European Commission – Eurostat/B1

WORKING DOCUMENT

ESS data validation project

Main types of validation rules for ESS data

2 February 2018

Document Service Data

Type of Document	Deliverable		
Reference:	Main types of Validation rules V0.99.docx		
Version:	1.03	Status:	Draft
Created by:	Vincent TRONET	Date:	02/02/2018
Distribution:	European Commission – Eurostat/B1, B3, B5		
For Internal Use Only			
Reviewed by:	Task Force Validation, ESSnet Validat Integration, Eurostat B1/B3/B5, Eurostat Pilot domains		
Approved by:			

Document Change Record

Version	Date	Comments
0.81	6 Nov 2017	Chapters 1, 2, 3, 4.1 and 4.2 ready for review First version of chapters 4.3 and 4.4 drafted but further work is needed before review Annexes and reference list to be added.
0.99	21 Dec 2017	All Chapters covering the 21 main types of validation rules are drafted with a first version of the VTL scripts for a pre-release of VTL 2.0. This version still needs a review, in particular for the following points: - the description of the parameters and algorithm still needs to be harmonised and probably made more clear, - the 2 “most complex” types of validation rules (VNO: Values are Not Outliers in Time Series and VMP: Values for Mirror Data are Plausible, - VTL scripts to be adapted to official version 2.0 and to be optimised (simplified)
1.03	02 Feb 2018	- Description parameters and algorithm harmonised - “VNO” main type updated to have examples closer to the default. - Feedback ESSnet and TF Validation taken into account

Contact Information

EUROSTAT
Vincent TRONET
Unit B1
BECH Building
Tel.: +352 4301 32906
Email: Vincent.Tronet@ec.europa.eu

Table of Contents

1. Purpose of the document

The purpose of this document is to support the description of validation rules in specific statistical domains. It aims at covering the 21 main types of validation rules needed for validating ESS data (i.e. statistical data to be provided to Eurostat according to legal commitments or voluntary agreements).

It can be used for instance:

- as a check-list for the coverage of validation rules in domains,
- to structure validation handbooks in domains (documents describing validation rules in domains),
- to easily setup and maintain validation rules in the future ESS service "Validation Rule Manager",
- to help understanding how VTL, the standard Validation and Transformation Language, can be used to validate ESS data,
- to identify the main operators needed in VTL to validate ESS data.

The 21 main types of validation rules aim at covering:

- 100% of the validation rules related to the structure and the code-lists used in datasets,
- close to 100% of the content validation rules applied inside files,
- at least 80% of the content validation rules applied between files (or between files and databases).

They refer to validations performed on data files transmitted to Eurostat. They can be related to validations performed within the file transmitted to Eurostat (validation levels 0 or 1) or between the file transmitted to Eurostat and data available in other files or databases (validation levels 2 to 5)[1].

Are not covered in this document:

- Advanced validations or data analysis that are performed in the Eurostat databases, after data have been imported in production environments,
- Complex or non-common validation rules. Many of them can nevertheless be derived from the 21 main types (i.e. the main types can be used as starting point to define many rules not covered).

1 *See Annex-I «Validation levels»*.

Validation rules are linked to a data structure and to code-lists defined in a repository such as a SDMX repository. They apply to records with 3 types of fields:

- the key or dimension fields (that constitute the identification or ID_Key when combined together),
- the observation value(s),
- the attributes.

They can be classified in different groups according to different perspectives:

- If they are mandatory for all datasets or they apply only to some,
- Validation Levels 0 to 5 depending from the distance to complementary data or information needed to perform the rules,
- The share of responsibilities of Eurostat and data providers in data validation (usually at least rules within the file are checked by data providers),
- The availability of the information in the SDMX context or not (structural validation or content validation),
- Their applicability to micro-data or not (considering that they can all apply to aggregated data),
- Severity levels (E: Error, W: Warning, I: Information)[2],
- The concept validated (observation value or not),
- The possibility to have "pre-conditions" to filter the cases where the rules are applicable.

SDMX provides a framework to define data structures and code-lists. When SDMX is used, validation rules of level zero and several validation rules of level 1 can be checked with a structural validation service. VTL is then used to describe content validation rules applicable to levels 1 to 5.

Provided that the data structure is described in a data structure repository, the validation rules can be applied to SDMX-ML files but also to structured flat files such as CSV for fixed length records files.

In this document, for the sake of clarity validation rules are described for the most common case of one single observation value (OBS_VALUE) per record.

Each validation rule described for an individual domain can be considered as either:

1. Directly linked to a main type of rule, meaning that the parameters described in this document for the relevant main type of rule are sufficient to define the rule,
2. Derived from a main type of rule, meaning that the parameters described below can help to define the rule, but further information is needed[3],
3. Not linked to a main type of validation rule, meaning that the rule will have to be fully defined without any reference to one of the main types.

---

2 *see Annex-II for an extract of the "Business Architecture for ESS Validation*

3 *This applies in particular to validation rules of level 2 and more that are described in this document (when relevant) only for validation level 1*

The main input for this document have been:

- document “2-4 - Exhaustive and detailed typology of validation rules_v01306” from Angel Simon (Sept. 2013),
- an inventory of existing validation rules covering the following pilot statistical domains: National Accounts, Short Term Business Statistics, Energy Statistics, Asylum statistics, animal and milk statistics.

2. Structure of the document

Chapter 3 introduces a fictive statistical domain (“Intra-EU travellers”) linked to a fictive Data Structure Definition. It is used in this document as basis for giving examples that illustrate validation rules for each main type.

Chapter 4 describes the 21 main types of validation rules with the following information:

- A code on 3 positions that reflects first the concept validated (E: Envelope, FL: FiLe, FD: FielD, C: Code, R: Record, V: Value) and then “what is validated”,
- A title that reflects first the concept validated and then “what” is validated,
- A detailed description in plain English,
- The mandatory status or not of the type of rule for all datasets,
- The possibilities of default values or formula,
- The link with validation levels,
- The possibilities for pre-conditions (conditions or filters to select specific cases to which the rule applies),
- The availability in the SDMX environment (sufficient for structural validation) or not,
- The applicability to micro-data,
- The possible severity level(s) in case of failure,
- A table with the parameters, the algorithm and, if relevant, the VTL script to perform the rule,
- At least one example to illustrate the type of rule with good and bad data based on the fictive domain “Intra-EU travellers”.

Annex-I: explains the validation levels (from zero to 5),

Annex-II: gives an extract of the “Business Architecture for ESS Validation” that provides explications on severity levels and the related process in case of failure,

A list of references is also provided for further reading.

3. Summary table with the main types of rules and fictive domain used to illustrate the examples

The table below provides a summary of the 21 main types of validation rules described in this document with their main characteristics.

Rule type	Mandatory	Default	Validation level						Pre-condition	SDMX (STRUVAL)	Micro data	Severity level			Comments
			0	1	2	3	4	5				E	W	I	
(EVA) Envelope is Acceptable	X		X							(X)	X	X			
(FLF) File Format	X		X							X	X	X			
(FDD) Fields Delimiter	(X)	";"	X							X	X	X			Mandatory for CSV files
(DES) Decimals Separator	(X)	"."	X							X	X	X			Mandatory for CSV files (always "." For SDMX-ML)
(FDT) Field Type	X		X	(X)						(X)	X	X			Level 1 in case exceptions to the type are possible
(FDL) Field Length	X		X							X	X	X			
(FDM) Field is Mandatory or empty			X	(X)					(X)	(X)	X	X	(X)		Level 1 in case the status depends from datasets
(COV) Codes are Valid	(X)			X					(X)	(X)	X	X	(X)		Mandatory for key fields
(RWD) Records are Without Duplicate id-keys	(X)			X						X	(X)	X			Mandatory for aggregated (tabular) data Applicable for micro-data with id-keys
(REP) Records Expected are Provided				X	(X)				(X)		X	X	(X)		
(RTS) Records are all present for Time Series				X								X	(X)		No hole in time series
(RNR) Records' Number is in a Range	X	>=1		X	(X)				(X)		X	X	(X)	(X)	Default: at least one record (file not empty)
(COC) Codes are Consistent				X	(X)				(X)	(X)	X	X	(X)		
(VIR) Values are in Range		>=0		X	(X)				(X)	(X)	X	X	(X)	(X)	Default: values are zero or positive (Min=zero)
(VCO) Values are Consistent				X	(X)	(X)	(X)	(X)	(X)		X	X	(X)	(X)	
(VAD) Values for Aggregates are consistent with Details	(X)	=		X	(X)				(X)			X	(X)	(X)	Mandatory if aggregates and details are provided Default: aggregate = sum of details
(VNO) Values are Not Outliers in Time Series		MAD		X	(X)				(X)			(X)	X	(X)	Defaults: non seasonal data, most recent period, median value and AAGR on 5 periods and MAD
(VSA) Values for Seasonally Adjusted data are plausible				X	(X)				(X)			X	(X)	(X)	
(RRL) Records Revised are Limited in number or %		10%			X				(X)		(X)	(X)	X	(X)	Default: Records revised (upd., del., add) <= 10%
(VRT) Values are Revised within a Tolerance level		10%			X				(X)		(X)	(X)	X	(X)	Default: Values revised (updates only) <= 10%
(VMP) Values for Mirror data are Plausible		10%				X			(X)			(X)	X	(X)	Default: Diff mirror values <= 10%

The examples of validation rules in this document are fictive and based on a fictive domain (Intra EU Travellers), a fictive Data Structure Definition (DSD) and fictive datasets (called “INEUTRAV_T01_A” and “INEUTRAV_T02_Q”). Nevertheless, some similarities may be found with European statistics on tourism, asylum, air transport and external trade.

The usage of this fictive domain for the examples described in this document aims at simplifying the document as it allows describing only one data structure and “inventing” simple rules for all the main types of rules.

Two “Fictive tables” use the data structure described below: T01 (dataset “INEUTRAV_T01_A”: Intra-EU travellers – Annual data) and T02 (dataset “INEUTRAV_T02_Q”: Intra-EU travellers – Quarterly data).
They allow the transmission of time series from 2008 onwards (until the reference period specified in the envelope).

What is called “the envelope” in this document is the identification of the data file provided by the data sender when transmitting it to Eurostat. It can for instance be the identification provided to EDAMIS (the ESS tool for official transmission of data to Eurostat) or in the header of a SDMX-ML message. It usually contains at least the references to: (a) the sending country, (b) the dataset-id including its periodicity, (c) the reference period of the data.

The Validation rules appear in the table below next to the first or main concept to which they apply, except for the statistical observation (OBS_VALUE) where they appear next to the first or most important concept in addition to OBS_VALUE that is part of the validation rule. If only OBS_VALUE is considered in the validation rule, then the rule appears next to OBS_VALUE.

For example, in the table next page, rule “(G1) The aggregates (such as “TOTAL”) correspond to the sum of the details +/- 1% tolerance” checks the statistical observations (OBS_VALUE) for the aggregates and details of dimension AGE (Age Group). This rule appears then next to concept AGE instead of OBS_VALUE.
See more examples in the table below for rule types starting by “V” (for Values).

Domain « intra-EU travellers » - overview of data structure and validation rules:

Type	Field	Description	Format, size, exceptions	Content (code list and/or range)	Validation Rules (ID and name)	Rules type	Validation Levels	Severity
Key (Dimension)	**(A) TABLE**	Table-ID	AN(3)	CL_TABLE: *T01,T02*				
Key (Dimension)	**(B) FREQ**	Frequency of data	A(1)	CL_FREQ: *A (Annual),* *Q (Quarterly)*	(B1) For T01: A, For T02: Q (B2) Annual Values (in T01) correspond to the sum of the 4 quarters (in T02)	(B1) (COC) Codes are consistent (B2) (VAD) Values for aggregates are consistent with details	(B1) 1b (B2) 2	(B1) E (B2) W
Key (Dimension)	**(C) TIME_PERIOD**	Period (year or quarter) of data	AN(4..7)	*2008, 2009, …* *2008-Q1, 2008-Q2, …*	(C1) For T01: 2008,… For T02: 2008-Q1, … (C2) Time series are complete since 2008 (without missing period) (C3) The travellers in the most recent year do not differ from more than 2*MAD (2 times the Median Absolute Diff) compared to the median of the previous 5 years.	(C1) (COV) Codes are valid (C2) (RTS) Records are all present for Time series (C3) (VNO) Values are Not Outliers in Time Series	(C1) 1a (C2) 1c (C3) 1c	(C1) E (C2) E (C3) W
Key (Dimension)	**(D) REPORTING**	Reporting country	A(2)	CL_REPORTING: *BE,BG,CZ,DK,DE,EE,IE,EL,ES,FR, HR,IT,CY,LV,LT,LU,HU,MT,NL, AT,PL,PT,RO,SI,SK,FI,SE,UK*	(D1) Reporting country in the data is the same as the reporting country in the envelope	(D1) (COC) Codes are consistent	(D1) 1a	(D1) E
Key (Dimension)	**(E) PARTNER**	Partner country (country of departure or of destination)	AN(2..4)	CL_PARTNER: *EU28,* *BE,BG,CZ,DK,DE,EE,IE,EL,ES,FR, HR,IT,CY,LV,LT,LU,HU,MT,NL, AT,PL,PT,RO,SI,SK,FI,SE,UK*	(E1) PARTNER country is different from REPORTING country (PARTNER <> REPORTING)	(E1) (COC) Codes are consistent	(E1) 1b	(E1) E
Key (Dimension)	**(F) DIRECTION**	Direction of the trip (incoming or outgoing)	A(2..3)	CL_DIRECTION: *IN (INcoming from PARTNER),* *OUT (OUTgoing to PARTNER)*	(F1) Travellers reported by country X as Incoming from Country Y correspond to travellers reported by country Y as outgoing to country X with 10% tolerance	(F1) (VMP) Values for mirror data are plausible	(F1) 3	(F1) W
Key (Dimension)	**(G) AGE**	The age group to which the travellers belong	AN(2..7)	CL_AGE : *TOTAL,* *Y0_18 (from 0 to 18),* *Y0, Y1, Y2…, Y18* *Y19_64,* *Y19, Y20, …, Y64* *Y65_MAX,* *Y65, …., Y122* *UNK,*	(G1) The aggregates (such as “TOTAL”) correspond to the sum of the details +/- 1% tolerance (G2) The children that travel (Y0_18) represent less than half of the TOTAL travellers	(G1) (VAD) Values for Aggregates are consistent with details (G2) (VCO) Values are consistent	(G1) 1c (G2) 1c	(G1) W (G2) W
Key (Dimension)	**(H) ADJUST**	Seasonal adjustment	A(1)	CL_ADJUST: *N: Not adjusted* *S: Seasonally adjusted*	(H1) T01: only “N” is accepted (H2) T02: “N” and “S” are provided (H3) T02: for each year, the sum of the 4 quarters for seasonally adjusted data (“S”) do not differ from more than 1% compared to non-seasonally adjust. data ‘(“N”).	(H1) (COC) Codes are consistent (H2) (REP) Records expected are provided (H3) (VSA) Values for seasonally adjusted data are plausible	(H1) 1b (H2) 1c (H3) 1c	(H1) E (H2) E (H3) W
Measure	**(I) OBS_VALUE**	Number of travellers	N(1..9) or “NA”	>=0	(I1) >=1 (I2) Number of travellers do not differ more than 20% compared to previous version	(I1) (VIR) Values are in range (I2) (VRT) Values are revised within a tolerance	(I1) 1a (I2) 2	(I1) W (I2) W
Attribute	**(J) OBS_STATUS**	Status of the observation	A(0..1)	empty, P (Provisional)	(J1) empty for T01, empty or “P” for T02	(J1) (FDM) Field is mandatory or empty	(J1) 0	(J1) W

Explanations:

- Type:
  - *Key (i.e. Dimension or Identifier): combined all together, the keys constitute the identification (ID) key of the* data,
  - *Measure* – values of the observed phenomena (observation value),
  - *Attribute* – information about observed data,
- Field (i.e. Concept): field code,
- Description: field description,
- Field format, size, exceptions:
  - Format: A (Alphabetic), AN(Alphanumeric), N(Numeric), ND(Numeric with Decimals),
  - Size: "x": fixed size (x characters)," w..x" (between w and x characters), for type "ND" "w..x,y…z" (between w and x characters including y to z decimal figures),
  - Exceptions: list of special codes (or values) that are acceptable even if they do not respect the format and size (for example, an observation value should be numeric or contain the character "NA" which stands for Not Available).
- Content: contains either all the possible codes for the concept (e.g. "A", "Q", "M" for FREQ) or the range of acceptable values (either between 2 values or just a minimum or maximum value),
- Validation rules: an identification and a brief description of the validation rules including pre-conditions to apply the rules if any,
- Rules type: among the 21 main types of validation rules,
- Validation Levels: levels (0 to 5) of the validation rules[4],
- Severity: E (Error), W (Warning), I (Information).

