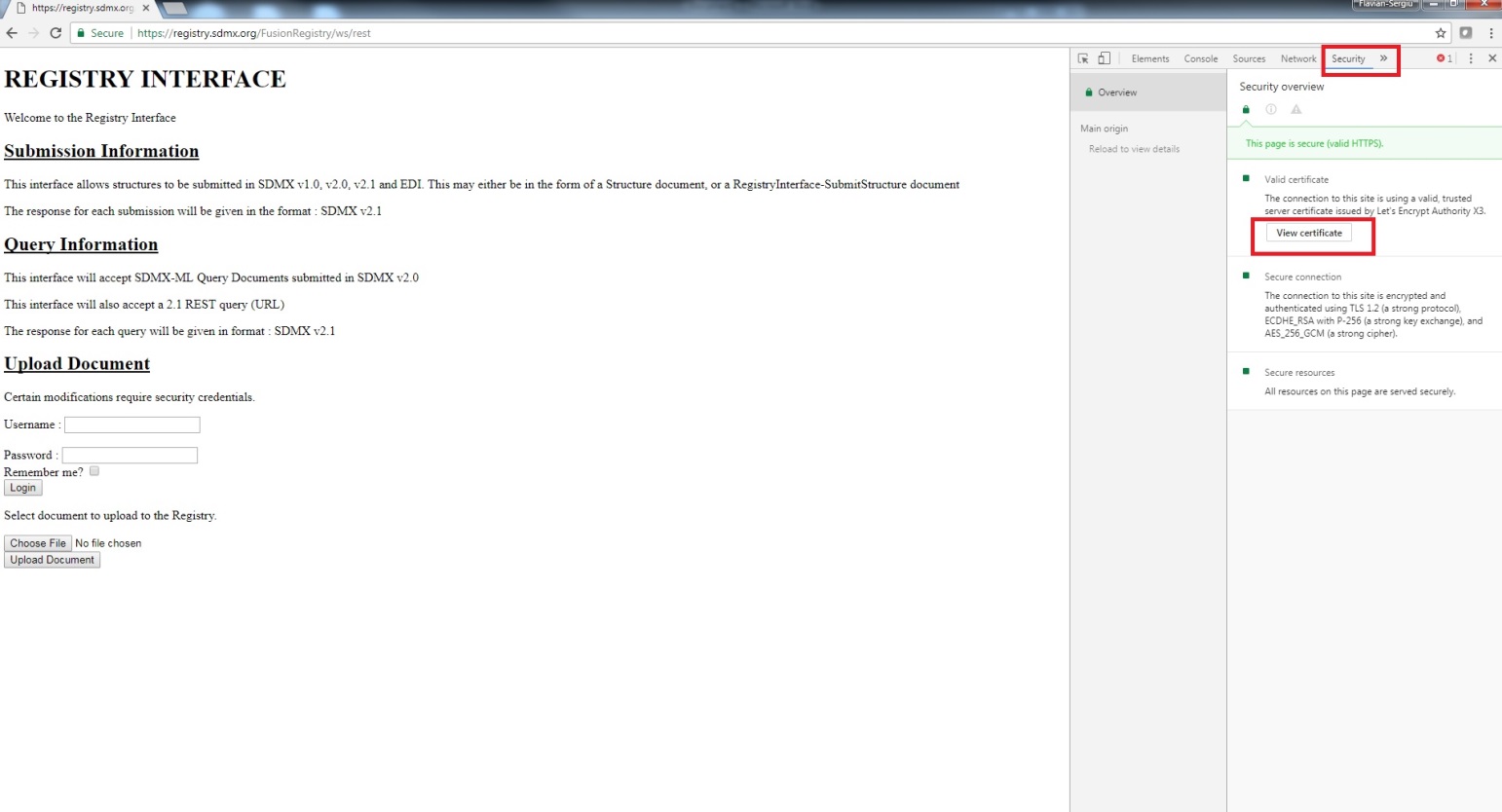
## Obtain your own certificate

To obtain the certificate please follow the steps provided in the attached word document. Please contact us if you have any questions.

