# EDIT User Guide – Version 1.0

## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
<td>3</td>
</tr>
<tr>
<td>1.1</td>
<td>WHAT IS EDIT?</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>HOW EDIT WORKS?</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>GETTING STARTED</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>LOGGING TO EDIT</td>
<td>5</td>
</tr>
<tr>
<td>2.1.1</td>
<td>EDIT User types</td>
<td>5</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Username and password</td>
<td>6</td>
</tr>
<tr>
<td>2.1.3</td>
<td>Welcome Page</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>A QUICK TOUR OF EDIT (USER AND PROGRAMMERS)</td>
<td>6</td>
</tr>
<tr>
<td>2.2.1</td>
<td>User workflow</td>
<td>6</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Import datasets</td>
<td>7</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Create and Execute a job</td>
<td>10</td>
</tr>
<tr>
<td>2.2.4</td>
<td>Job results</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>A TOUR OF EDIT</td>
<td>17</td>
</tr>
<tr>
<td>3.1</td>
<td>DOMAINS</td>
<td>17</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Definition</td>
<td>17</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Create Formats</td>
<td>18</td>
</tr>
<tr>
<td>3.1.3</td>
<td>Import Formats</td>
<td>19</td>
</tr>
<tr>
<td>3.1.1.1</td>
<td>Basics of EDIT Scripting Language</td>
<td>22</td>
</tr>
<tr>
<td>3.1.1.1.1</td>
<td>Introduction</td>
<td>22</td>
</tr>
<tr>
<td>3.1.1.2</td>
<td>Possible types of rules/checks</td>
<td>22</td>
</tr>
<tr>
<td>3.1.1.2.1</td>
<td>Cell level rules</td>
<td>22</td>
</tr>
<tr>
<td>3.1.1.2.2</td>
<td>Horizontal rules</td>
<td>23</td>
</tr>
<tr>
<td>3.1.1.2.3</td>
<td>Vertical rules</td>
<td>23</td>
</tr>
<tr>
<td>3.1.1.2.4</td>
<td>Hierarchical – use of multiple data sets</td>
<td>24</td>
</tr>
<tr>
<td>3.1.1.3</td>
<td>Validation rules layout</td>
<td>25</td>
</tr>
<tr>
<td>3.1.1.3.1</td>
<td>Calculation rule</td>
<td>26</td>
</tr>
<tr>
<td>3.1.3.1</td>
<td>Import a program</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>DATASETS</td>
<td>27</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Definition</td>
<td>27</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Import/Export Datasets</td>
<td>30</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Import Code lists</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>FREQUENTLY ASKED QUESTIONS (FAQ)</td>
<td>33</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

1.1 What is EDIT?

EDIT is a tool for data validation and data edit/imputation

- What is data validation? - An activity aimed at verifying whether the value of a data item comes from the given set of acceptable values;
- What is data editing? - The activity aimed at detecting and correcting errors in data;
- What is data imputation? - Substitution for a specific data item where the response is missing or unusable.

1.2 How EDIT works?

Roughly speaking, when a user is working with it, EDIT’s workflow is this:

- uploads dataset(s) from external files,
- according to a predefined format,
- a program containing rules and file operations is executed on the dataset(s) and
- a report containing errors (if any), is created

However, this is only the tip of the iceberg, as in order for this path to be possible, programmers need to do a preparatory work, to define all the necessary ingredients like formats, lists of rules, programs, datasets, code lists and so on.

Later on, we will see what a format, a program and datasets are in EDIT. For now, it’s enough to bear in mind that:

- A format contains a description of the data in a dataset;
- A program contains instructions on what operations and checks should be performed on dataset(s);
- A dataset is a set of data according to a specific format.

1.3 Scope of this guide

The aim of this user guide is to provide an overview of the EDIT functionality to help users to validate datasets and to give some basics to the EDIT scripting language to programmers. This language is a real and powerful language of programming, allowing not only complex intra and inter datasets validations, but also dataset processing (e.g. aggregation). The scope of this guide is limited to the explanation of simple validation rules (the most used). Programmers are invited to consult the following documents to get deeper information on programming with EDIT:
- “Building Blocks Evolutionary and Corrective Maintenance - EBB Scripting Language Specification”
- Definition of validation levels and other related concepts
- Guidelines for the attribution of responsibility of validation activities in the whole production chain
- Procedure for the selection of rules to satisfy minimum quality standards
- Structured inventory of existing and possible validation rules (2nd_draft)
- Example descriptions of validation and production processes_20130308.doc
- Typology of data validation rules and imputation methods_20120921.doc
- Template and guidelines for the documentation of the validation process_20130311.doc
- Template and guidelines for the documentation of statistical production processes (4th_final).doc
- Exhaustive and detailed typology of validation rules_v3.doc
2  GETTING STARTED

2.1 LOGGING TO EDIT

2.1.1 EDIT USER TYPES

User
– Executes programs on datasets and accesses the reports.