4 *Validation Level 1 is broken down in levels "1a" which refers to fields, "1b" to records (between fields) and 1c to the file (between records)*,

4. Typology of validation rules

4.1 Basic file structure check (Level zero)

The group of basic file structure corresponds to preliminary check needed before the data can be further validated. The severity in case of failure of validation rules under this group is always “Error” as a failure may prevent IT services to process the files further. All types of rules under this group correspond to level 0 and are performed in the initial step of structural validation services.

For datasets described in the SDMX context, these types of validation rules are implicit and do not need to be further described. They can be checked automatically by structural validation services that use as input the relevant DSD (Data Structure Definition) as available in a SDMX registry.

The following 4 types of validation rules fall into this group:

- (EVA) Envelope is acceptable,
- (FLF) File format,
- (FDD) Fields delimiter,
- (DES) Decimals separator.

4.1.1 (EVA) Envelope is Acceptable

Check that the identification envelope of the data file is acceptable. The identification envelope usually refers to the Dataset-ID, the periodicity of data, the reference period(s) and the sender (country, organisation, user).

It can be considered in many cases as a security check that controls at data reception that the sender is allowed to send the dataset for the reference period(s). For transmissions of data to Eurostat, the identification envelope is provided to EDAMIS at the time of data transmission. It can also be included in the header of SDMX-ML files.

Profile:

- *Mandatory for all datasets*,
- *Validation level zero (file structure)*,
- *Covered by description of DSD in SDMX, but check to be done by data transmission tool*,
- *Relevant for aggregated and micro-data*,
- *Severity level: Error (envelope not acceptable)*,

- *Pre-conditions are possible on countries and datasets.*

Parameters, Algorithm and VTL script (for level 0):

Parameters	Algorithm	VTL
List of acceptable combinations in the envelope for countries/datasets/periods	Check that the elements of the envelope are in the list of acceptable combinations	Not relevant (This check is implemented in EDAMIS).

Examples:

- check that the reference period in the envelope is lower than the current period (in case data to be reported should refer to the past),
- check that the sending country is an EU country that is expected to report the dataset.

4.1.2 (FLF) File Format

Check that the format of the file corresponds to what was agreed (SDMX-ML, CSV, flat file with fixed length records, …). The specification of the format needs to be precise. For instance for SDMX-ML, it should be specified which version (e.g. 2.0 or 2.1) and which type of message (generic, compact, cross sectional, …). Usually one single file format can be accepted.

Profile:

- *Mandatory for all datasets,*
- *Validation level zero (file structure),*
- *Can be checked by a structural validation service based on a DSD present in a SDMX registry and that can work on SDMX-ML and CSV files,*
- *Relevant for aggregated and micro-data,*
- *Severity level: Error,*
- *Pre-conditions are not possible.*

Parameters, Algorithm and VTL script (for level 0):

Parameters	Algorithm	VTL
1. Detailed specifications of the file formats acceptable	Check that the file format is acceptable	Not relevant. This check is implemented in a structural validation service.

4.1.3 (FDD) Fields Delimiter

Check that only the expected character is used as field delimiter.

Profile:

- *Mandatory for CSV files,*
- *Validation level zero (file structure),*
- *Can be checked by a structural validation service based on a DSD present in a SDMX registry and that can work on SDMX-ML and CSV files,*
- *Relevant for aggregated and micro-data,*
- *Severity level: Error,*
- *Pre-conditions are not possible*
- *Default field separator: "';"*

Parameters, Algorithm and VTL script (for level 0):

Parameters	Algorithm	VTL

1. Field Separator (FS) **-> ";" *is the default value***	Check for each record, that the number of field separators (FS) corresponds to the number of fields in the DSD minus one	Not relevant. This check is implemented in a structural validation service.

4.1.4 (DES) Decimals Separator

Check for numeric fields which can contain decimals that, when used, only the expected decimal separator is used. The decimal separator should be different from the field separator (if any).

Profile:

- *Mandatory to specify for CSV files (default ".") . For SDMX-ML files, only "." is acceptable,*
- *Validation level zero (file structure),*
- *Can be checked by a structural validation service based on a DSD present in a SDMX registry and that can work on SDMX-ML and CSV files,*
- *Relevant for aggregated and micro-data,*
- *Severity level: Error,*
- *Pre-conditions are not possible*
- *Default: "." (for SDMX-ML files, only "." is acceptable),*

Parameters, Algorithm and VTL script (for level 0):

Parameters	Algorithm	VTL
1. Decimal separator (DS) **-> "." *Is the default value***	For each record, Check that each numeric field which can contain decimals, either contains only figures or figures and one decimal separator (DS) located before the last position	Not relevant. This check is implemented in a structural validation service.

4.2Basic intra-file check (Level zero and 1)

Validation rules from this second group are usually performed just after the basic file structure check. The severity in case of failure of validation rules under this group is usually "Error" as a failure may prevent in many cases IT services to process the files further. In this group, only validation rules falling under validation levels 0 and 1 can be found.

For datasets described in the SDMX context, these types of validation rules may be implicit in many cases and therefore may not need to be further described. They can be checked automatically by structural validation services that use as input the relevant DSD (Data Structure Definition) as available in a SDMX registry.

The following 5 types of validation rules fall into this group:

- (FDT) Field Type,
- (FDL) Field Length,
- (FDM) Field is Mandatory or empty,
- (COV) Codes are Valid,
- (RWD) Records are Without Duplicates,

4.2.1 (FDT) Field Type

Check the correct type of content is used in the field. It can be Alphabetic, Alphanumeric, Numeric or Numeric with Decimals. Exceptions to the type can be listed.

Profile:

- *Mandatory,*
- *Validation level zero (file structure) or 1a (field)*
- *Validation usually of level zero, in case of exception(s) to the type it can be level 1a,*
- *Can be checked by a structural validation service based on a DSD present in a SDMX registry or by a content validation service linked to a DSD,*
- *Relevant for aggregated and micro-data,*
- *Severity level: Error,*
- *Pre-conditions are not possible.*

Parameters, Algorithm and VTL script (for level 1a):

Parameters	Algorithm	VTL
1. Field (F) 2. Field Type (FT): - Alphabetic, - Alphanumeric, - Numeric, - Numeric with Decimals 3. *Exceptions to type: Codes (C1, … Cn)*	Check for each record, that the content of Field (F) is of field type (FT) *or is in the code-list of accepted exceptions (C1, …Cn)*.	Not relevant. This check is implemented in a structural validation service.

Example (Ia): Check that field OBS_VALUE is numeric or contains "NA[5]".

Validation level	1a (field)
Severity	Error
Pre-condition	None

Parameters	Algorithm	VTL
1. Field (OBS_VALUE) 2. Field type (Numeric) 3. *Exceptions to type "NA"*	Check for each record, that the content of Field (OBS_VALUE) is of type (Numeric) *or is "NA"*	Not relevant. This check is implemented in a structural validation service.

5 "NA" (Not Applicable) is used in some domains that report data for instance in CSV format. In SDMX, for numeric field "OBS_VALUE", "NaN" (Not a Number) is also accepted as a standard exception.

Example of good and bad data:

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T02	Q	2016Q3	DE	ES	IN	TOTAL	N	20	
T02	Q	2016Q3	DE	ES	IN	TOTAL	S	15x	
T02	Q	2016Q3	DE	ES	OUT	TOTAL	N	NA	P
T02	Q	2016Q3	DE	ES	OUT	TOTAL	S	:	P

Bad case: 15x and ":" are not numeric and different from "NA".

4.2.2 (FDL) Field Length

Check that the length of the data in the field is acceptable.

Profile:

- *Mandatory,*
- *Validation level zero (file structure),*
- *Can be checked by a structural validation service based on a DSD present in a SDMX registry,*
- *Relevant for aggregated and micro-data,*
- *Severity level: Error,*
- *Pre-conditions are not possible.*

Parameters, Algorithm and VTL script (for level 0):

Parameters	Algorithm	VTL
1. Field (F) 2. Field length: - Case 1: fixed number of characters: x - Case 2: variable number of characters: Min, Max - Case 3: numeric with decimals: Min, Max, MinDec, MaxDec	Check for each record, that length of Field (F) is Case 1: x characters Case 2: between Min and Max characters Case 3: between Min and Max characters and the number of decimal figures is between MinDec and MaxDec	This check can be implemented in a structural validation service or as a VTL script. Case 1: check(**length**(Dataset-ID# **F**) = **x** **errorcode**("F field length should be x characters"), **errorlevel**("Error")) Case2: check(**length**(Dataset-ID# **F**) >= **Min** **and length**(Dataset-ID# **F**) <= **Max**, **errorcode**("**F** field length should be from **Min** to **Max** digits"), **errorlevel**("Error")) Case 3: check(between((**length**(Dataset-ID# **F**) – instr(Dataset-ID# **F**,"**.**")), MinDec, MaxDec), **errorcode**("**F** field length should be from **Min** to **Max** digits"), **errorlevel**("Error"))

For Case 3, "MaxDec" should always be lower to "Max" (the number of decimal figures should be lower to the "Max" number of characters of the field).

Example (Ea): Check that the length of field PARTNER is between 2 and 4 characters

Validation level	0
Severity	Error
Pre-condition	None

Parameters	Algorithm	VTL
1. Field (PARTNER) 2. Field length: - Case 2 - variable number of	Check for each record, that length of Field (PARTNER) Case 2: is between 2 and 4	check(**length**(INEUTRAV#PARTNER) >= 2 **and length**(INEUTRAV#PARTNER) <= 4, **errorcode**("PARTNER field length should be between 2 and 4 characters"), **errorlevel**("Error"))

characters: Min=2, Max=4	characters	

Example of good and bad data:

Field length PARTNER is between 2 and 4 characters.

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T02	Q	2016Q3	DE	ES	IN	TOTAL	N	20	
T02	Q	2016Q3	DE	EUR28	IN	TOTAL	S	150	
T02	Q	2016Q3	DE	E	OUT	TOTAL	N	21	P
T02	Q	2016Q3	DE	EU28	OUT	TOTAL	S	160	P

Bad case: “E” and “EUR28” are not between 2 and 4 characters long.

4.2.3 (FDM) Field is Mandatory or empty

Check that the field is mandatory (always with content) or is always empty.

Profile:

- *Optional (Not applicable to keys which are always mandatory, applicable to observation values or attributes only if pre-conditions drive the mandatory or empty status of their content)*
- *Validation level zero (file structure), 1a (in a field) or 1b (in a record)*
- *Validation usually of level zero, in some cases it can be level 1a (e.g. if a field should be empty only for some datasets that share the same DSD),*
- *Can be checked by a structural validation service based on a DSD available in a SDMX registry or by a content validation service linked to a DSD,*
- *Relevant for aggregated and micro-data,*
- *Severity level: usually Error, in some cases Warning*
- *Pre-conditions are possible,*
- *Default: “Mandatory” => The field should always be filled with something.*

Parameters, Algorithm and VTL script (for level 1a):

Parameters	Algorithm	VTL
1. Field (F) 2. Mandatory level: - Case 1: "**Mandatory**" (is filled): - Case 2: "Empty" (is empty) ***=> Default: Case 1: "Mandatory"***	Check for each record, that the content of Field (F): Case1: "**Mandatory**": is filled Case2: "Empty": is empty.	**//Option 1 (without pre-condition): direct rules** //Case 1: "**M**" Mandatory Check (**not isnull**(Dataset-ID#**F**), **errorcode**("**F** should be filled"), **errorlevel**("Error")) //Case 2: "**E**": Empty Check (**isnull**(dataset-ID#**F**), **errorcode**("**F** should be empty"), **errorlevel**("Error")) **//Option 2 (with pre-conditions): datapoint ruleset** **define datapoint ruleset** PresenceOfContent**F** (*table*, **F**) **is** **when** (*pre-condition-1)* **then not isnull**(**F**) // Case 1: "**M**" Mandatory **errorcode**("**F** should be filled for pre-condition-1") **errorlevel**("Error"); **when** *(pre-condition-2)* **then isnull**(**F**) // Case 2: "**E**": Empty **errorcode**("**F** should be empty for pre-condition-2") **errorlevel**("Error"); **end datapoint ruleset;** check (Dataset-ID, PresenceOfContent**F**)

Example (J1): Check that for table T01, field OBS_STATUS is always empty

Validation level	1b (record)
Severity	Warning
Pre-condition	Applicable only to table **(T01)**

Parameters	Algorithm	VTL
1. Field (OBS_STATUS) 2. Mandatory level: - Case 2: "Empty" (is empty)	Check for each record, that the content of Field (OBS_STATUS) Case2: "Empty": is empty.	**//Option 1 (without pre-condition): direct rules** Check (**isnull**(INEUTRAV#**OBS_STATUS**), **errorcode**("**OBS_STATUS** should be empty"), **errorlevel**("Warning")) **//Option 2 (with pre-conditions): datapoint ruleset** **define datapoint ruleset** PresenceOfContent (TABLE, **OBS_STATUS**) **is** **when** TABLE = "T01" **then isnull**(**OBS_STATUS**) **errorcode**("**OBS_STATUS** should be empty for T01") **errorlevel**("Warning"); **end datapoint ruleset;** check (INEUTRAV, PresenceOfContent)

Example of good and bad data:

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T01	A	2016	DE	ES	IN	TOTAL	N	20	
T01	A	2016	DE	ES	OUT	TOTAL	N	21	P

Bad case: "P" in "OBS_STATUS" whereas it should be empty for table T01.

4.2.4 (COV) Codes are Valid

Check that the codes used in a field are in a list (or a table) of acceptable codes.

Profile:

- *Mandatory for key fields, not relevant for OBS_VALUE and optional for attributes (some may be linked to list of codes, some not),*
- *Validation level 1a (field),*
- *Can be checked by a structural validation service based on a DSD available in a SDMX registry or by a content validation service linked to a DSD,*
- *Relevant for aggregated and micro-data,*
- *Severity level: usually Error, in some cases Warning*
- *Pre-conditions are possible.*

Parameters, Algorithm and VTL script (for level 1a):

Parameters	Algorithm	VTL
1. Field (F) 2. Link to Codes: - Case 1: Valid Code_List file (CL_F) - Case 2: Valid Codes (C1, ,...,Cn)	Check for each record, that Case 1: the content of Field (F) is in Valid Code_list file (CL_F) Case 2: the content of Field (F) is in Valid Codes (C1, ..., Cn)	This check can be implemented in a structural validation service or as a VTL script. //Case 1: From a code_list – proposed syntax (to be confirmed) check (Dataset-ID#**F** in **CL_F**) //Case 2: From a specified list of codes check (Dataset-ID#**F** in **["C1", "C2",** ...])

The code_List file can be expressed as a table with 2 columns

CL_F	
Codes	**Labels**
xx	aaaaaaaaaaaa
yy	bbbbbbbbbbbb

Example (Fa): Check that the content of field DIRECTION is in the valid code list CL_DIRECTION: IN (Departure from), OUT (Destination to)

Validation level	1a (field)
Severity	Error
Pre-condition	None

Parameters	Algorithm	VTL
1. Field (DIRECTION) - Case1 : Valid Code_List table (CL_DIRECTION)	Check for each record, that Case 1: the content of Field (DIRECTION) is in Valid Code_list (CL_DIRECTION).	check (INEUTRAV#F in CL_F)

CL_DIRECTION	
Codes	**Labels**
IN	INcoming (from partner)
OUT	OUTgoing (to partner)

Example of good and bad data:

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T02	Q	2016Q3	DE	ES	IN	TOTAL	N	20	
T02	Q	2016Q3	DE	ES	FROM	TOTAL	S	15	
T02	Q	2016Q3	DE	ES	OUT	TOTAL	N	21	P
T02	Q	2016Q3	DE	ES	OUT	TOTAL	S	16	P

Bad case: “FROM” is not in the list of acceptable codes for field “DIRECTION”.