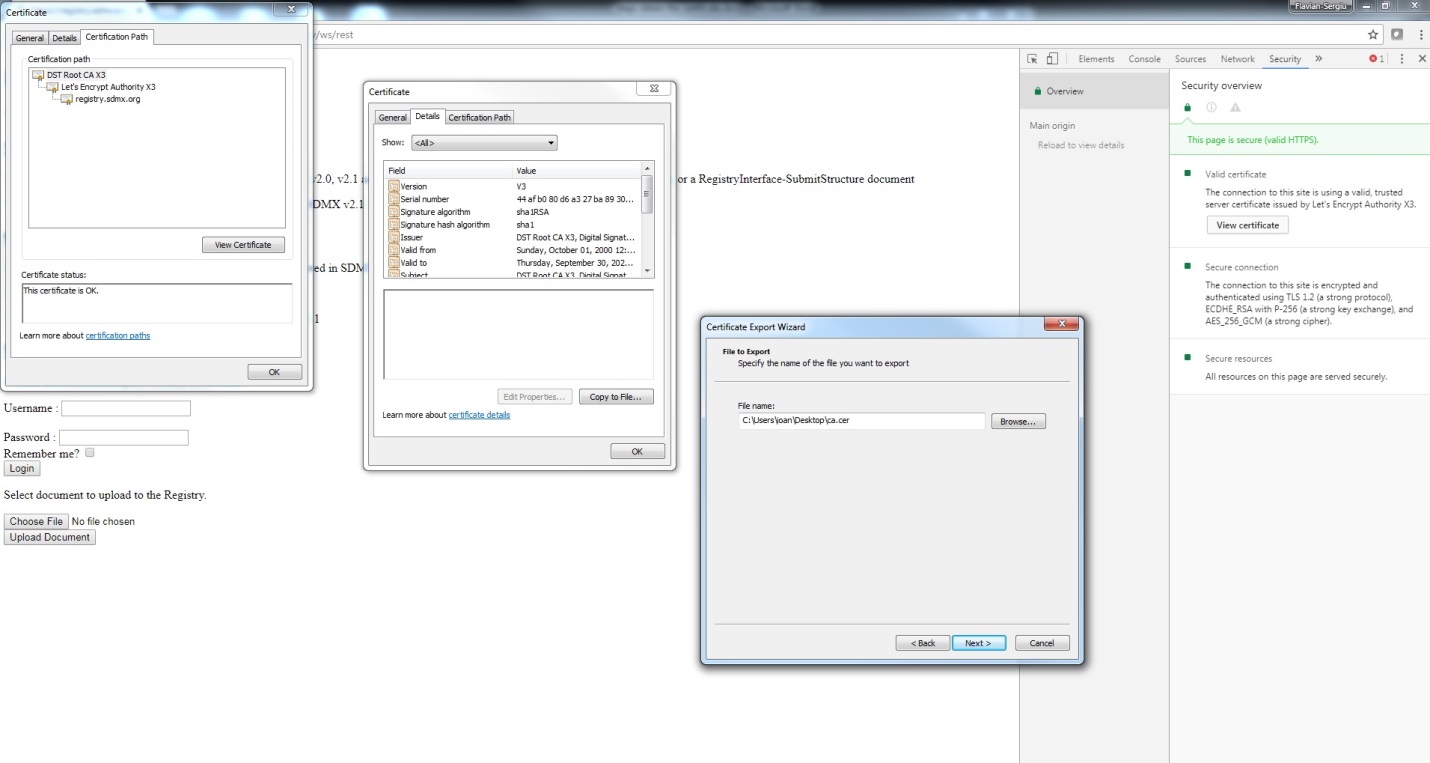
We expect that details to be provided by the Registry that needs this but a method to obtain the certificate and put it in trust store java is presented in the following steps:

**Step 1**: For you to view the SSL certificate details please open the URL <https://registry.sdmx.org/FusionRegistry/ws/rest> with Google Chrome and go to the Three Dots Menu -> More Tools -> Developer Tools. 

**Step 2**: Click on the Security Tab. This will give you a Security Overview with a View Certificate Button.



**Step 3**: In the certificate window click Certification Path->Select the ROOT-> View Certificate-> Copy to File for the Certificate Export Wizard, and click Next. Select DER encode binary X.509 (.CER) option, select a name and a path, for example (C:\Users\ioan\Desktop\ca.cer), click Next and Finish to save.



**Step 4**: In the command line. Convert the root certificate to DER format. This can be done with help of the **openssl** toolkit, where ca.cer is the original certificate filename in PEM format, and ca.der the filename to output, in DER format (which the Java keytool utility can understand). If you were able to obtain the root certificate in DER format, skip this step:

openssl x509 -in ca.pem -inform pem -out ca.der -outform der

**Step 5**: Validate the root certificate content. Ensure that the Java keytool can parse the certificate and display its content, with the following cmd command:

keytool -v -printcert -file ca.der

**Step 6**: Import the root certificate into the JVM trust store, with the following command:

keytool -importcert -alias startssl -keystore “C:\Program Files\Java\jdk1.7.0\_79\jre\lib\security\cacerts” -storepass changeit -file ca.der

(the default password for the CA store is changeit)

The keytool will prompt you for confirmation, enter yes to complete the operation.

**Step 7**: Verify that the root certificate has been imported with the command:

keytool -list -v -keystore “C:\Program Files\Java\jdk1.7.0\_79\jre\lib\security\cacerts” -alias startssl

**Step 8**: Restart JVM/PC

## Use an existing certificate

You can define a certificate to be loaded from the application that contains all the necessary certificates for the connection with registry. Inside the properties file (configuration file) you can set the path for the file to be used. The settings for configuring the certificate to be used when connecting to the registry service are the following:

* registry.jks.path=
* registry.jks.password=changeit

The setting *registry.jks.path* is set with the path of the jks/cacerts file to be used. If this setting is empty, the default keystore from the project resources is used (cacerts). The setting *registry.jks.password* is for setting the password for the keystore.