Programmer
– Manages the metadata needed by the user to execute programs;
– Implements ‘formats’;
– Implements ‘validation rules’ by mean of ‘programs’;
– Defines other operations on files by mean of ‘programs’;
– Sets up the unattended mode configuration.

Administrator
– Manages users and permissions.

The Programmer and the User screens of EDIT are displayed below. The difference is that the User has access to three menus out of the seven and consequently, has access to less functionality.
2.1.2 **USERNAME AND PASSWORD**

When you use EDIT as a standalone application, you choose any of the predefine roles (User, Programmer or Administrator). If you access to the application running in a Data Centre you will need ECAS credentials.

![Login Screen](image)

2.1.3 **WELCOME PAGE**

Once logged in EDIT, you arrive to the home page where are displayed the User Profile information. Here you can change your password by clicking on the icon at the bottom left of the screen.

![Welcome Page](image)

2.2 **A QUICK TOUR OF EDIT (USER AND PROGRAMMERS)**

2.2.1 **USER WORKFLOW**

The ‘User’ Workflow is somehow linear: Import Dataset -> Execute a job on imported data -> See the result of the execution and, optionally, Export the result of the execution in the form of new datasets or validation reports (see below a picture listing them).
The link between User workflow and User interface can be better understood by following the schema below. In this schema, the user interface is represented only by the three menus accessible to a User, i.e. Home, Datasets and Jobs.

### 2.2.2 IMPORT DATASETS

To import a Dataset, just go in menu Datasets>>Import Datasets. The description of the screen is split into two parts.

In the first part, follow these steps:
1. In section *File Import*, select a file (CSV or FLR format) on your hard drive. You can customize the maximum number of errors found during the importation by providing a value for *Max number of errors*. Indeed, a validation is made during the importation against the information contained into the format you will select.

2. In section *File properties*, select the input file format (CSV or FLR). The combo box *File properties* is populated by previously saved configurations. It is empty if no configuration were saved.

3. In the *CSV section*, enter the parameters corresponding to your file. Set the starting row by providing a value for *Skip Header Lines*. Be careful to enter the correct *Field Delimiter* to avoid errors during the importation.

At the bottom right of this section, icons allow you to *delete, save or save as* your current configuration.

In the second part of the screen, follow these steps:

1. In section *File Fields*, select the format to be used for your dataset in the combo box *Dataset Format*. The combo box *File Fields* is populated by previously saved configurations. It is empty if no configuration were saved.

2. In section *Select Fields*, you choose which columns of your dataset you want to import. On the left side are listed the fields contained in the format selected above. Button ‘===>’ move all fields on the right side. You can of course click on each field separately and then click on button ‘=>’ to move it on the right. Fields listed on the right will be the columns imported.

3. In section *Dataset*, you can either create a new dataset or repopulate an existing one. In the first case, enter a name for your new dataset. The second case, select an existing dataset in the combo box.

Then these steps are completed, click on *Import* at the bottom right of the screen.
Then EDIT redirects you to the Dataset Imports/Exports screen, in which you can monitor the importing process. After a while, according to the size of your dataset, EDIT displays a message in the status column:

FAILED - means that there are critical errors preventing from importing the file. To download the importing report, click on the yellow triangle in the Action column.

![Dataset Import / Export History](image)

COMPLETED (with a yellow triangle in the Action column) - means that part of the file was imported. To download the importing report, click on the yellow triangle in the Action column.

![Dataset Import / Export History](image)

COMPLETED - means that the entire file was imported.

![Dataset Import / Export History](image)

In each situation above, you see the imported dataset by clicking on icon 📊 in the Action column. You do so, EDIT will display the specific screen below, in which different filters allow you to customize the dataset view.
2.2.3 CREATE AND EXECUTE A JOB

A job is just launching a program that will check a dataset. To see the list of available programs, go to menu Jobs>>Create Job.