4.2.5 (RWD) Records are Without Duplicate id-keys

Check that there are no duplicate id-keys between the different records (no duplicate combinations of the content of key fields).

Profile:

- *Mandatory for aggregated data, optional for micro-data (for some micro-data, identification keys of records are not obligatory and duplicate records can be accepted).*
- *Validation level 1c (file: check between records of the same file),*

- *Can be done by a structural validation service based on a DSD available in a SDMX registry or by a data loader or a content validation service linked to a DSD,*
- *Relevant for aggregated and potentially micro-data,*
- *Severity level: Error*
- *Pre-conditions are not possible.*

Parameters, Algorithm and VTL script (for level 1c):

Parameters	Algorithm	VTL
	Within the file, Check for each record, that the ID_key (merge of all key dimensions) is unique (has not already been used in a previous record)	Not relevant. This check is implemented in a structural validation service or data loader

Example (a): Check that the combination of the key fields TABLE, FREQ, TIME_PERIOD, REPORTING, PARTNER, DIRECTION, AGE, ADJUST is unique.

Validation level	1c (file)
Severity	Error
Pre-condition	None

Parameters	Algorithm	VTL
	Within the file, Check for each record, that the ID_key (*TABLE*, *FREQ*, *TIME_PERIOD*, *REPORTING*, *PARTNER*, *DIRECTION*, *AGE*, *ADJUST*) is unique (has not already been used in a previous record)	Not relevant. This check is implemented in a structural validation service or data loader but can be implemented also in VTL (additional check or using a logical key instead of structural key) KeyCount:= count(INEUTRAV) group by *TABLE*, *FREQ*, *TIME_PERIOD*, *REPORTING*, *PARTNER*, *DIRECTION*, *AGE*, *ADJUST*; check (KeyCount = 1)

Example of <u>good and bad data:</u>

Combination of the key fields TABLE, FREQ, TIME_PERIOD, REPORTING, PARTNER, DIRECTION, AGE, ADJUST is unique.

Key fields (dimensions)	Measure	Attribute

TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTIO N	AGE	ADJUST	OBS_VALUE	OBS_STATUS
T02	Q	2016Q3	DE	ES	IN	TOTAL	N	20	
T02	Q	2016Q3	DE	ES	IN	TOTAL	S	15	
T02	Q	2016Q3	DE	ES	OUT	TOTAL	N	21	P
T02	Q	2016Q3	DE	ES	OUT	TOTAL	N	25	P
T02	Q	2016Q3	DE	ES	OUT	TOTAL	S	16	P

Bad case: Duplicate key: for 2 records, same combination of the key fields TABLE, FREQ, TIME_PERIOD, REPORTING, PARTNER, DIRECTION, AGE, ADJUST.

4.3Check intra or inter files (levels 1 to 5)

Validation rules from this third group are usually performed after the rules related to the basic file structure and to basic intra-file check have been performed. The severity in case of failure of validation rules under this group is in most cases “error”, but it can also be “warning” or “information”. In this group, all types of validation rules are applicable to levels 1 (inside a file) and 2 (between files or datasets from the same domain and country). One type (Values are consistent) is also applicable to levels 3, 4 and 5, so may require access to external data.

Only 2 types of validation rules of this group may be described in SDMX: (COC) Codes are Consistent and (VAR) Values are in a Range. If sufficient information is available in a SDMX registry for these 2 types, they can then be checked automatically by a structural validation service.

The following 9 types of validation rules fall into this group:

- (REP) Records Expected are Provided,
- (RTS) Records are all present for Time Series,
- (RNR) Records’ Number is in a Range,
- (COC) Codes are Consistent,
- (VIR) Values are in Range,
- (VCO) Values are Consistent,
- (VAD) Values for Aggregates are consistent with Details,
- (VNO) Values are Not Outliers,
- (VSA) Values for Seasonally Adjusted data are plausible.

4.3.1 (REP) Records Expected are Provided

Check the coverage of the records provided: Expected codes or Periods for a specific field or combination of fields is present (check that no combination has been missed out).

Profile:

- *Mandatory for aggregated data, optional for micro-data (for some micro-data, identification keys of records are not obligatory and duplicate records can be accepted),*
- ***Validation level "1b" (between fields in the same record), "1c" (between records in the same file)*** and "2" (between different datasets from the same domain/country).
- *Level "1b" applies to coverages "Max" and "Excl". Level "1c" applies to coverages "Min" and "Only". Level "2" applies to cases of referential integrity between 2 datasets (the list of expected codes or values for one dataset may depend of what is provided in another dataset).*
- *Can be checked by a content validation service linked to a DSD,*
- *Relevant for aggregated and micro-data,*
- *Severity level: Error (or Warning in some cases),*
- *Pre-conditions are possible,*
- *Default: Min: The combinations of codes for relevant fields represents the minimum coverage of the records to be provided (More combinations are acceptable)*

Parameters, Algorithm and VTL script (for levels 1b and 1c):

Parameters	Algorithm	VTL
1. Fields with content checked (F1,..., Fn) 2. Combinations of Codes/values checked (CC_F1,..., CC_Fn) 3. Coverage: - **Min**: The combinations represent the minimum coverage of the records to be provided (More combinations are acceptable) - Max: The combinations represent the maximum coverage of the records to be provided (Less combinations are acceptable) - Only: The combinations are represented in all the records to be provided (not less, not more combinations can be accepted) - Excl: The combinations should not be provided in records ***=> Default: Min***	For each combination of key fields which are not in the list (F1, ..., Fn), If Coverage="Only" or "**Min**" then Check for all records within the file, that for fields (F1,..., Fn), each combination of expected codes in the list (CC_F1,..., CC_Fn) is represented in at least one record. If Coverage="Only" or "Max" then Check for each record, that the combinations of codes for Fields (F1, ...,Fn) are in the list (CC_F1,..., Fn) If Coverage="Excl" then Check for each record, that the combinations of codes for Fields (F1, ..., Fn) are not in the list (CC_F1,..., Fn)	//Based on a table with combination of Codes/values checked COMB (list of acceptable/non-acceptable codes identifiers) to be checked on a Dataset-ID //Case ="**Only**" or "**Min**" check(**exists_in** (COMB, Dataset-ID), errorcode("Combination should be provided "), errorlevel("Error")) //Case ="**Only**" or "**Max**" check(**exists_in** (Dataset-ID , COMB), errorcode("Combination should not be provided "), errorlevel("Error")) //Case ="**Excl**" check(**not exists_in** (DataTable , COMB), errorcode("Combination should not be provided "), errorlevel("Error"))

Example (H2): Check that for Table T02, both Seasonally adjusted (S) and non-adjusted (N) data are provided. i.e. check that for all the combination of TABLE, FREQ, TIME_PERIOD, REPORTING, PARTNER, DIRECTION, AGE, 2 records are provided: one for seasonally adjusted data (code "S" for field "ADJUST"), one for non-seasonally adjusted data (code "N" for field "ADJUST").

Validation level	1c (file)
Severity	Error
Pre-condition	applicable only to table (**T02**)

Parameters	Algorithm	VTL

1. Fields with expected content (ADJUST) 2. Combinations of codes/values expected (CC_ADJUST= "N", "S") 3. Coverage: - Only: The combinations are represented in all the records to be provided: not less, not more combinations can be accepted.	For each combination of key fields which are not in the list of field with expected content (TABLE, FREQ, TIME_PERIOD, REPORTING, PARTNER, DIRECTION, AGE, ~~ADJUST~~), check: If Coverage="Only" or "Min" then Check in all records within the file, that for field (ADJUST), the expected codes in the list (CC_ADJUST= "N", "S") are represented in at least one record. If Coverage="Only" or "Max" then Check for each record, that the codes for Field (ADJUST) are in the list (CC_ADJUST="N", "S")	* Test that both S and N Adjust for table T02 on the provided series (Warning: if both of "N" and "S" adjustment datapoints for a series are not provided, error is not detected) *\ CC_ADJUST_N := INEUTRAV [sub ADJUST = "N"] [filter TABLE = "T02"]; CC_ADJUST_S := INEUTRAV [sub ADJUST = "S"] [filter TABLE = "T02"]; check (**exists_in** (CC_ADJUST_N, CC_ADJUST_S) and **exists_in** (CC_ADJUST_S, CC_ADJUST_N), **errorcode**("Both Seasonally adjusted (S) and non-adjusted (N) series should be provided for table T02"), **errorlevel**("Error"))

Example of good and bad data:

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T02	Q	2016Q3	DE	ES	OUT	TOTAL	N	21	P
T02	Q	2016Q3	DE	ES	OUT	TOTAL	S	16	P
T02	Q	2016Q3	DE	ES	IN	TOTAL	N	20	

Good case: Both Seasonally adjusted (S) and non-adjusted (N) data are provided for travellers reported by DE as outgoing to ES
Bad case: Seasonally adjusted (S) data is missing in case of travellers reported by DE as incoming from ES.

4.3.2 (RTS) Records are all present for Time Series

Check that time series provided in one file are complete (between the oldest and the most recent time period expected in the file, no period is missing).

Profile:

- *Applies only to time series provided in separate records in the same file (where one record represents only one time period in a file),*
- *The latest (most recent) time period in the file is provided by the identification envelope of the file,*
- *Validation level "1c" (between records in the same file),*
- *Can be checked by a content validation service linked to a DSD (in SDMX, field TIME_PERIOD is used),*
- *Relevant only to aggregated data,*
- *Severity level: Error (or Warning in some cases),*
- *Pre-conditions are not possible,*

Parameters, Algorithm and VTL script (for level 1c)*:*

Parameters	Algorithm	VTL
1. First TIME_PERIOD expected (FTP) *Last TIME_PERIOD expected (LTP) is not a parameter to be set in the rule. The rule needs it but takes it from the envelope provided with the data file.*	Within the file, Check in all records that for each combination of key fields except TIME_PERIOD, all values of TIME_PERIOD between FTP and LTP (Last Time Period in the envelope) are represented. Check for each record that TIME_PERIOD >= FTP and <= LTP	* **Assumption**: The **fill_time_series** function only complete hole in a series but not the border of a provided interval. Here, we extend the function definition: (Warning: at least one value of the series must be sent) *\ **fill_time_series** (<Dataset>, <Periodicity=A/Q/M/...>, **<First Period>, <Last Period>**) (optional fields) EXTENDED_SERIES := **fill_time_series** (Dataset-ID, Periodicity, FTP, LTP) ; check (**exists_in** (EXTENDED_SERIES, Dataset-ID), **errorcode**("full time-series periods should be provided"), **errorlevel**("Error"))

Example (C2): Check that time series are complete between 2008 and the last time period provided in the envelope (the assumption below is that "2010" is the last time period mentioned in the envelope).

Validation level	1c (file)
Severity	Error
Pre-condition	None

Parameters	Algorithm	VTL
1. First TIME_PERIOD expected (FTP=2008)	Within the file, Check in all records that for each combination of key fields except TIME_PERIOD (TABLE, FREQ, ~~TIME_PERIOD~~, REPORTING, PARTNER, DIRECTION, AGE, ADJUST), that all values of TIME_PERIOD between 2008 and LTP (Last Time Period in the envelope) are represented. Check for each record that TIME_PERIOD >= 2008 and <= LTP	LTP := ENV(Year); EXTENDED_SERIES := **fill_time_series** (INEUTRAV, "A", 2008, LTP) ; check (**exists_in** (EXTENDED_SERIES, INEUTRAV), **errorcode**("full time-series periods should be provided"), **errorlevel**("Error"))

Example of good and bad data:

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T01	A	2008	FR	DE	IN	TOTAL	N	200	
T01	A	2009	FR	DE	IN	TOTAL	N	203	
T01	A	2010	FR	DE	IN	TOTAL	N	202	
T01	A	2008	FR	ES	IN	TOTAL	N	150	
T01	A	2010	FR	ES	IN	TOTAL	N	158	
T01	A	2011	FR	DE	OUT	TOTAL	N	210	

Good case: All TIME_PERIOD between "2008" and "2010" are provided for the specific combination of key fields (T01, A, …, FR, DE, IN, TOTAL, N),

Bad case 1: TIME_PERIOD "2009" is missing for the specific combination of key fields (T01, A, …, FR, ES, IN, TOTAL, N),

Bad case 2: TIME_PERIOD 2011 is higher than the Last Time Period (LTP).

4.3.3 (RNR) Records' Number is in a Range

Check that the number of records in a file is higher or equal to a minimum and (optionally) is lower or equal to a maximum.

Profile:

- *Mandatory (empty files are not accepted),*
- *Validation level "1c" (between records in the same file) and potentially "2" (between datasets from the same domain and country),*
- *Validation level "2" in case of referential integrity between 2 datasets (e.g. the number of records for one dataset may depend of what is provided in another dataset),*
- *Can be checked by a content validation service linked to a DSD,*
- *Relevant to aggregated and micro data,*
- *Severity level: Error (or Warning or Information in some cases),*
- *Pre-conditions are possible (e.g. on countries),*
- *Default: Min=1 => at least one record should be provided in the data files.*

Parameters, Algorithm and VTL script (for level 1c):

Parameters	Algorithm	VTL
1. Minimum number of records expected in one file: **Min** -> "**1**" is the default value *2. (optional) Maximum number of records expected in one file: Max*	Within the file, Check in all records that The total number of records >= **Min** *And (optionally) <= Max*	Case 1: only MIN provided RECORD_NUMBER := **count** (Dataset-ID) ; check (RECORD_NUMBER >= **Min**, **errorcode**("The number of records should be minimum: MIN"), **errorlevel**("Error")) Case 2: MIN and MAX provided RECORD_NUMBER := **count** (Dataset-ID) ; check (RECORD_NUMBER >= **Min** and RECORD_NUMBER <= **Max**, **errorcode**("The number of records should be between MIN and MAX"), **errorlevel**("Error"))

Example: Check that the minimum number of records is 4.

Validation level	1c (file)
Severity	Error
Pre-conditions	Applicable to Malta and Table T01

Parameters	Algorithm	VTL
1. Min=4: Minimum number of records expected in one file	Within the file, Check in all records that The total number of records >= Min=4	DATA := INEUTRAV [**filter** REPORTING = "MT" **and** TABLE = "T01"]; RECORD_NUMBER := **count** (DATA) ; check (RECORD_NUMBER >= **4** , **errorcode**("The number of records should be minimum: 4 "), **errorlevel**("Error"))

Example of bad data file: Malta only transmitted a file for Table T01 with 3 records (4 minimum are expected).

Key fields (dimensions)								Measure	Attribute
TABLE	FREQ	TIME_	REPORTING	PARTNER	DIRECTIO	AGE	ADJUST	OBS_VALUE	OBS_STATUS

		PERIOD			N				
T01	A	2008	MT	DE	IN	TOTAL	N	200	
T01	A	2009	MT	DE	IN	TOTAL	N	203	
T01	A	2010	MT	DE	IN	TOTAL	N	202	

4.3.4 (COC) Codes are Consistent

Check that the codes used in fields are consistent with other codes used in (level 1b) another field of the same record, (level 1c) the same field in different records of the same file or (level 2) in different datasets from the same country.