# Annex U – Reporting Period for SDMX 2.1 Output Formats

With SDMX 2.1 the time periods can be provided in relation to a reporting year. Saying that Reporting Period. If Reporting Period is used then it will be as an attribute of DataSet under name REPORTING\_START\_YEAR\_DAY of format **--mm-dd**. For example --03-01 means that the reporting year starts with first of March. Then when a Time Period is provided e.g. 2013-M02 it means it is the second month of the Reporting Year, in this case the Gregorian Period would be 2013-04 (first month is between 1st of March and 1st April, second month is between 1st April and 1st of May, hence the Gregorian Time is April with TIME\_FORMAT P1M).

The following cases exist:

* Conversion from a format without Reporting Period to SDMX 2.1 without using the Reporting Period then the same Gregorian Time remains with no action needed from the user
* Conversion from an SDMX 2.1 without Reporting Period also needs no action from the user
* Conversion to SDMX 2.1 when a Reporting Period is needed. In this case the REPORTING\_START\_YEAR\_DAY must be specified otherwise the default is --01-01. The TIME\_PERIOD from the input value is transformed accordingly to the REPORTING\_START\_YEAR\_DAY.

For example, when Gregorian Time Period is 2010-03 and the user wants the output file to be expressed reporting it to the Reporting Year and provides REPORTING\_START\_YEAR\_DAY= --05-01 then the Time Period in the output file will be calculated starting with the previous year 2009-05-01 meaning **2010-M10** (the Gregorian time 2010-03 is the tenth month from the beginning of the reporting year day).

* Conversion from an SDMX 2.1 with Reporting Period to a format that does not need a Reporting Period (it can also be another SDMX 2.1 format). The starting from REPORTING\_START\_YEAR\_DAY and TIME\_PERIOD in the input, the Gregorian time is calculated. E.g. REPORTING\_START\_YEAR\_DAY= --05-01 TIME\_PERIOD = **2010-M10**,in input file then it will be 2010-03 in output file (the Reporting Period Year is considered previous year).

# Annex V: The structure of a transcoding sheet for Excel parsing

The Transcoding sheet inside an Excel must have the following structure:

|  |  |  |
| --- | --- | --- |
| Text | Dimension | Value |
| something | STS\_ACTIVITY | New Value |
| nothing | STS\_ACTIVITY | N100CO |
| cool | STS\_ACTIVITY | N100CO |
| ble | ADJUSTMENT | W |

The name of a Transoding sheet must start from “Trans” and the first row must have the headers as appear above for the converter to pare it correctly.

# Annex W: The structure of a properties file

The following parameters are used in the properties files for csv, multi level csv and sdmx csv validation.

csv.delimiter=;

csv.level=1

csv. headerrow=NO\_COLUMN\_HEADERS

csv.quotecharacter=”

errorIfEmpty=true

default.validation.inlineReportFormat=true

The following parameters are used in the properties files for flr.

flr. headerrow=NO\_COLUMN\_HEADERS

flr.expilicitmeasures=true

default.validation.inlineReportFormat=true

The following parameter is used in the properties file for xml files:

default.validation.inlineReportFormat=true

The following parameter is used in the properties file of Converter-Webapp only. It concerns the default timeout in the process of uploading input files, expressed in seconds. The default timeout is set to 180 seconds (3 minutes).

default.webapp.input.timeout=180

# Annex X: Internal SDMX CSV format

Any format could be converted to internal SDMX CSV format. The iSDMX\_CSV is a format with metadata information about the input file data. The additional information added differs from format to format. For example, for XLS the cell of each observation is written as last column, for CSV files only the row and for SDMX-ML the line and row of each observation. Moreover, a column with the format of the input file is written.

The iSDMX\_CSV file is the same with the SDMX\_CSV except from the two last columns and that the delimiter is semicolon as well as double quotes are used by default.  
If in an iSDMX\_CSV file there are new line, carriage return or end of line characters the user has the option to remove(trim) those characters by setting the default.iSdmxCsv.adjustment property of the configuration file to true. Furthermore, if the input file contains semi-colon(‘;’) within the value of an attribute this character gets transformed to comma(‘,’). If the aforementioned property is set to false those special characters remain in the output iSDMX\_CSV file.

An example of an internal SDMX\_CSV file is provided below:



# Annex Y: Character Encoding

The input files the user provides for Validation or Conversion should only have UTF-8 encoding, as UTF-8 encoded files are only supported.

# Annex Z: How Conversion/Validation Handles Excel Formulas

## 1.1 Formulas in External Parameter Sheet

Conversion and Validation application do not support custom destinations denoted in an external parameter sheet with the use of formulas, in excel files.

## 1.2 Manual Formulas Calculation Detection

Conversion and Validation do not support detection of event specifying manual formulas calculation in excel files. The following flags “Application.calculation = VbManual”, “Application.EnableEvents = false”, are not supported from Excel Readers.

1. “Xml unexpected attribute” is shown only when the SDMX attribute is also an xml attribute. [↑](#footnote-ref-1)
2. “Xml undefined attribute” appears only when the sdmx attribute is also an xml attribute. [↑](#footnote-ref-2)
3. “xml unexpected attribute” appears only when the sdmx component is also a xml attribute. [↑](#footnote-ref-3)