The Program Search section displays a form to enter search criteria to filter the list of available programs. The Program list section shows the list of programs according to your search criteria. The information in columns Program Name, Description, and Created by will help you to identify the program to run.
If you need to see the content of the program, just click on icon in the Action column.

Keep in mind that the language used is specific to EDIT and need to consult the EDIT specific documentation to be understood.

Once you identified the program to run, click on icon in the Action column to create the job. Then EDIT displays the screen below in which:

1. You provide the Job Name and Description in section General Information;
2. You select the dataset you imported previously in section Dataset References;
3. You click on Execute at the bottom right of the screen to start the validation.
Then EDIT displays the following screen where you monitor the validation process.

When COMPLETED is displayed in the Status Column, you access to the results by clicking on the blue circle in the Action column. EDIT displays another screen that shows you the error reports and the input dataset.

### 2.2.4 JOB RESULTS

The validation report of EDIT contains:

1. Job results - information about the job;
2. **Error statistics** - summary of the errors;

3. **Error report** - list of errors.

The Job Results contains information about the Job name, the program launched, the error report, input datasets and output datasets.

See below the picture of the screen reporting on job results:

Click on icon 📄 next to an *error* dataset to see the error report to display a detailed screen in which different filters allow you to customize the report and then export it.
Click on icon next to an error dataset to download the detailed statistics report in CSV format. The next figure provides an example of report:

The Error statistics contains an overall statistic by rule. They are displayed in a table format and it consists of the following columns:

- Rule name: The name of the program rule that failed;
- No of Failures: Individual rows that the error appeared through job execution;
- Rule Message: Rule’s error message as defined in the program.
The screen for error statistics in EDIT:

A more comprehensible result is obtained for the domain EHSIS as the messages are more explicit.

The detailed error report lists all failures and provide information about the rule, row of the table were the failure appeared, the wrong values, the error message for that rule. See below an example coming from EHSIS.
### Example

The rule no 2 in the previous table, as seen in the EHSIS_test programme and in the table above. Try to identify the correspondent elements in the program and in the validation report. This is very important as we can tailor partially the error report by specifying in a proper way the error messages and the information we ask to be displayed in case of failure of a rule.

<table>
<thead>
<tr>
<th>No</th>
<th>MESSAGE</th>
<th>SEVERITY</th>
<th>BLOCK INDEX</th>
<th>EXP INDEX</th>
<th>EXP ID</th>
<th>EXP NAME</th>
<th>PARTICION</th>
<th>AUXILIARY DATA</th>
<th>DSI ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>This region code is not valid</td>
<td>Error</td>
<td>0</td>
<td>5</td>
<td>2786</td>
<td>RC07</td>
<td>ROW_NUMBER=1</td>
<td>REGION = &quot;AT&quot;</td>
<td>112</td>
</tr>
<tr>
<td>2</td>
<td>Invalid value (IF MAINSTAT in(20, 31, 32, 33, 34, 35, 36, -1) then JOBISCO should be -2)</td>
<td>Error</td>
<td>0</td>
<td>275</td>
<td>3056</td>
<td>SC04</td>
<td>ROW_NUMBER=3</td>
<td>MAINSTAT = 20, JOBISCO = 2</td>
<td>112</td>
</tr>
<tr>
<td>3</td>
<td>Invalid value (IF MAINSTAT in(20, 31, 32, 33, 34, 35, 36, -1) then LOCNACE should be -2)</td>
<td>Error</td>
<td>0</td>
<td>276</td>
<td>3057</td>
<td>SC05</td>
<td>ROW_NUMBER=3</td>
<td>MAINSTAT = 30, LOCNACE = 7</td>
<td>112</td>
</tr>
<tr>
<td>4</td>
<td>This region code is not valid</td>
<td>Error</td>
<td>0</td>
<td>5</td>
<td>2786</td>
<td>RC07</td>
<td>ROW_NUMBER=4</td>
<td>REGION = &quot;AT&quot;</td>
<td>112</td>
</tr>
<tr>
<td>5</td>
<td>Invalid value (IF MAINSTAT in(20, 31, 32, 33, 34, 35, 36, -1) then LOCNACE should be -2)</td>
<td>Error</td>
<td>0</td>
<td>276</td>
<td>3057</td>
<td>SC05</td>
<td>ROW_NUMBER=4</td>
<td>MAINSTAT = 31, LOCNACE = 7</td>
<td>112</td>
</tr>
<tr>
<td>6</td>
<td>This region code is not valid</td>
<td>Error</td>
<td>0</td>
<td>5</td>
<td>2786</td>
<td>RC07</td>
<td>ROW_NUMBER=5</td>
<td>REGION = &quot;AT&quot;</td>
<td>112</td>
</tr>
<tr>
<td>7</td>
<td>Invalid value (IF MAINSTAT in(20, 31, 32, 33, 34, 35, 36, -1) then LOCNACE should be -2)</td>
<td>Error</td>
<td>0</td>
<td>276</td>
<td>3057</td>
<td>SC05</td>
<td>ROW_NUMBER=5</td>
<td>MAINSTAT = 32, LOCNACE = 7</td>
<td>112</td>
</tr>
</tbody>
</table>
3 A TOUR OF EDIT