Profile:

- *Validation level "1a" (between field and element of the envelope), "1b" (between fields in the same record), "1c" (between records in the same file) and "2" (between different datasets from the same domain and country),*
- *Can be checked by a structural validation service for a DSD in a SDMX registry with constraints matrixes, if (a) the codes to be cross-checked depend of the dataset-id, (b) In case of failure, an error should be generated.*
- *Can be checked with a content validation service linked to a DSD,*
- *Relevant to aggregated and micro data,*
- *Severity level: Error (or Warning in some cases),*
- *Pre-conditions are possible.*

Parameters, Algorithm and VTL script (for levels 1a and 1b):

Parameters	Algorithm	VTL
1. Field to validate (FV) 2. Field to compare with (FC) or Element of the envelope: (ENV: "Dataset", "Periodicity", "Country", "Period") 3. Relational Operator (RO) between the codes of the 2 fields FV and FC (or ENV):	For each record, For Fields FV and (FC or ENV) If RO="=" then Check that code for (FV) = code for (FC or ENV) If RO="<>" then Check that code for (FV) <> code for (FC or ENV) If RO="match" then check in constraint matrix (CM_*)	//Direct implementation: check with FC using **RO (= or <>)** **check** (Dataset-ID# **FV RO** Dataset-ID# **FC**, **errorcode**("**FV** and **FC** should be equal/different"), **errorlevel**("Error")) //Direct implementation: check with **ENV** using **RO (= or <>)** **check** (Dataset-ID# **FV RO ENV(<Element>)**, **errorcode**("**FV** and **FC** should be equal/different"), **errorlevel**("Error")) //Match using datapoint Ruleset

- "=" the 2 codes should be identical, - "<>" the 2 codes should be different, - "match the 2 codes should match according to a constraint matrix, - "excl" the 2 codes are exclusive from each other (they are not compatible) according to a constraint matrix. 4. (for RO="match" or "excl") Constraints matrix between the 2 fields: CM_* (correspondence table with list of codes for the 2 fields)	that the code for (FV) matches with a related code in Field FC (or ENV) If RO="excl" then check in constraint matrix (CM_*) that the code for (FV) excludes (does not correspond to) a related code in Field (FC or ENV)	**define datapoint ruleset** *CM_match*(**FV**, **FC**) **is** **when FV** = "xx" **then FC** ="aa" **errorcode** ("Error Message 1") **errorlevel** ("Error"); **when FV** = "yy" **then FC** = "bb" **errorcode** ("Error Message 2") **errorlevel** ("Error"); **end datapoint ruleset ;** **check** (Dataset-ID, *CM_match*) //Remark: Syntax **check** (Dataset-ID, *CM_match*, *not valid*) is supposed to work VTL1.1 but not implemented yet //Exclusion using datapoint Ruleset **define datapoint ruleset** *CM_Excl*(**FV**, **FC**) **is** **when FV** = "xx" **then FC** <>"aa" **errorcode** ("Error Message 1") **errorlevel** ("Error"); **when FV** = "yy" **then FC** <> "bb" **errorcode** ("Error Message 2") **errorlevel** ("Error"); **end datapoint ruleset ;** **check** (Dataset-ID, *CM_Excl*) //Remark: no Envelope input inside a datapoint Ruleset, only direct implementation can used. //additional Table based implementation COMB_FV_FC := count(Dataset-ID) group by **FV**, **FC** ERROR := check (**exists_in** (COMB_FV_FC, **CM_***)) But ERROR only return the wrong (**FV**, **FC**) combinations, it must be join to Dataset-ID to return the full records.

The constraints matrix can be expressed as a table with 2 columns:

CM_*	
Codes for FV	**Codes for FC (field to compare)**

(Field to Validate)	or ENV (element of the envelope)
xx	aa
yy	bb

Example (B1): Check that in TABLE T01, for field FREQ, only code “A” (Annual) is provided; In TABLE T02, only code “Q” (Quarterly) is provided

Validation level	1b (record)
Severity	Error
Pre-conditions	None

Parameters	Algorithm	VTL
1. Field to Validate (FREQ) 2. Field to compare with (TABLE) 3. Relational Operator (RO) between the codes of the 2 fields: FREQ and TABLE: “match”: the 2 codes should match according to a constraint matrix, 4. (for RO=”match”) Constraints matrix between the 2 fields: CM_FREQ_TABLE (correspondence table with list of codes for the 2 fields)	For each record, for Fields FREQ and TABLE If RO=”match” then check in constraint matrix (CM_FREQ_TABLE) that the code for (FREQ) matches with a related code in Field (TABLE)	**define datapoint ruleset** *TABLE_Periodicity* (*TABLE*, *FREQ*) **is** **when** *TABLE* = "T01" **then** *FREQ*="A" **errorcode** ("*Table* T01 should contain only Annual series (FREQ=A)”) **errorlevel** ("Error"); **when** *TABLE* = "T02" **then** *FREQ*="Q" **errorcode** ("Table T02 should contain only Quarterly series (FREQ=Q)") **errorlevel** ("Error"); **end datapoint ruleset ;** **check** (INEUTRAV **[keep [***OBS_VALUE***]]**, *TABLE_Periodicity***)**

CM_FREQ_TABLE	
FREQ	**TABLE**
A	T01
Q	T02

Example of good and bad data:

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T01	Q	2016	FR	DE	IN	TOTAL	N	200	
T01	A	2016	FR	ES	IN	TOTAL	N	150	
T01	A	2016	FR	ES	OUT	TOTAL	N	160	

Base case: Code "Q" in FREQ is not allowed for TABLE T01

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T02	Q	2016Q3	FR	ES	IN	TOTAL	N	20	
T02	A	2016Q3	FR	ES	IN	TOTAL	S	15	
T02	A	2016Q3	FR	ES	OUT	TOTAL	N	21	P
T02	Q	2016Q3	FR	ES	OUT	TOTAL	S	16	P

Bad case: code "A" in FREQ is not allowed for TABLE T02.

Example (D1): Check that Reporting country in the data is the same as the reporting country in the envelope

Validation level	1b (record)
Severity	Error
Pre-conditions	None

Parameters	Algorithm	VTL
1. Field to validate (REPORTING) 2. with element of the envelope: ENV(Country)	For each record, for Field REPORTING and element of the envelope: ENV(Country) If RO="=" then	CURREPORTING := **ENV(Country);** check (INEUTRAV#REPORTING = CURREPORTING ,

3. Relational Operator (RO) between the codes of the 2 fields REPORTING and ENV(Country): "=" the 2 codes should be identical,	Check that code for (REPORTING) = code for (ENV(Country))	**errorcode**("Reporting country in the data should be the same as the reporting country in the envelope"), **errorlevel**("Error"))

Example of good and bad data:

-> Envelope: Dataset-ID= INEUTRAV_T01_A, Periodicity=A, **Country="EL",** Period=2016

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T01	A	2016	EL	ES	IN	TOTAL	N	150	
T01	A	2016	EL	ES	OUT	TOTAL	N	160	
T01	A	2016	GR	DE	OUT	TOTAL	N	210	

Bad case: Code in REPORTING (DK) is not consistent with the country in the envelope (EL)

Example (E1): Check that PARTNER country is different from REPORTING country (PARTNER <> REPORTING)

Validation level	1b (record)
Severity	Error
Pre-conditions	None

Parameters	Algorithm	VTL
1. Field to validate (PARTNER) 2. Field to compare with (REPORTING) 3. Relational Operator (RO) between the codes of the 2 fields PARTNER and REPORTING: "<>" the 2 codes should be	For each record, for Fields PARTNER and REPORTING If RO="<>" then Check that code for PARTNER <> code for REPORTING	**check** (INEUTRAV**#***PARTNER* **<>** INEUTRAV**#***REPORTING*, **errorcode** ("Reporting and Partner Countries should be different "), **errorlevel** ("Error")**)**

different,		

Example of good and bad data:

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T02	Q	2016Q3	FR	ES	IN	TOTAL	S	15	
T02	Q	2016Q3	FR	ES	OUT	TOTAL	N	21	P
T02	Q	2016Q3	FR	FR	IN	TOTAL	N	20	
T02	Q	2016Q3	FR	ES	OUT	TOTAL	S	16	P

Bad case: REPORTING and PARTNER fields are the same (FR) in the third record whereas they should differ

Example (H1): Check for table T01 (Annual), that field "ADJUST" is always set to "N" (Not seasonally adjusted)

Validation level	1b (record)
Severity	Error
Pre-conditions	None

Parameters	Algorithm	VTL
1. Field to validate (ADJUST) 2. Field to compare with (TABLE) 3. Relational Operator (RO) between the codes of the 2 fields ADJUST and TABLE: (match) the 2 codes should match according to a constraint matrix, 4. (for RO="match") Constraints matrix between the 2 fields: CM_ ADJUST_TABLE (correspondence table with list of codes for the 2 fields)	For each record, for Fields ADJUST and TABLE If RO="match" then check in constraint matrix (CM_ADJUST_TABLE) that the code for (ADJUST) matches with a related code in Field (TABLE)	**define datapoint ruleset** *TABLE_Adjustment* (*TABLE*, *ADJUST*) **is** **when** *TABLE* = "T01" **then** *ADJUST* ="N" **errorcode** ("T01 table should contain only Non-Adjusted series (field ADJUST="N")") **errorlevel** ("Error"); **end datapoint ruleset ;** **check** (INEUTRAV **[keep [***OBS_VALUE***]]**, *TABLE_ Adjustment***)**

CM_ADJUST_TABLE	
ADJUST	**TABLE**
N	T01

Example of good and bad data:

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTION**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T01	A	2016	FR	DE	IN	TOTAL	N	200	
T01	A	2016	FR	ES	IN	TOTAL	N	150	
T01	A	2016	FR	DE	OUT	TOTAL	S	210	
T01	A	2016	FR	ES	OUT	TOTAL	N	160	

Bad case: For table T01 (Annual), field ADJUST is S (Seasonally adjusted) in 1 case.

4.3.5 (VIR) Values are In a Range

Check that the observation value is higher (or equal) to a minimum value and/or is lower (or equal) to a maximum value.

Profile:

- *Validation level "1a" (in a field),*
- *Applies to "value fields". For the most common cases of SDMX compatible datasets, applies to field "OBS_VALUE" (Observation Value),*
- *Can be checked by a structural validation service for a DSD in a SDMX registry with constraints matrixes, if (a) the range has been specified in the DSD, (b) In case of failure, an error should be generated.*
- *Can be checked with a content validation service linked to a DSD,*
- *Relevant to aggregated and micro data,*
- *Severity level: Error (or Warning or Information in some cases),*
- *Defaults: Operator for "Min" value checked is by default ">=", for "Max" value checked is by default "<=",*
- *Pre-conditions are possible.*

Parameters, Algorithm and VTL script (for level 1a and field OBS_VALUE):

Parameters	Algorithm	VTL
1. Case1: Min 1a. Minimum value (Min) 1b. Operator (">=" or ">") **=> *Default: ">="*** 2. Case 2: Max 2a. Maximum value (Max) 2b. Operator ("<="or "<") **=> *Default: "<="*** 3. Case 3: range between Min -Max 2a. Minimum value (Min) 2b. Maximum value (Max)	For each record, for Field OBS_VALUE Case 1 (Min Value) OBS_VALUE "**>=**" or ">" Min Case2: (Max Value) OBS_VALUE "**<=**" or "<" Max Case3: (Min,Max Value) OBS_VALUE between Min and Max	**check** (Dataset-ID **[filter** not **isnull**(*OBS_VALUE*)**]** >= **MIN**, **errorcode** ("Values, should be higher or equal to **MIN** "), **errorlevel** ("Warning")) **check** (Dataset-ID **[filter** not **isnull**(*OBS_VALUE*)**]** <= **MAX**, **errorcode** ("Values, should be lower or equal to **MAX** "), **errorlevel** ("Warning")) **check** (between(Dataset-ID**, MIN, MAX)**, **errorcode** ("Values, should be lower or equal to **MAX** "), **errorlevel** ("Warning"))

2c. Operator (between) **=> *Default:* "<="** *Both cases (Min and Max) can apply to an observation value.*		

Example (I1): Check that value in field OBS_VALUE is equal or higher to 1

Validation level	1a (field)
Severity	Warning
Pre-conditions	None

Parameters	Algorithm	VTL
1a. Minimum value (Min=1) 1b. Operator **(>=)**	For each record, for Field OBS_VALUE Case1: (MIN Value) OBS_VALUE**>=1**	**check** (INEUTRAV **[filter** not **isnull**(*OBS_VALUE*)**]** >= 1, **errorcode** ("Values, when provided, should be higher or equal to 1"), **errorlevel** ("Warning")**)**

Example of good and bad data:

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T02	Q	2016Q3	FR	ES	IN	TOTAL	N	0	
T02	Q	2016Q3	FR	ES	IN	TOTAL	S	-15	
T02	Q	2016Q3	FR	ES	OUT	TOTAL	N	21	P
T02	Q	2016Q3	FR	ES	OUT	TOTAL	S	16	P

Bad cases: 2 values in field OBS_VALUE are lower than 1

4.3.6 (VCO) Values are Consistent

Check that values are consistent between fields, records or files.

Profile:

- *Validation levels "1b" (between fields in the same record),* ***"1c" (between records in the same file)****, "2" (between different datasets from the same country), 3 (between different countries for the same domain), 4 (between different domains within the ESS), 5 (between data sources from different organisations which do not share the same methodology),*
- *Applies to "value fields". For the most common cases of SDMX compatible datasets, applies to field "OBS_VALUE" (Observation Value),*
- *To be checked with a content validation service linked to a DSD,*
- *Relevant to aggregated and micro data,*
- *Severity level: Error (or Warning or Information in some cases),*
- *Pre-conditions are possible,*
- *A tolerance (acceptable margin) expressed in % or absolute number is possible.*

Parameters, Algorithm and VTL script (for level 1c and field OBS_VALUE):

Parameters	Algorithm	VTL
1. Key field (KF) 2. Code used for validation: "CV" 3. Code to compare with "CC" 4. Relational Operator: RO (=, <>, >=, >, <=, <,...) 5. (optional) Mathematical expression on "CC" 6. (optional) Tolerance (n) *Can be expressed in % or absolute value*	Within the file, Check in all records that for each combination of key fields except field KF, OBS_VALUE in record with code in KF="CV" is linked with OBS_VALUE in record with code in KF="CC" by relational operator RO (i.e. (KF="CV") RO (KF="CC")) (optionally) with a mathematical expression on "CC" (optionally)+/- Tolerance(n)	VALUE_KF_CV:= Dataset-ID **[sub KF="CV"];** // Case 1: Direct CC code VALUE_KF_CC:= Dataset-ID **[sub KF="CC"];** // Case 2: calculated CC code VALUE_KF_CC:= sum(Dataset-ID [filter **KF in** ["C1", "C2", ...]) along **KF;** **check** (VALUE_KF_CV **RO** VALUE_KF_CC, **errorcode**("Error message"), **errorlevel**("Error")) //Remark: Tolerance is difficult to specify without a clear formula, but it can be shown like this: **check** ((VALUE_KF_CV **MO** VALUE_KF_CC) <= **n**, **errorcode**("Error message"), **errorlevel**("Error")) //Where MO stand for any mathematical expression: abs(VALUE_KF_CV - VALUE_KF_CC) for the simplest one.

Example (G2): Check that the children that travel (Y0_18) represent less than half of the TOTAL travellers (OBS_VALUE for AGE code= "Y0_18", represents less than half of the OBS_VALUE for AGE code = TOTAL).

Validation level	1c (between records in the same file)
Severity	Warning
Pre-conditions	None

Parameters	Algorithm	VTL
1. Key field (AGE) 2. Code used for validation CV="Y0_18" 3. Code to compare with CC="TOTAL" 4. Relational Operator: RO ="<" 5. Mathematical expression ="TOTAL/2"	Within the file Check in all records that for each combination of key fields except field AGE, OBS_VALUE in record with code in AGE="Y0_18" is linked with OBS_VALUE in record with code in AGE="TOTAL" by relational operator RO="<" and with a mathematical expression on CV(Y0_18)=CC(TOTAL)/2 (i.e. OBS_VALUE (AGE="Y0_18") < (AGE="TOTAL" / 2))	**ds_All_Ages**:= INEUTRAV **[sub** *AGE* = "TOTAL" **]** ; **ds_Under_18**:= INEUTRAV **[sub** *AGE* = "Y0_18" **];** **check** (**ds_All_Ages** > 2 * **ds_Under_18**, **errorcode** ("Children aged 18 or under should represent less than half of total travellers "), **errorlevel** ("Warning")**)**

Example of good and bad data:

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T01	A	2016	FR	ES	IN	TOTAL	N	200	
T01	A	2016	FR	ES	IN	Y0_18	N	120	
T01	A	2016	FR	ES	OUT	TOTAL	N	100	
T01	A	2016	FR	ES	OUT	Y0_18	N	45	

Bad case: In 2016, children reported by France as incoming from Spain represented 60% (120/200) of the total travellers incoming from Spain (>=50%).

4.3.7 (VAD) Values for Aggregates are consistent with Details

Check that values for aggregates are consistent with the sum of values for detailed data.

Profile:

- *Mandatory if both aggregates and details are provided,*
- *Validation levels* ***"1c" (between records in the same file)*** *and "2" (between different datasets from the same country),*
- *Applies to "value fields". For the most common cases of SDMX compatible datasets, applies to field "OBS_VALUE" (Observation Value),*
- *To be checked with a content validation service linked to a DSD,*
- *Relevant only to aggregated data,*
- *Severity level: Error (or Warning or Information in some cases),*
- *Pre-conditions are possible,*
- *Default: aggregate = sum of details*
- *A tolerance (acceptable margin) expressed in % or absolute number is possible.*

Parameters, Algorithm and VTL script (for level 1c and field OBS_VALUE):

Parameters	Algorithm	VTL
1. Field (F) 2. Hierarchical table (HT_*) with aggregated and detailed codes (parents and children codes) 3. Relational Operator between aggregates and the sum of detailed data: RO (=, <>, >=, >, <=, <,...) **-> *Default: "="*** 4. (optional) Tolerance (n) *Can be expressed in % or absolute value*	Within the file, Check in all records that OBS_VALUE in Records with Aggregate code (with code of Field (F) in aggregate_code of table HT_*) are linked with the sum of OBS_VALUE in related records with detailed codes code (with code of Field (F) in Related_detailed_code of table HT_*) by relational operator RO (i.e. aggregate_code.OBS_VALUE RO (∑Related_detailed_code.OBS_VALUE)) (optionally)+/- Tolerance(n)	//Direct Implementation (single test) TOTAL:= Dataset-ID [sub F="xx"]; DETAIL:= sum(Dataset-ID [filter F **in** ["C1", "C2", ...]) along F; **check** (TOTAL = DETAIL, **errorcode**("Error message"), **errorlevel**("Error")) or **check** (abs(TOTAL – DETAIL) / TOTAL < n, **errorcode**("Error message"), **errorlevel**("Error")) //Hierarchical Ruleset (can be used among different datasets) **define hierarchical ruleset** *HR_F* (**variable=** *F*) **is** /* R1 */ xx = aa+bb; /* R2 */ yy = cc+dd+ee; /* R3 */ zz = ff + gg; **end hierarchical ruleset** **check** (Dataset-ID, *HR_F*) //For tolerance add: //Case flat tolerance n **check** (Dataset-ID, *HR_F*) [**filter** abs(IMBALANCE) >= n] //Case % tolerance n **check** (Dataset-ID, *HR_F*) [**filter** abs(IMBALANCE/OBS_VALUE) >= n]

The Hierarchical table can be expressed as a table with 2 columns:

HT_*	
Aggregate_code (parents)	**Related_detailed_codes (children)**
xx	aa
xx	bb
yy	cc

yy	dd
yy	ee
zz	ff
zz	gg

Example (G1): Check that the number of travellers for the aggregated AGE groups corresponds to the sum of the detailed AGE groups with +/-1% tolerance

Validation level	1c (between records in the same file)
Severity	Warning
Pre-conditions	None

Parameters	Algorithm	VTL
1. Field (AGE) 2. Hierarchical table (HT_AGE_GROUPS) with aggregates and detailed codes (parents and children codes) 3. Relational Operator between aggregates and the sum of detailed data: RO **-> Default: "="** 3. Tolerance (1%) *Can be expressed in % or absolute value*	Within the file, Check in all records that OBS_VALUE in Records with Aggregate code (with code of AGE in AGE_GROUP_AGGREGATE of table HT_AGE_GROUP) are linked with the sum of OBS_VALUE in related records with detailed codes code (with code of AGE in AGE_GROUP_DETAILS of table HT_AGE_GROUP) by relational operator RO: "=". (i.e. AGE_GROUP_AGGREGATE.OBS_VALUE = (∑AGE_GROUP_DETAILS.OBS_VALUE)) +/- Tolerance(1%)	//Direct Implementation (single test) **TOTAL**:= INEUTRAV **[sub** *AGE* = "TOTAL" **]** ; **DETAIL**:= **sum**(INEUTRAV **[filter** *AGE* **in [**"Y0", "Y1", "Y2", … , "Y122", "UNK"**]]**) **along** *AGE***;** **check** (**abs**(**TOTAL - DETAIL**) / **TOTAL** < 0.01, **errorcode** ("Total aggregate does not match the sum of detailed age information "), **errorlevel** ("Warning")**)** //Hierarchical Ruleset (can be used among different datasets) **define hierarchical ruleset** *HR_* **AGE_GROUP_AGGREGATE** (**variable= AGE**) **is** /* R1 */ TOTAL = Y0_18+ Y19_64+ Y65_MAX+ UNK; /* R2 */ Y0_18= Y0+…+ Y18; /* R3 */ Y19_64= Y19+…+ Y64; /* R4 */ Y65_MAX = Y65+…+ Y122; **end hierarchical ruleset;** **check** (INEUTRAV, *HR_* **AGE_GROUP_AGGREGATE**)) [**filter** abs(IMBALANCE/OBS_VALUE) >= 0.01]

The Hierarchical table can be expressed as a table with 2 columns:

HT_AGE_GROUP	
AGE_GROUP_AGGREGATE	**AGE_GROUP_DETAILS**
TOTAL	Y0_18
TOTAL	Y19_64
TOTAL	Y65_MAX
Y0_18	Y0
…	…
Y0_18	Y18
Y19_64	Y19
…	…
Y19_64	Y64
Y65_MAX	Y65
…	…
Y65_MAX	Y122

Example of good and bad data:

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T01	A	2016	FR	IT	IN	TOTAL	N	201	
T01	A	2016	FR	IT	IN	Y0_18	N	100	
T01	A	2016	FR	IT	IN	Y19_64	N	50	
T01	A	2016	FR	IT	IN	Y65_MA X	N	50	
T01	A	2016	FR	ES	IN	TOTAL	N	204	
T01	A	2016	FR	ES	IN	Y0_18	N	100	
T01	A	2016	FR	ES	IN	Y19_64	N	50	
T01	A	2016	FR	ES	IN	Y65_MA X	N	50	

Bad case: In 2016, the Total travellers reported by France as incoming from Spain corresponds to the sum of the travellers for the 3 related age classes + 2% (204=((100+50+50)*1.02)). This is beyond the acceptable tolerance of 1% difference.

4.3.8 (VNO) Values are Not Outliers in Time Series

For data with seasonality or without seasonality, check that values in time series are not outliers.

Several algorithms are available by default to check for outliers. They calculate first an estimated value and then they compare it with the observation value to be checked. If the observation value is beyond the acceptable distance (tolerance) to the estimated value, then a failure is reported.

Profile:

- *Validation levels* ***"1c" (between records in the same file)*** *and "2" (between different datasets from the same country),*
- *Applies to "value fields". For the most common cases of SDMX compatible datasets, applies to field "OBS_VALUE" (Observation Value),*
- *To be checked with a content validation service linked to a DSD,*
- *Relevant only to aggregated data,*
- *Severity level: Warning (or Error or Information in some cases). Up to 3 levels of "tolerance" can be defined with corresponding 3 severity levels ("error" for the highest tolerance, "warning" for the middle tolerance, "information" for the lowest tolerance),*
- *Pre-conditions are possible,*
- *Default: check only the most recent period (M) in the file with Method "f" (Median and average growth rate) used on 5 periods and with a tolerance expressed as the MAD (Median Absolute Deviation)*2,*
- *A tolerance (acceptable margin) is needed. It can be expressed in %, in absolute number or as a multiplier of the Median Absolute Deviation (MAD) based on (x) previous periods.*

Parameters, Algorithm and VTL script (for level 1c and field OBS_VALUE):

Parameters	Algorithm	VTL
The methods proposed to detect outliers in time series may need to be reviewed. 1. Data With Seasonality - DWS (Y/**N**) ***-> Default: DWS="N"*** 2. Period(s) Checked – (PC): - "**M**" Most recent - "A" All periods ***-> Default: (M) Most recent period***	Within the file Check in all records that Case 1: DWS="**N**" data without seasonality: Case 1.1: PC="**M**" Most recent period checked only OBS_VALUE for most recent period = expected OBS_VALUE estimated from x previous periods according to Method (a) to	//This Rule definition is still a Work in Progress: FILE is the source File // WARNING: Moving aggregate used below are only calculated when the X number of periods are available, Only PC="M" rules are implemented here as well as Tolerance case (a) // Define Seasonality // DWS="Y" – VTL Trick: append an extra identifier PERIOD extracting the period component of SOURCE:= FILE [**identifier** PERIOD := **substr**(TIME_PERIOD, 5, 3)]; // DWS="N"

3. Method for calculation of expected value and tolerance (a) Value previous (1) period +/- tolerance (n in %) (b) Average value of (x) periods +/- tolerance (n in %) (c) Median value of (x) periods +/- tolerance Median Absolute Deviation (MAD) * (n) ***-> Default: (d) Median value on 5 previous periods with tolerance (MAD)*2***	(f) +/- Tolerance Case 1.2: PC="A" all periods checked OBS_VALUE = expected OBS_VALUE estimated from x nearest periods according to Method (a) to (c) +/- Tolerance Case 2: DWS="Y" data with seasonality: Case 2.1: PC="**M**" Most recent period checked only OBS_VALUE for most recent period = expected OBS_VALUE estimated from the same period in the x previous years according to Method (a) to (c) +/- Tolerance Case 2.2: PC=A" all periods checked OBS_VALUE = expected OBS_VALUE estimated from the same period in the x nearest years according to Method (a) to (c) +/- Tolerance	SOURCE:= FILE ; // Define Comparison Value //Case Method a/d COMPARE:=**lag**(SOURCE, 1) **over** (**order by** time_period); //Case Method b/e COMPARE:= **avg**(SOURCE) **over** (**order by** time_period rows between X preceding and 1 preceding); //Case Method c/f COMPARE:= **median**(SOURCE) **over** (**order by** time_period rows between X preceding and 1 preceding); // Define Deflator Value //Case Method a/b/c DEFLATOR:=1; //Case Method d/e/f – DEFLATOR Dataset can also exist from another source DATA:= SOURCE [**calc** PREV:= **lag**(SOURCE, 1) **over** (**order by** time_period)]; GROWTHRATE:= DATA [**calc** GR:= (PREV-OBS_VALUE)/PREV]; DEFLATOR:= avg(GROWTHRATE#GR) over (order by time_period rows between X preceding and 1 preceding); //Check Rule: //case (b) % Tolerance **check** (abs(SOURCE - (COMPARE * DEFLATOR)) / SOURCE <= **n**, **errorcode** ("Values should not be Outliers in Time Series "), **errorlevel** ("Warning")); //case (c) Flat Tolerance **check** (abs(SOURCE - (COMPARE * DEFLATOR)) <= **n**, **errorcode** ("Values should not be Outliers in Time Series "), **errorlevel** ("Warning"));

Example (C3.1): Check in table T01 (Annual data) that the number of travellers in the most recent year do not differ from more than 2*MAD (2 times the Median Absolute Diff) compared to the median of the previous 5 years.

Validation level	1c (between records in the same file)
Severity	Warning
Pre-conditions	T01 (Table T01 – Annual data)

Parameters	Algorithm	VTL (part in pink to be reviewed)
1. Data With Seasonality – DWS="N" 2. Period(s) checked: PC="M" Most recent 3. Method for calculation of expected value: (c) Median value 4. Number of Periods (or years) taken into account in the formula (5) 5. Tolerance (n) – acceptable distance (2) (a) Median Absolute Deviation (MAD) * (2) based on (5) previous periods	Within the file Check in all records that Case 1: DWS="N" data without seasonality: Case 1.1: PC="M" Most recent period checked only OBS_VALUE for most recent period = expected OBS_VALUE estimated from **5** previous periods according to Method (c) – Median Value +/- Tolerance (MAD*2)	DATA:= INEUTRAV [**filter** TABLE="T01" **and** FREQ="A"] COMPARE:= **median**(DATA) **over** (**order by** TIME_PERIOD rows between 5 preceding and 1 preceding); **check** (abs(DATA - COMPARE) / DATA <= **0.5**, **errorcode** ("Values should not be Outliers in Time Series "), **errorlevel** ("Warning"))

Example of good and bad data:

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T01	A	2011	FR	IT	IN	TOTAL	N	220	
T01	A	2012	FR	IT	IN	TOTAL	N	160	
T01	A	2013	FR	IT	IN	TOTAL	N	240	
T01	A	2014	FR	IT	IN	TOTAL	N	200	
T01	A	2015	FR	IT	IN	TOTAL	N	190	
T01	A	2016	FR	IT	IN	TOTAL	N	230	
T01	A	2011	FR	ES	IN	TOTAL	N	220	
T01	A	2012	FR	ES	IN	TOTAL	N	160	
T01	A	2013	FR	ES	IN	TOTAL	N	240	
T01	A	2014	FR	ES	IN	TOTAL	N	200	

T01	A	2015	FR	ES	IN	TOTAL	N	190	
T01	A	2016	FR	ES	IN	TOTAL	N	250	

Good case: For travellers reported by France as incoming from Italy, the number of travellers in the last year (2016) is 230 and the median for the 5 periods before 2016 is 200 (160, 190, 200, 220, 240). 230 represents an increase of 30 compared to the median (200) which is below the tolerance of 40=2*MAD (MAD=20 considering absolute differences of: 0, 10, 20, 40, 40) .

Bad case: For travellers reported by France as incoming from Spain, the number of travellers in the last year (2016) is 250 and the median for the 5 periods before 2016 is 200 (160, 190, 200, 220, 240). 250 represents an increase of 50 compared to the median (200) which is above the tolerance of 40=2*MAD (MAD=20 considering absolute differences of: 0, 10, 20, 40, 40) ..

Example (C3.2): Check in table T02 (Quarterly data) that the number of travellers in the most recent year do not differ from more than 2*MAD (2 times the Median Absolute Diff) compared to the median of the previous 5 years.

Validation level	1c (between records in the same file)
Severity	Warning
Pre-conditions	T02 (Table T02 – Quarterly data)

Parameters	Algorithm	VTL
1. Data With Seasonality – DWS="**Y**" 2. Period(s) Checked - PC: "**M**" Most recent 3. Method for calculation of expected value: (c) Median value 4. Number of Periods (or years) taken into account in the formula (**5**) 5. Tolerance (n) – acceptable distance (**2**) (a) Median Absolute Deviation (**MAD**) * (**2**) based on (**5**) previous periods	Within the file Check in all records that Case 2: DWS="**Y**" data with seasonality: Case 2.1: PC="**M**" Most recent period checked only OBS_VALUE for most recent period = expected OBS_VALUE estimated from the same period in the **5** previous years according to Method (c) – Median Value +/- Tolerance (**MAD*2**)	

Example of good and bad data:

Key fields (dimensions)	Measure	Attribute

TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTIO N	AGE	ADJUST	OBS_VALUE	OBS_STATUS
T02	Q	2011Q1	FR	IT	IN	TOTAL	N	60	
T02	Q	2011Q2	FR	IT	IN	TOTAL	N	55	
T02	Q	2011Q3	FR	IT	IN	TOTAL	N	45	
T02	Q	2011Q4	FR	IT	IN	TOTAL	N	60	
T02	Q	2012Q1	FR	IT	IN	TOTAL	N	35	
…									
T02	Q	2013Q1	FR	IT	IN	TOTAL	N	60	
…									
T02	Q	2014Q1	FR	IT	IN	TOTAL	N	45	
…									
T02	Q	2015Q1	FR	IT	IN	TOTAL	N	40	
…									
T02	Q	2016Q1	FR	IT	IN	TOTAL	N	60	
T02	Q	2011Q1	FR	ES	IN	TOTAL	N	60	
T02	Q	2011Q2	FR	ES	IN	TOTAL	N	55	
T02	Q	2011Q3	FR	ES	IN	TOTAL	N	45	
T02	Q	2011Q4	FR	ES	IN	TOTAL	N	60	
T02	Q	2012Q1	FR	ES	IN	TOTAL	N	35	
…									
T02	Q	2013Q1	FR	ES	IN	TOTAL	N	60	
…									
T02	Q	2014Q1	FR	ES	IN	TOTAL	N	45	
…									
T02	Q	2015Q1	FR	ES	IN	TOTAL	N	40	
…									
T02	Q	2016Q1	FR	ES	IN	TOTAL	N	80	

Good case: For travellers reported by France as incoming from Italy, the number of travellers in the most recent period (2016-Q1) is 60 and the median for Q1 of the 5 years before 2016 is 45 (35, 40, 45, 60, 60). 60 represents an increase of 15 compared to the median (45) which is below the tolerance of 20=2*MAD (MAD=20 considering absolute differences of: 0, 5, 10, 15, 15) .