3.1 DOMAINS

A Domain in EDIT is an isolated workspace inside EDIT, or a virtual area with personal access rights;

- It contains formats, code lists, datasets, programs (see below);
- A program can use different formats inside the same domain;
- No object in two different domains can interact with each other.

A Domain in EDIT simulates what in Eurostat is known as a ‘project’, projects are sometimes called even ‘Domains’.

3.1 FORMATS

3.1.1 DEFINITION

The Format is the Dataset Definition. It defines the structure of the data. It is identified by a Name inside a Domain. It has also a Description and is composed by a set of Fields.

A field is defined by Name, Description, Caption, Type, Default value, Length, Precision, Mandatory.

EDIT allows the user to define formats by mean of a specific interface. However, as some domains have formats containing several hundreds of Fields, for the ease of the user, EDIT can import formats externally built by using a normal editor.

Below there is an example the way EAS format is encrypted in an external file created by using a normal editor (see next figure):
For this format:

- the name is ‘aesf_v05’,
- the description is ‘Format for eas_2011’

5 field are described by ‘Name’, ‘Description’, ‘Caption’, ‘Type’, ‘Length’ and ‘Mandatory’.

For example, the name of the first field is ‘CTRY’.

Below, one can see the way EDIT lists the content of a format file by mean of the user interface.

<table>
<thead>
<tr>
<th>Order</th>
<th>Name</th>
<th>Caption</th>
<th>Default Value</th>
<th>Mandatory</th>
<th>Data Type</th>
<th>Length</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ROW_NUMBER</td>
<td>ROW_NUMBER</td>
<td>false</td>
<td>NUMBER</td>
<td>10</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>COUNTRY</td>
<td>None</td>
<td>false</td>
<td>STRING</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ENTERPRI</td>
<td>None</td>
<td>false</td>
<td>NUMBER</td>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>REPYEAR</td>
<td>None</td>
<td>false</td>
<td>NUMBER</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>WEIGHT</td>
<td>None</td>
<td>false</td>
<td>DOUBLE</td>
<td>20</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>NACE_SP</td>
<td>None</td>
<td>false</td>
<td>STRING</td>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>SIZE_SP</td>
<td>None</td>
<td>false</td>
<td>NUMBER</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>NETRA_SP</td>
<td>None</td>
<td>false</td>
<td>NUMBER</td>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>N_SP</td>
<td>None</td>
<td>false</td>
<td>NUMBER</td>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SUB_SP</td>
<td>None</td>
<td>false</td>
<td>NUMBER</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>N_RESFET</td>
<td>None</td>
<td>false</td>
<td>NUMBER</td>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>N_EMPREG</td>
<td>None</td>
<td>false</td>
<td>NUMBER</td>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>RESPONSE</td>
<td>None</td>
<td>false</td>
<td>NUMBER</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>PROC</td>
<td>None</td>
<td>false</td>
<td>NUMBER</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>ISLANGLA</td>
<td>None</td>
<td>false</td>
<td>STRING</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>IDREGION</td>
<td>None</td>
<td>false</td>
<td>STRING</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>EXTRA1</td>
<td>None</td>
<td>false</td>
<td>NUMBER</td>
<td>10</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>EXTRA2</td>
<td>None</td>
<td>false</td>
<td>NUMBER</td>
<td>10</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>EXTRA3</td>
<td>None</td>
<td>false</td>
<td>NUMBER</td>
<td>10</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>AI1</td>
<td>None</td>
<td>false</td>
<td>STRING</td>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>AI2</td>
<td>None</td>
<td>false</td>
<td>DOUBLE</td>
<td>12</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