Bad case: For travellers reported by France as incoming from Spain, the number of travellers in the most recent period (2016-Q1) is 80 and the median for Q1

of the 5 years before 2016 is 45 (35, 40, 45, 60, 60). 80 represents an increase of 35 compared to the median (45) which is above the tolerance of 20=2*MAD (MAD=20 considering absolute differences of: 0, 5, 10, 15, 15) .

4.3.9 (VSA) Values for Seasonally Adjusted data are plausible

Check that the totals annual of the time series that are seasonally adjusted is consistent with the corresponding totals of the series that are not seasonally adjusted.

Profile:

- *Validation levels* ***"1c" (between records in the same file)*** *and "2" (between different datasets from the same country),*
- *Applies to "value fields". For the most common cases of SDMX compatible datasets, applies to field "OBS_VALUE" (Observation Value),*
- *To be checked with a content validation service linked to a DSD,*
- *Relevant only to aggregated data,*
- *Severity level: Error (or Warning or Information in some cases),*
- *Pre-conditions are possible,*
- *A tolerance (acceptable margin) expressed in % or absolute number is possible.*

Parameters, Algorithm and VTL script (for level 1c and field OBS_VALUE):

Parameters	Algorithm	VTL
1. Field with code for seasonal adjustment (FSA) 2. Code for seasonally adjusted data: "SA" 3. Code for not-seasonally adjusted data: "NSA" 4. (optional) Tolerance (n) *Can be expressed in % or absolute value*	Within the file, Check in all records that For Record with code of Field FSA = "SA" ∑ OBS_VALUE of all periods in the same year = ∑ OBS_VALUE for all corresponding records with code of Field FSA = "NSA" *(optionally)+/- Tolerance(n)*	//Calculation from periodicity P to Annual CALCULATED:= **sum**(Dataset-ID **[filter** *FREQ* = " P " **]**) **time_aggregate** ("P ", "A"); //Get adjusted and Non-Adjusted series CALC_**S**:= CALCULATED **[sub FSA** = " **SA** " **] [filter** *OBS_VALUE* <> 0 **]**; CALC_**N**:= CALCULATED **[sub FSA** = "**NSA**" **]**; **check** (**abs**(CALC_**S** - CALC_**N**) / CALC_**S** < **n**, **errorcode** ("Annual sum of both P Adjusted and P Non-Adjusted series should match within a **n** % tolerance "), **errorlevel** ("Error")**)**

Example H3: Check in T02 (Quarterly data) that for each year, the sum of the 4 quarters for seasonally adjusted data ("S") do not differ from more than 1% compared to non-seasonally adjusted data '("N").

Validation level	1c (between records in the same file)
Severity	Warning
Pre-conditions	Table T02 (quarterly data)

Parameters	Algorithm	VTL
1. Field with code for seasonal adjustment (ADJUST) 2. Code for seasonally adjusted data SA="S" 3. Code for not-seasonally adjusted data NSA="N" 4. Tolerance (1%)	Within the file, Check in all records that For Record with code of Field ADJUST = "S" ∑ OBS_VALUE of all periods in the same year = ∑ OBS_VALUE for all corresponding records with code of ADJUST = "N" +/- Tolerance (1%)	//Sum quarterly series **ds_QuartSum**:= **sum**(INEUTRAV **[filter** *FREQ* = "Q" **]**) **time_aggregate** ("Q", "A"); //Get adjusted and Non-Adjusted series **ds_Adjust** := **ds_QuartSum [sub** *ADJUST* = "S" **] [filter** *OBS_VALUE* <> 0 **]**; **ds_NonAdj**:= **ds_QuartSum [sub** *ADJUST* = "N" **]**; **check** (**abs**(**ds_Adjust** - **ds_NonAdj**) / **ds_Ajust** < 0.01, **errorcode** ("Annual sum of both Quarterly Adjusted and Quarterly Non-Adjusted series should match within a 1 % tolerance "),

		errorlevel ("Warning")**)**

Example of good and bad data:

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T02	Q	2016Q1	FR	ES	OUT	TOTAL	S	30	
T02	Q	2016Q2	FR	ES	OUT	TOTAL	S	31	
T02	Q	2016Q3	FR	ES	OUT	TOTAL	S	32	
T02	Q	2016Q4	FR	ES	OUT	TOTAL	S	33	
T02	Q	2016Q1	FR	ES	OUT	TOTAL	N	19	
T02	Q	2016Q2	FR	ES	OUT	TOTAL	N	30	
T02	Q	2016Q3	FR	ES	OUT	TOTAL	N	55	
T02	Q	2016Q4	FR	ES	OUT	TOTAL	N	23	
T02	Q	2016Q1	FR	ES	IN	TOTAL	S	30	
T02	Q	2016Q2	FR	ES	IN	TOTAL	S	31	
T02	Q	2016Q3	FR	ES	IN	TOTAL	S	32	
T02	Q	2016Q4	FR	ES	IN	TOTAL	S	33	
T02	Q	2016Q1	FR	ES	IN	TOTAL	N	19	
T02	Q	2016Q2	FR	ES	IN	TOTAL	N	30	
T02	Q	2016Q3	FR	ES	IN	TOTAL	N	55	
T02	Q	2016Q4	FR	ES	IN	TOTAL	N	33	

Good case: For travellers reported by France as outgoing to Spain, for seasonally adjusted data (ADJUST=”S”), the sum of OBS_VALUE for the 4 quarters of 2016 is 126 (30+31+32+33). For not-seasonally adjusted data (ADJUST=”N”), the sum of OBS_VALUE for the 4 quarters of 2016 is 127 (19+30+55+23). The difference is 0.8% which is within the tolerance of +/-1%.

Bad case: For travellers reported by France as incoming from Spain, for seasonally adjusted data (ADJUST=”S”), the sum of OBS_VALUE for the 4 quarters of 2016 is 126 (30+31+32+33). For not-seasonally adjusted data (ADJUST=”N”), the sum of OBS_VALUE for the 4 quarters of 2016 is 137 (19+30+55+33). The difference is 8.7% which is beyond the tolerance of +/-1%.

4.4Check inter-files in the same statistical domain (Level 2 and 3)

Validation rules from this fourth group are usually performed after the validation rules from the first 3 groups have been checked. The severity in case of failure of validation rules under this group is usually warning, but it can also be "error" or "information". In this group, the 2 types of validation rules related to revised data are applicable to level 2 (between files from the same domain and country) and the type of validation rules related to mirror data is applicable to level 3. All require access to data in an internal database related to the domain.

All validation rules depending from this group need more than what is in SDMX registries. All of them require then a content validations service to be checked.

The following 3 types of validation rules fall into this group:

- (RRL) Records Revised are Limited in number or %
- (VRT) Values are Revised within a Tolerance level
- (VMP) Values for Mirror data are Plausible

4.4.1 (RRL) Records Revised are Limited in number or %

Check that the number or percentage of records revised does not exceed a maximum.

Records are considered as revised when their observation value (OBS_VALUE) has been updated compared to the previous version of data transmitted to Eurostat (records with the same id-key have different observation values in the new version and in the previous version). Records added (with new-id keys compared to data in the previous version) or deleted (with id-keys missing in the incoming data file compared to data in the previous version) can also be counted as revisions. The update of an attribute cannot be considered as a revision.
This type of validation rule is more relevant for the case of "full revision of data" where the new version of the file replaces fully data sent in the previous version (for "time series": the new version contains data for the full time series else the new version contains all data for a specific period).

Profile:

- ***Validation level** "2" (between files from the same dataset, period, domain and country),*
- *Applies to "value fields". For the most common cases of SDMX compatible datasets, applies to field "OBS_VALUE" (Observation Value),*
- *To be checked with a content validation service linked to a DSD,*
- *Relevant to aggregated and micro data (with id-keys),*
- *Severity level: Warning (or Error or Information in some cases),*
- *Pre-conditions are possible,*
- *Default: check that the percentage of records revised (including updated, deleted and added records) does not exceed a maximum of 10%.*

Parameters, Algorithm and VTL script (for level 2):

Parameters	Algorithm	VTL
1. Maximum records revised: - Case 1: MaxAbs: Maximum absolute number of records revised - Case2: MaxPC: Maximum percentage of records revised in the file ***-> Default: 1b. MaxPC=10%*** 2. Added Records counted as revised - AR (Y/N) ***-> Default: AR="Y"*** 3. Deleted Records counted as revised - DR (Y/N) ***-> Default: DR="Y"***	Between incoming file and previous version of the file for the same period (TIME_PERIOD) or time series, Within the incoming file, for each record, Count RevisedUpd= Number of records with an ID-key in the incoming file which was already in the previous version but with a different observation value (OBS_VALUE), If AR="Y", then Count RevisedAdd= Number of records with a new ID-key in the incoming file compared to the previous version If DR="Y", then Count RevisedDel= Number of records in the previous version with no corresponding ID-Key in the new version. Calculate Revised= RevisedUpd + RevisedAdd + RevisedDel Case 1 (MaxAbs): Check that Revised <= MaxAbs Case 2 (MaxPC): Check that percentage of Revised / (Total number of records) <= MaxPC	// using DATASET_REV dataset containing previous version of the sent data with a *REVISION* **identifier** **FLP :=** DATASET_REV [sub REVISION = "X"] [rename OBS_VALUE to OLD _VALUE] [keep [OLD _VALUE]]; **FLN :=** Dataset-ID [rename OBS_VALUE to NEW_VALUE] [keep [NEW_VALUE]]; ALLREC := outer_join (**FLN, FLP)** ; REVISEDREC := count(ALLREC [filter NEW_VALUE<> OLD_VALUE]); TOTALREC := count(ALLREC); **check** (REVISEDREC <= **n**, **errorcode** ("Number of record revised should not exceed **n**"), **errorlevel** ("Warning")) **check** (abs(REVISEDREC- TOTALREC)/ TOTALREC <= **n**, **errorcode** ("Number of record revised should not exceed **n %**"), **errorlevel** ("Warning"))

Example: check that the percentage of records revised (including updated, deleted and added records) does not exceed a maximum of 10%.

Validation level	2 (*between files from the same dataset, period, domain and country*)
Severity	Warning
Pre-conditions	None

Parameters	Algorithm	VTL
1. Maximum records revised: ***-> Default: 1b. MaxPC=10%*** 2. Added Records counted as revised - AR (**Y**/N) ***-> Default: AR="Y"*** 3. Deleted Records counted as revised - DR (**Y**/N) ***-> Default: DR="Y"***	Between incoming file and previous version of the file for the same period (TIME_PERIOD) or time series, Within the incoming file, for each record, Count RevisedUpd= Number of records with an ID-key in the incoming file which was already in the previous version but with a different observation value (OBS_VALUE), If AR="Y", then Count RevisedAdd= Number of records with a new ID-key in the incoming file compared to the previous version If DR="Y", then Count RevisedDel= Number of records in the previous version with no corresponding ID-Key in the new version. Calculate Revised= RevisedUpd + RevisedAdd + RevisedDel Case 2 (MaxPC): Check that percentage of Revised / (Total number of records) <= 10%	**FLP :=** INEUTRAV _REV [sub REVISION = "X"]] [rename OBS_VALUE to OLD _VALUE] [keep [OLD _VALUE]]; **FLN :=** INEUTRAV [rename OBS_VALUE to NEW_VALUE] [keep [NEW_VALUE]]; ALLREC := outer_join (**FLN**, **FLP**); REVISEDREC := count(ALLREC [filter NEW_VALUE<> OLD_VALUE]); TOTALREC := count(ALLREC); **check** (abs(REVISEDREC- TOTALREC)/ TOTALREC <= **0.1**, **errorcode** ("Number of record revised should not exceeds **10 %**"), **errorlevel** ("Warning ")**)**

Example of bad version 2 :

Version 1 :

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T01	A	2008	MT	BE	IN	TOTAL	N	11	
T01	A	2008	MT	BG	IN	TOTAL	N	7	
T01	A	2008	MT	CZ	IN	TOTAL	N	10	
T01	A	2008	MT	DK	IN	TOTAL	N	5	
T01	A	2008	MT	DE	IN	TOTAL	N	82	
T01	A	2008	MT	EE	IN	TOTAL	N	1	
T01	A	2008	MT	IE	IN	TOTAL	N	4	
T01	A	2008	MT	EL	IN	TOTAL	N	11	
T01	A	2008	MT	ES	IN	TOTAL	N	46	
T01	A	2008	MT	FR	IN	TOTAL	N	64	
T01	A	2008	MT	IT	IN	TOTAL	N	58	
T01	A	2008	MT	CY	IN	TOTAL	N	1	

Version 2 :

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T01	A	2008	MT	BE	IN	TOTAL	N	11	
T01	A	2008	MT	BG	IN	TOTAL	N	7	
T01	A	2008	MT	CZ	IN	TOTAL	N	10	
T01	A	2008	MT	DK	IN	TOTAL	N	5	
T01	A	2008	MT	DE	IN	TOTAL	N	82	
T01	A	2008	MT	EE	IN	TOTAL	N	1	
T01	A	2008	MT	IE	IN	TOTAL	N	4	

T01	A	2008	MT	EL	IN	TOTAL	N	10	
T01	A	2008	MT	ES	IN	TOTAL	N	46	
T01	A	2008	MT	IT	IN	TOTAL	N	58	
T01	A	2008	MT	CY	IN	TOTAL	N	1	
T01	A	2008	MT	LV	IN	TOTAL	N	2	

Bad case: In version 2, compared to version 1, 3 revisions could be noticed: (1) the number of incoming travellers from EL (Greece) was updated from 11 to 10, (2) the incoming travellers from FR (France) were deleted, (3) incoming travellers from LV (Latvia) were added. In total, the percentage of revised records is then 25% (3 out of 12 records) which is beyond the maximum of 10%

4.4.2 (VRT) Values are Revised within a Tolerance level

Check that the revisions of the observation values do not exceed a maximum distance (tolerance) compared to the values provided in the previous version. *Deleted and added records can potentially be considered. A deleted record is then considered as a revision with an observation value (OBS_VALUE) which is "zero" to be compared with the observation value provided in the previous version.*

Profile:

- ***Validation level** "2" (between files from the same dataset, period, domain and country),*
- *Applies to "value fields". For the most common cases of SDMX compatible datasets, applies to field "OBS_VALUE" (Observation Value),*
- *To be checked with a content validation service linked to a DSD,*
- *Relevant to aggregated and micro data (with id-keys),*
- *Severity level: Warning (or Error or Information in some cases),*
- *Pre-conditions are possible,*
- *Default: Check, for updated records only, that the revised observation values do not differ from more than 10% compared to the values provided in the previous version. Deleted and Added records are not considered as default because the comparison implies a "virtual zero" +/- tolerance in % (that remains a zero). So this always generates a failure except if the other OBS_VALUE to be compared with is also zero (unusual case).*

Parameters, Algorithm and VTL script (for level 2):

Parameters	Algorithm	VTL
1. Tolerance compared to value in previous version (**n**) *Tolerance can be expressed in % or absolute value* -**> Default: 10%** 2. Added Records counted as revised - AR (Y/**N**) ***-> Default: AR="N"*** 3. Deleted Records counted as revised - DR (Y/**N**) ***-> Default: DR="N"***	Between incoming file and previous version of the file for the same period (TIME_PERIOD) or time series, Within the incoming file, for each record, If ID-key in the incoming file was already in the previous version but with a different observation value (OBS_VALUE), then check that revised OBS_VALUE corresponds to previous version of OBS_VALUE +/- Tolerance If ID-key in the incoming file was not in the previous version and AR="Y", then check that revised OBS_VALUE corresponds to zero +/- Tolerance (a virtual record with OBS_VALUE=0 is considered for the previous version) If DR="Y", then For records in the previous version with no corresponding ID-Key in the new version, then check that zero (virtual record with OBS_VALUE=0 for revised version) corresponds to previous version of OBS_VALUE +/- Tolerance (n).	// using DATASET_REV dataset containing previous version of the sent data with a *REVISION* **identifier** **FLP := ** DATASET_REV [sub REVISION = "X"]; **FLN := ** Dataset-ID; **check** (abs(**FLN - FLP**) / **FLN** <= **n**, **errorcode** ("Data should not differs more than **n** % from revision X "), **errorlevel** ("Error")) **FLP2 := FLP** [rename OBS_VALUE to OLD _VALUE] [keep [OLD _VALUE]]; **FLN2 := FLN** [rename OBS_VALUE to NEW_VALUE] [keep [NEW_VALUE]]; ADDDELREC := outer_join (**FLN2**, **FLP2**); /* IF AR="Y" then */ ADDEDREC := ADDDELREC [**filter isnull**(OLD_VALUE)]; **check** (abs(ADDEDREC#NEW_VALUE) <= **n**, **errorcode** ("Data should not differs more than **n** as new record "), **errorlevel** ("Error")) /* IF DR="Y" then */ DELREC := ADDDELREC [**filter isnull**(NEW_VALUE)]; **check** (abs(DELREC#OLD_VALUE) <= **n**, **errorcode** ("Data should not differs more than **n** as removed record "), **errorlevel** ("Error"))

Example (I2): Check that the number of travellers do not differ from more than 20% compared to previous version.

Validation level	2 (*between files from the same dataset, period, domain and country*)

Severity	Warning
Pre-conditions	None

Parameters	Algorithm	VTL
1. Tolerance compared to value in previous version: (20%) 2. Added Records counted for tolerance - AR (Y/**N**) 3. Deleted Records counted for tolerance - DR (Y/**N**)	Between incoming file and previous version of the file for the same period (TIME_PERIOD) or time series, Within the incoming file, for each record, If ID-key in the incoming file was already in the previous version but with a different observation value (OBS_VALUE), then check that revised OBS_VALUE corresponds to previous version of OBS_VALUE +/- 20% Tolerance.	// using INEUTRAV_REV containing previous version of the sent data with a *REVISION* **identifier** PREVIOUSDATA := INEUTRAV_REV **[sub** *REVISION* = "1" **];** CURRENTDATA := INEUTRAV**;** **check** (abs(CURRENTDATA **-** PREVIOUSDATA**)** / CURRENTDATA <= 0.2 , **errorcode** ("Data should not differs more than 20% from revision 1"), **errorlevel** ("Warning")**)**

Example of good and bad data in version 2:

Version 1 :

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T01	A	2016	FR	ES	IN	TOTAL	N	204	
T01	A	2016	FR	ES	IN	Y0_18	N	100	
T01	A	2016	FR	ES	IN	Y19_64	N	52	
T01	A	2016	FR	ES	IN	Y65_MA X	N	52	
T01	A	2016	FR	IT	IN	TOTAL	N	250	

Version 2 :

Key fields (dimensions)								Measure	Attribute	Diff Version2 / Version1
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTIO N	AGE	ADJUST	OBS_VALUE	OBS_STATUS	
T01	A	2016	FR	ES	IN	TOTAL	N	200		-1.96%
T01	A	2016	FR	ES	IN	Y0_18	N	100		0%
T01	A	2016	FR	ES	IN	Y19_64	N	70		+34.62%
T01	A	2016	FR	ES	IN	Y65_MA X	N	30		-42.31%
T01	A	2016	FR	DE	IN	TOTAL	N	300		NA

Bad cases: two revised observation values (in version 2) are beyond the limit of +/- 20% differences with corresponding values of version 1. The added record for PARTNER=DE is not considered for the check. The deleted record for PARTNER=IT is not considered as well for the check.

4.4.3 (VMP) Values for Mirror data are Plausible

Check the consistency of observation values for opposite flows (incoming and outgoing flows) between countries.

Check that flows reported by country A as incoming from country B are consistent with flows reported by country B as outgoing to country A (check that the values of both flows do not differ from more than an acceptable tolerance/distance).

Profile:

- ***Validation level** "3" (between data from the same dataset but coming from different countries),*
- *Applies to "value fields". For the most common cases of SDMX compatible datasets, applies to field "OBS_VALUE" (Observation Value),*
- *To be checked with a content validation service linked to a DSD,*
- *Relevant to aggregated data only,*
- *Severity level: Warning (or Error or Information in some cases),*
- *Pre-conditions are possible,*
- *Default: Check that mirror observation values (for opposite flows) do not differ from more than 10%.*

Parameters, Algorithm and VTL script (for level 3):

It is assumed that data from reporting country are in a file to be validated. These data are compared for mirror check with mirror data from partner countries stored in a database. Both datasets, in file and in database have in their data structure the same ID-Key with at least 5 common fields: (1) TIME_PERIOD, (2) reporting country/region, (3) partner country/region, (4) Direction of the flow, (5) observation value (OBS_VALUE).

Filed "Direction of the flow" may contain:

- *one code for INcoming flow (imports, arrivals, …) from partner to reporting country,*
- *or one code for OUTgoing flow (exports, departures, …) from reporting to partner country.*

Parameters	Algorithm	VTL
1. Field for reporting country/region (FRC) 2. Field for partner country/region (FPC) 3. Field for Direction of the Flow (FDF) 4. Code for Direction of the Flow 4a. incoming flow: INFLOW 4b. outgoing flow: OUTFLOW 5. (optional) Tolerance (**n**) *Can be expressed in % or absolute value* ***-> Default: 10%***	Between incoming file from reporting country "X" and database with data from partner countries "Y", For each record of incoming file, If Direction FDF=INFLOW then Check in file that OBS_VALUE in record for field reporting country FRC="X" and field partner country FPC="Y" corresponds to OBS_VALUE in database for field reporting country FRC="Y" and field partner country FPC="X" for Direction FDF = OUTFLOW +/– tolerance(n). If Direction FDF=OUTFLOW then Check in file that OBS_VALUE in record for field reporting country FRC="X" and field partner country FPC="Y" corresponds to OBS_VALUE in database for field reporting country FRC="Y" and field partner country FPC="X" for Direction FDF = INFLOW +/– tolerance(n). *Special case 1: No mirror data at all in database for the period and partner country – no mirror check is made with this partner country as data from partner country is considered as not yet received for that period.*	//This Rule definition is still a Work in Progresss // Check Traveller from Reporting Country with mirror **FLA_IN**:= **FLA [sub FDF** = "IN" **]** ; **FLA_OUT**:= **FLA [sub FDF** = " OUT " **]** ; **FLB_IN**:= **FLB [sub FDF** = " IN " **] [rename FRC to** tmp, **FPC to FRC**, tmp **to FPC**]; **FLB_OUT**:= **FLB [sub FDF** = "OUT" **] [rename FRC to** tmp, **FPC to FRC**, tmp **to FPC**]; //Special case 1: **check** (abs(**FLA_IN - FLB_OUT)** / **FLA_IN** <= **n**, **errorcode** ("Incoming travelers reported differ more than **n** % from Outgoing declared by partner country "), **errorlevel** ("Warning")) **check** (abs(**FLA_ OUT - FLB_ IN)** / **FLA_ OUT** <= **n**, **errorcode** ("Outgoing travelers reported differ more than **n** % from Incoming declared by partner country "), **errorlevel** ("Warning")) //Initialisation for Special cases 2 and 3: **FLA_IN2** := **FLA_IN** [rename OBS_VALUE to SRC _VALUE] [keep [SRC _VALUE]]; **FLA_OUT2** := **FLA_ OUT** [rename OBS_VALUE to SRC _VALUE] [keep [SRC _VALUE]]; **FLB_IN2** := **FLB_IN** [rename OBS_VALUE to MIR _VALUE] [keep [MIR _VALUE]]; **FLB_OUT2** := **FLB_ OUT** [rename OBS_VALUE to MIR _VALUE] [keep [MIR _VALUE]];

	Special case 2: Data is available in the database for the period and partner country, but a mirror record in the database (with same ID-Key except opposite flow) is missing. In this case, a "virtual mirror record" in the database with OBS_VALUE=zero is generated for the mirror check with the file. *Special case 3: Data is available in the database for the period and partner country, but a mirror record in the file (with same ID-Key except opposite flow) is missing. In this case, a "virtual mirror record" in the file with OBS_VALUE=zero is generated for the mirror check with the database.*	**FL_IN_EXT**:= outer_join (**FLA_IN2, FLB_OUT2) ;** **FL_OUT_EXT**:= outer_join (**FLA_OUT2, FLB_IN2)** ; //Special case 2: **check** (not isnull(**FL_IN_EXT#**SRC_VALUE) and isnull(**FL_IN_EXT#**MIR_VALUE) , **errorcode** ("Incoming travelers reported should exist in the database"), **errorlevel** ("Warning")) **check** (not isnull(**FL_ OUT _EXT#**SRC_VALUE) and isnull(**FL_ OUT _EXT#**MIR_VALUE) , **errorcode** ("Outgoing travelers reported should exist in the database"), **errorlevel** ("Warning")) //Special case 3: **check** (isnull(**FL_IN_EXT#**SRC_VALUE) and not isnull(**FL_IN_EXT#**MIR_VALUE) , **errorcode** ("Incoming travelers reported should be provided"), **errorlevel** ("Warning")) **check** (isnull(**FL_ OUT _EXT#**SRC_VALUE) and not isnull(**FL_ OUT _EXT#**MIR_VALUE) , **errorcode** ("Outgoing travelers reported should be provided"), **errorlevel** ("Warning"))

Example (F1): Check that travellers reported by country X as Incoming from Country Y correspond to travellers reported by country Y as outgoing to country X with a tolerance of 10%.

Validation level	3 (*between data from the same dataset but coming from different countries*)
Severity	Warning
Pre-conditions	None

Parameters	Algorithm	VTL
1. Field for reporting country/region -FRC="REPORTING" 2. Field for partner country/region – FPC="PARTNER"	Between incoming file from reporting country "X" and database with data from partner countries "Y", For each record of incoming file, If Direction FDF="IN" then Check in file that OBS_VALUE in record for field	// Check Traveller Incoming from Reporting Country with mirror **ds_Reporter_Incoming**:= *IntraEUTravellers* **[sub** *DIRECTION* = "IN" **]** ; **ds_Partner_Outgoing**:= *IntraEUTravellers* **[sub** *DIRECTION* = "OUT" **]**

3. Field for Direction of the Flow – FDF=”DIRECTION” 4. Code for Direction of the Flow 4a. incoming flow – INFLOW=”IN” 4b. outgoing flow – OUTFLOW=”OUT” 5. Tolerance: **10%**	REPORTING=“X” and field PARTNER=“Y” corresponds to OBS_VALUE in database for field REPORTING=“Y” and PARTNER=“X” for Direction FDF = “OUT” **+/– 10%**. in record for reporting country FRC=“REPORTING” and partner country FPC=“PARTNER” corresponds to OBS_VALUE in database for reporting country FRC=“REPORTING” and partner country FPC=“” for Direction FDF = “OUT” **+/– 10%**. If Direction FDF=”OUT” then Check in file that OBS_VALUE in REPORTING= “X” and PARTNER “Y” = OBS_VALUE in database for REPORTING “Y” and PARTNER “X” for Direction FDF = “IN” **+/– 10%.** Check in file that OBS_VALUE in record for field REPORTING=“X” and field PARTNER=“Y” corresponds to OBS_VALUE in database for field reporting country REPORTING=“Y” and field PARTNER=“X” for Direction FDF = INFLOW +/– tolerance(n). *<u>Special case 1:</u> No mirror data at all in database for the period and partner country – no mirror check is made with this partner country as data from partner country is considered as not yet received for that period.* *<u>Special case 2:</u> Data is available in the database for the period and partner country, but a mirror record <u>in the database</u> (with same ID-Key except opposite flow) is missing. In this case, a “virtual mirror record” <u>in the database</u> with OBS_VALUE=zero is generated for the mirror check with the file.*	**[rename** *REPORTING* **to** tmp, *PARTNER* **to** *REPORTING*, tmp **to** *PARTNER*]; **check** (abs(**ds_Reporter_Incoming - ds_Partner_Outgoing)** / **ds_Reporter_Incoming** <= 0.1 , **errorcode** ("Incoming travellers reported should not differ more than 10% from Outgoing declared by partner country "), **errorlevel** ("Warning")) // Check Passenger Outgoing from Reporting Country with mirror **ds_Reporter_Outgoing**:= *IntraEUTravellers* **[sub** *DIRECTION* = "OUT" **]** ; **ds_Partner_Incoming**:= *IntraEUTravellers* **[sub** *DIRECTION* = "IN" **]** **[rename** *REPORTING* **to** tmp, *PARTNER* **to** *REPORTING*, tmp **to** *PARTNER*]; **check** (abs(**ds_Reporter_Outgoing - ds_Partner_Incoming)** / **ds_Reporter_Outgoing** <= 0.1 , **errorcode** ("Outgoing travellers reported should not differ more than 10% from Incoming declared by partner country "), **errorlevel** ("Warning"))

	Special case 3: Data is available in the database for the period and partner country, but a mirror record in the file (with same ID-Key except opposite flow) is missing. In this case, a "virtual mirror record" in the file with OBS_VALUE=zero is generated for the mirror check with the database.	

Example of good and bad data in a file prepared for Eurostat:

1. File prepared for Eurostat: Data reported by France with partner Spain

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T01	A	2016	FR	ES	IN	TOTAL	N	150	
T01	A	2016	FR	ES	OUT	TOTAL	N	230	
T01	A	2016	FR	ES	IN	Y19_6 4	N	100	

2. Mirror data available in database as reported by Spain with partner France

Key fields (dimensions)								Measure	Attribute
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTIO N**	**AGE**	**ADJUST**	**OBS_VALUE**	**OBS_STATUS**
T01	A	2016	ES	FR	OUT	TOTAL	N	157	
T01	A	2016	ES	FR	IN	TOTAL	N	200	
T01	A	2016	ES	FR	OUT	Y65_MA X	N	50	

Recapitulative table with mirror data reported by FR and ES for Table="T01", FREQ="A", TIME_PERIOD="2016", ADJUST="N"

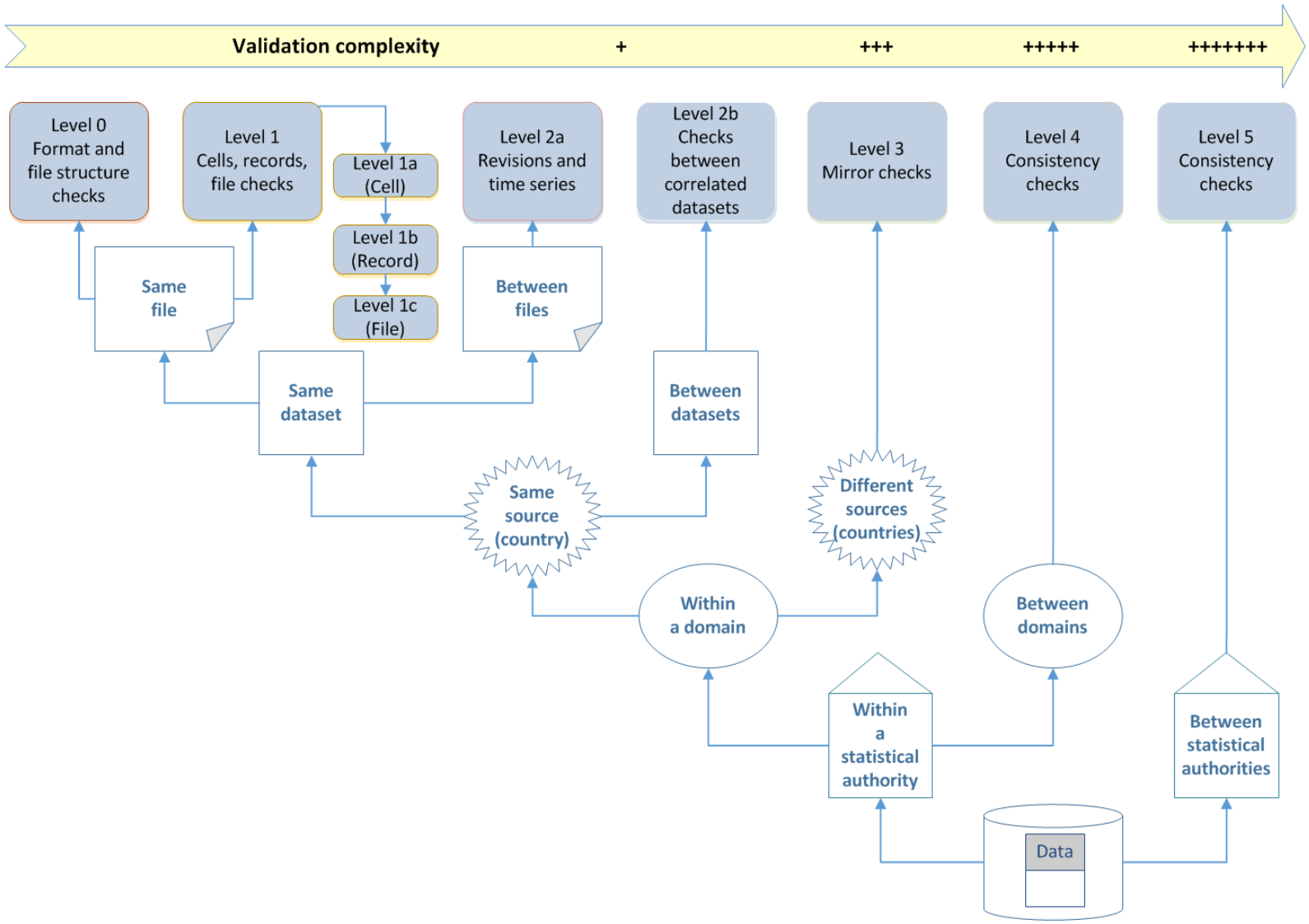
Reported by France (in file)					Mirror as reported by Spain (in Database)					Mirror Diff IN (FR<-ES) OUT (FR->ES)
Selected Key fields				Measure	Selected Key fields				Measure	
REPORTING	PARTNER	DIRECTION	AGE	OBS_VALUE	REPORTING	PARTNER	DIRECTION	AGE	OBS_VALUE	
FR	ES	IN	TOTAL	150	ES	FR	OUT	TOTAL	157	-4.5%
FR	ES	OUT	TOTAL	230	ES	FR	IN	TOTAL	200	+15%
FR	ES	IN	Y19_64	100						infinite
					ES	FR	OUT	Y65_MAX	50	-100%