### 3.1.2 CREATE FORMATS

To add a new format, go to menu Formats>>Add Format. Enter a Name and a Description. Then, click on Add field to add a new field in the Fields table. Finally, fill in each column. Repeat the operation for each new field. Note that a Default value must be provided for mandatory fields.
Once your table is completed, click on Save. EDIT redirects you to the «search format screen» where you will see your format displayed in the list. The four icons in the Action column allow you to view, edit, delete or export the Format.

In the Format Search section, you can enter the name and the author of a Format you may look for. EDIT will filter the Format List accordingly.

### 3.1.3 IMPORT FORMATS

To import a format, go to menu Formats > Import Format. Select a file (format .fdl) and click on Upload.
If the importation was successful, EDIT displays the format in the search format screen.

3.1 Programs

3.1.1 Definition

A Program is a set of operations to be performed on a specified dataset definition (Format). No specific dataset is associated with a program; only Formats (dataset definitions) should be specified. Programs allow several operations types of rules/checks: Single and Multiple columns, Vertical and Hierarchical.

Below is a simple example of a program in EDIT. It contains a ‘Header’ and a list of rules. The header and the first rule are explained line by line.

```
# EXPORTED BY EDIT v01  //Comment line
PROGRAM cvts_train  //Keyword ‘PROGRAM’ and the name of the program: ‘cvts_train’
```
EDIT User Guide – Version 1.0

INPUT cvts_train inputDataSet; //Keyword ‘INPUT’, the name of the format which defines the layout of the dataset ‘cvts_train’

STEPS {} //Keyword STEPS announces the beginning of validation checks/rules/operations

VALIDATION annex8_error {} //The first step will be a ‘VALIDATION’ step. There can be other types of steps like ‘DATAOPERATION’ steps

INPUT inputDataSet; //Tells the compiler to use this dataset. Several different datasets can be used and changed when //changing the step. All datasets used in different steps have to be declared once in line 3 and then in //this line

ERROR err_annex8_error; // The name of the dataset containing errors that will be created by the EDIT.

RULES {} //Tells EDIT that a set of rules starts

RECORD FL002 {} //The first rule is a ‘RECORD’ rule i.e. concerning a record in the dataset. The name of the rule is //‘FL002’.

CONDITION (ENTERPR>=0 AND ENTERPR<=999996); //The condition the rule is asking to be fulfilled. ENTERPR is the //name of a field in the dataset.

ERRMSG “Rule FL2 failed for field [ENTERPR]: In the range 0 to 999996” SEVERITY “Error” (ENTERPR) ; } //This line contains the error message to be displayed when the rule //fails, the SEVERITY of the failure and the name(s) of fields to be //displayed in the validation report as having wrong values.

3.1.2 CREATE A PROGRAM

In order to create a program from scratch, EDIT has a specific editor you access through menu Programs>>Add Program. The Editor is composed of four frames: Toolbox, Navigator, Editor and Editor Toolbox. You need of course to master the EDIT Scripting Language to use the editor. The interface is just a wizard helping you to avoid mistyping.
3.1.3 Basics of EDIT Scripting Language

3.1.3.1 Introduction

EDIT has a custom ‘Scripting Language’ designed specifically for data validation. This language tries to be as simple as possible and still flexible enough to fit the requirements of any existing domain. It allows the definition of ‘Formats’ and validation ‘Programs’. Formats (Dataset Definitions) describe the structure of the data (Format Definition Language). Validation ‘Programs’ describe the validation rules and are composed from a set of steps with inputs and outputs (Program Definition Language).

3.1.3.2 Possible types of rules/checks

3.1.3.2.1 SINGLE COLUMN RULES

- Specify the column,
- For every record,
- A check or a calculation is performed on the cell at the intersection of the two.

Example for this format in the picture: RESPWEIGHT < 20;

In the EDIT philosophy, any operation defined on a cell inside a record is automatically performed on all records, in order, one by one.