3 Bad cases: (1) FR reported 15% more travellers outgoing to ES than ES reported incoming travellers from FR (15%=(230-200)/200) which is beyond the tolerance of 10%

(2) FR reported 100 incoming travellers from ES with AGE= « Y19_60 » whereas ES did not report outgoing travellers to FR for this AGE group

(3) ES reported 50 outgoing travellers to FR with AGE= « Y65_MAX » whereas FR did not report incoming travellers from ES for this AGE group

Annex-I Validation levels (from zero to 5)



Extract of the ESS handbook on “Methodology for data validation 1.1 – revised edition November 2017 – chapter 4”

4 Validation levels from a business perspective

(Sarah Giessing, Katrin Walsdorfer)

As it was observed in the introduction, data validation is not a new process, and some common elements can be derived from the existing practice of data validation. To this aim, a survey on the data validation procedures currently used in the NSIs has been carried out in the ESSnet ValiDat Foundation to find out how the typical validation process is practically implemented in the Member States of the ESS, see *Giessing and Walsdorfer, 2015* for information on design and results of this survey.

Looking at the practical implementation of the validation process means to take a business perspective. In the business perspective the attention is focused on the validation activities. The amount of information needed and the phases of the validation process are important for determining the validation levels. This approach is particularly useful for classifying and designing validation activities within an organization.

It is generally assumed that there are basically two general categories:

A. Technical integrity of the file, i.e., consistency with the expected IT structural requirements (Structural Validation)
B. Logical and statistical consistency of the data (Content Validation)

The second category is generally split into different sub-categories (levels) involving more and more information. The two general categories can then be expanded forming the following validation levels from a business perspective.

Validation level B0: consistency with the expected IT structural requirements
Validation level B1: consistency within the data set
Validation level B2: consistency with other data sets within the same domain and within the same data source
Validation level B3: consistency within the same domain between different data sources
Validation level B4: consistency between separate domains in the same data provider
Validation level B5: consistency with data of other data providers

It could be noticed that validation level B0 corresponds to the general category A earlier mentioned.

Validation level B0: consistency with the expected IT structural requirements

At this level, it is checked the consistency of the data with their expected IT requirements, for instance

- if the file has been sent/prepared by the authorised authority (data sender);
- if the column separator, the end of record symbol are correctly used;
- if the file has the expected number of columns (agreed format of the file);
- if the column have the expected format of the data (i.e., alphanumeric, numeric, etc.)

For these quality check only the structure of the file or the format of the variables are necessary as input.

Validation level B1: consistency within the data set

It is checked the consistency within the elements of the data set. For these quality check, it is needed only the (statistical) information included in the file itself.

For instance:

- check whether the number included in column 4 is not negative (as expected);
- check whether the year in the second column is 2011, as in the file name;
- check whether the content of the third column is one of the codes of the dictionary "Sex";
- check whether the content of the first column is consistent with the data sender (let's assume that there is a dictionary including the list of the data senders associated to the specific data set): data for Luxembourg should not be sent by another country.
- based on information available before data collection (for example from previous survey or other sources) one could establish a "plausibility range" for a certain variable (for instance number of components of a household).
- check consistency at (micro-level) of two (or more) variables: a certain combination of codes is illogical, a variable has to be reported only for a certain combination of codes.
- check consistency at macro-level of two (or more) variables: Total inhabitants = male inhabitants + female inhabitants, or Female inhabitants = (total inhabitants / 2) +/- 10%

Validation level B2: consistency with other data sets within the same domain and within the same data source

Validation levels B2 is concerned with the check of consistency based on the comparison of the content of the file with the content of "other files" referring to the same statistical system (or domain) and the same data source.

For instance:

Case a) the "other files" can be other versions of exactly the same file.

In this case the quality check are meant to detect "revisions" compared to previously sent data. Detection and analysis of revisions can be useful for example to verify if revisions are consistent with outliers detected in previous quality check (corrections) or to have an estimate of the impact of the revisions in the "to be published" results, for the benefit of the users.

Case b) the "other files" can be versions of the same data set referring to other time periods.

These check are usually referred to as "time series check" and are meant to verify the plausibility of the time series.

Case c) the "other files" can refer to other data sets from the same data provider (e.g., Countries in the ESS), referring to the same or other correlated time periods. Sometimes a group of data sets (same country, same reference period) is sent at the same time.

Example: three files could be sent at the same time, from the same country and referring to the same time period: one file includes data for "females", one for "male" and one for "total". Consistency between the results of the three files can be checked.

Another example: results from annual data sets can be compared with the results of the corresponding quarterly data sets.

Validation level B3: consistency within the same domain between different data sources

Validation levels B3 is concerned with the check of consistency based on the comparison of the content of the file with the content of "other files" referring to a different data provider on the same harmonised statistical system or domain (sharing common standards with respect to scope, definitions, units and classifications in the different surveys and sources).

For instance:

Case d) the "other files" can refer to the same data set, but from another data provider (e.g., Countries of the ESS, sharing harmonized methodologies).

Mirror check are included in this class. “Mirror statistics involve coherence, geographical comparability as well as accuracy issues.” (Eurostat, 2014). Often such statistics is important for data analysis on Eurostat level. Mirror check verify the consistency between declarations from different sources referring to the same phenomenon, e.g., export declared by country A to country B should be the same as import declared by country B from country A.

Validation level B4: consistency between separate domains in the same data provider

Validation level B4 could be defined as plausibility or consistency check between separate domains available in the same Institution. The availability implies a certain level of "control" over the methodologies by the concerned Institution.

These check could be based on the plausibility of results describing the "same" phenomenon from different statistical domains. Examples: unemployment from registers and from Labour Force Survey, or inhabitation of a dwelling (from survey of owners of houses and dwellings vs. from population register)

Check could also be made between results from correlated micro-data and macro-data sources.

Other plausibility check could be based on known correlations between different phenomena: for example external trade and international transport activity in ports.

Validation level B5: consistency with data of other data providers

Validation level B5 could be defined as plausibility or consistency check between the data available in the data provider (Institution) and the data / information available outside the data provider (Institution). This implies no "control" over the methodology on the basis of which the external data are collected, and sometimes a limited knowledge of it.

Statistical indicators collected by Eurostat might also be compiled for their own needs by national institutions such as National Statistical Institutes or Ministries; by private entities (ports, airports, companies, etc.) and also by international organisations (World Bank, United Nations, International Monetary Fund, etc.).

For example, EU road freight statistics are prepared by Member States according to the EU Commission legal acts and in addition countries can carry out specific surveys for national purposes. A benchmarking between indicators common to these different surveys allows assessing the coherence of these data and could help improving the methodologies for data collection.

To summarize, the classification of validation levels presented above implicitly assumes a growing degree of complexity from one level to another. However, this must not necessarily be reflected by a growing technical complexity of the validation check themselves. From the technical point of view, the distinction made with respect to data sets is an artifice, since data sets and files could be merged into single databases in advance of implementing the check.

A likely rise in complexity might be regarding organizational and management matters. On the higher levels of this classification more parties and stake-holders will be involved, potentially with different needs and requirements regarding data validity. This certainly tends to make it more difficult to harmonize technical and methodological concepts. However, this may depend very much on the concrete situation and circumstances.

Annex-II - Severity levels of validation rules and accept data business function.

Extract of document “Business Architecture for ESS Validation” – March 2017 – Chapter 6.4

6.4 Accept data

Description

The *Validate data* business function has as its main output a validation report outlining which rules are not satisfied by the data. On the basis of this validation report, Eurostat must judge whether the data can be accepted or not. If the data can be accepted depends on the severity level of the rules breached. The severity level of each rule is defined during the design of the rules themselves (see section 6.2) and can have three possible values:

- Error: errors are rules that must necessarily be satisfied for mathematical or logical reasons. Data that do not satisfy these rules will not be considered acceptable, except in exceptional circumstances (see section 6.4 for more details).
- Warning: warnings are rules that, when not satisfied, highlight suspicious values. Data that do not satisfy these rules are not automatically rejected. The data provider must however supply a justification/comment. The data are accepted if Eurostat deems the justification to be sufficient.
- Information: rules whose severity level is marked as "Information" highlight potentially suspicious values. Contrary to the case of warnings, when data do not satisfy these rules, a justification/comment by the data provider is not required for them to be accepted.

In exceptional cases, the data provider or data receiver may agree upon a one-time override of the severity of a given rule. This may be necessary in case the severity level of a given rule is considered to be too lax or too strict due to special circumstances. There are two possible kinds of overrides:

- Member States may request to treat an "error" as a "warning". This would mean that data that do not satisfy the rule in question can be accepted, provided that a suitable justification/comment is given. Eurostat must agree to this override request.
- Eurostat may request to treat an "information" as a "warning". This would mean that a suitable justification/comment is required in order to accept data that do not satisfy the rule in question. The concerned Member State(s) must agree to this override request.

It should be stressed that these overrides should only be used when extraordinary or unusual circumstances justify it. The occurrence of regular override requests for a given rule suggests that the default severity level for the rule in question should be changed. Working Groups should therefore take into account the statistics on the number of override requests when reviewing the severity of validation rules.

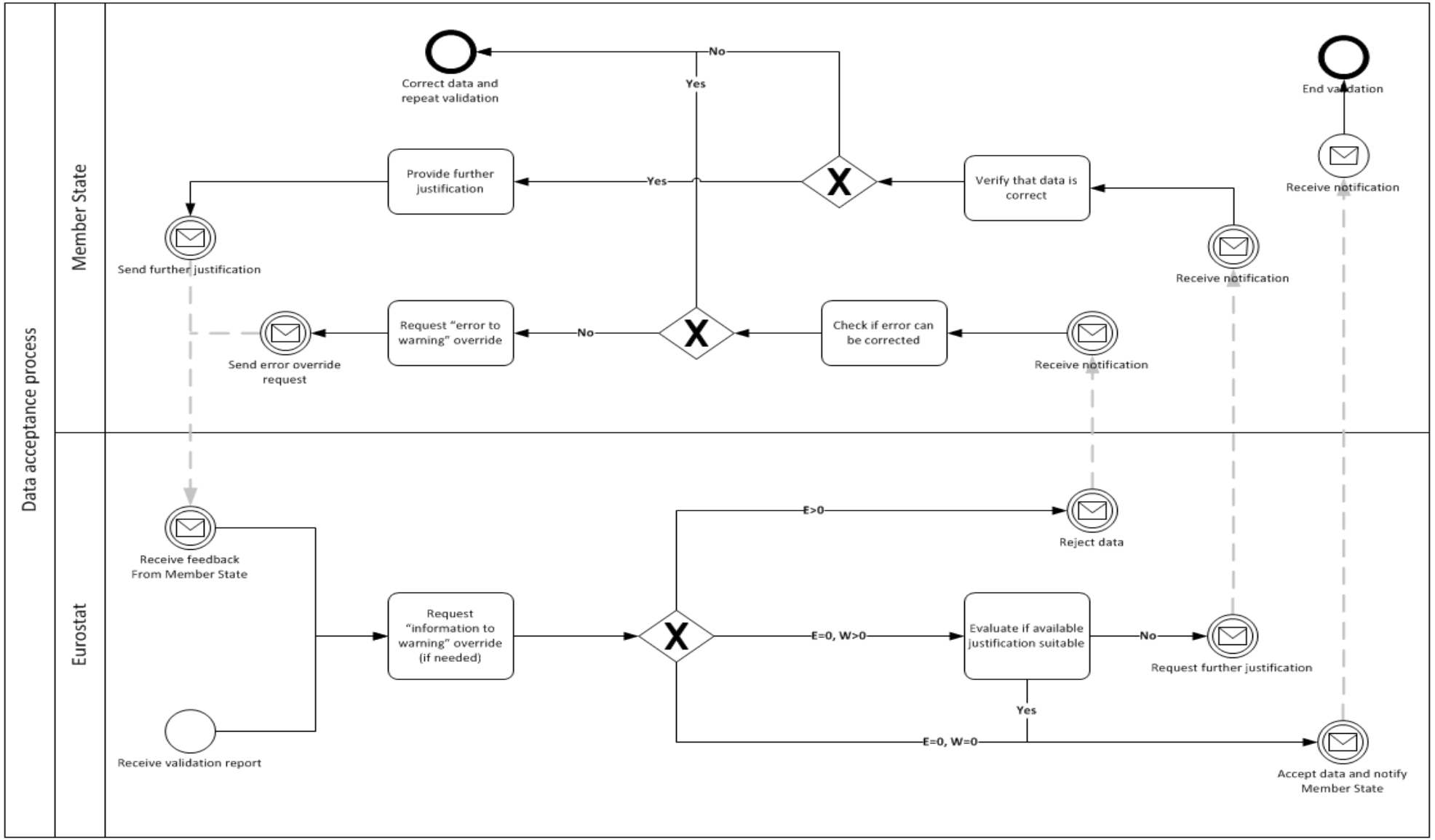
Many Eurostat data collections are based on a legal framework which specifies deadlines for the transmission of data. Different domains have established different procedures to judge whether these legal obligations have been respected by data providers: some domains only look at the date of the first

transmission, while others take into account quality criteria to judge when the first transmission of valid data occurred. While each domain retains the possibility to set its own quality standards, Eurostat will instruct statistical domains to consider as a minimum quality requirement that data providers must provide data *without errors* before the legal transmission deadline in order to be deemed compliant.

The diagram on next page illustrates the acceptance process. It should be noted that, given the importance of justifications/comments by data providers in the data acceptance process, Eurostat will ensure that Member States have the possibility to directly provide such comments alongside the data itself when transmitting the data to Eurostat.

Relevant principles

The *Accept data* business function implements principle 5, *Comply or explain*. Data sent to Eurostat must abide by the agreed upon validation rules. If they do not, Member States should be prepared to provide justification.


Data acceptance process
Member State
Eurostat
Correct data and repeat validation
No
Yes
End validation
Receive notification
Provide further justification
Yes
Verify that data is correct
Send further justification
Receive notification
Request "error to warning" override
No
Check if error can be corrected
Send error override request
Receive notification
Receive feedback From Member State
E>0
Reject data
Request "information to warning" override (if needed)
E=0, W>0
Evaluate if available justification suitable
No
Request further justification
Yes
Receive validation report
E=0, W=0
Accept data and notify Member State

References

Eurostat (2013) Exhaustive and detailed typology of validation rules v01306. *Working document* available at https://webgate.ec.europa.eu/fpfis/mwikis/essvalidserv/images/3/30/Eurostat_-_definition_validation_levels_and_other_related_concepts_v01307.doc

Eurostat (2017) Business Architecture for ESS Validation, at https://ec.europa.eu/eurostat/cros/system/files/business_architecture_for_ess_validation_-_final.pdf

ESSnet Validat Foundation (2015), ESSnet Validat Integration (2017), ESS Handbook – Methodology for data validation 1.1 at https://ec.europa.eu/eurostat/cros/content/work-package-1_en