This kind of rules may check:

- Whether the recorded entries are among a set of valid codes for the field;
- Whether the entry of any field is an invalid blank (for some fields ‘blank’ entries may be valid).

Examples:

- ‘Production’ is not negative;
- ‘Quantity’ in tons for a given industry should be less than 75,000 tons.
3.1.3.2.2 MULTIPLE COLUMNS RULES

- Specify a column,
- For every record,
- If the information in the cell at the intersection fulfils a condition,
- On another column, on the same record,
- Check if the information fulfils another condition, or do a computation.

Example for the format in the picture above: If HATLEVEL in \{1, -1\} then HATYEAR = -2

- They can refer ‘sets of values’ for specified combinations of fields which are jointly unacceptable (or, equivalently, which are jointly acceptable);
- Can refer different cells on the same row of the table (same record);
- Are usually specified on the basis of extensive knowledge of the subject matter of data.

Example:
The input of patent fuel plants (of a given fuel type) must be larger than its own output (reported in the same record) = it means that the efficiency of such patent fuel plant cannot be more than 100%.

3.1.3.2.3 VERTICAL RULES

To explain such a rule, let’s take an example.

- Four variables: COUNTRY, MONTH, PRODUCT, VALUE;
- Aggregation of VALUE by COUNTRY -> a partition of the dataset;
- Two countries => the partition is composed by two sets;
  - Set 1: Records 1 to 6;
  - Set 2: Records 7 to 12;
- The two sets are merged on the rest of three variables and a volatile version of the dataset is produced (see below);
- A supplementary record is added for the aggregate (in this example: SUM).
These rules involve a data integrity check for entries across a collection of related records.

**Examples:**

- Total number of imports for a given product is equal to the sum of imports from individual countries;
- Stock value in the beginning of month is equal to the closing stocks in the previous month;

### 3.1.3.2.4 Hierarchical - Use of Multiple Data Sets

To explain such a rule, let’s take an example.

Two datasets: ‘Households’ and ‘Persons’

- **Households** contains:
  - The column called ‘Id’ to identify the household;
  - A column called ‘NoP’ = number of members in the household;
  - A column called ‘HIncome’, the total income of the members of the household.

- **‘Persons’** contains:
  - A column called ‘Name’;
  - A column called ‘Hid’ = the Id of the household the person belongs to;
  - A column called ‘Income’.

One can do then:

- To check whether the number of persons for each household in ‘Households’ equals the number of persons in ‘Persons’ belonging to that household;
- To calculate the total income by household using data in ‘Persons’ and to check the coherency with data in ‘Households’.

**Example for hierarchical rule**
For the household 1002 in ‘Households’ dataset, for all the records in dataset ‘Persons’ for which the ‘Hid’ is 1002, the sum of ‘Income’ should equal the ‘Hincome’ in the first dataset.

### 3.1.3.1 Validation rules layout

- Rule type (RECORD, VERTICAL, HIERARCHICAL);
- Rule name;
- Rule body:
  - Condition = logical expression referencing fields in the data set;
  - 'Then' compute part - optional;
  - 'Else' compute part - optional;
  - Error part (error message, selected data to be displayed).

The list of validation rules to be implemented should then contain:

- 'Id' of the rule;
- Name of the rule;
- The condition to be fulfilled by the rule;
- ‘Then’ part, if necessary;
- ‘Else’ part, if necessary;
- Error message - the reason why the rule failed (if needed);
- The information to be provided to the user if failure.

**Example:** validation rule specified by the user and the implementation in EDIT. In the small table below one can see the validation rule ‘RC02’ as it appears in the table specifying validation rule for the domain EHSIS.

<table>
<thead>
<tr>
<th>RCO2</th>
<th>WHOINTER in(1,2,3,-1,-2)</th>
<th>ERROR</th>
<th>WHOINTER is missing or does not contain the expected values</th>
</tr>
</thead>
</table>

Then, the way the rule is implemented in EDIT is the following:
EDIT User Guide – Version 1.0

RECORD RC02

{ CONDITION in(WHOINTER,1, 2, 3, -1, -2);
  ERRMSG 'WHOINTER is missing or does not contain the expected value' SEVERITY 'Error' (WHOINTER) ;
}

Note: most of type ‘Single column’ validation rules can be specified and implemented in a similar way;

3.1.3.1.1 Calculation rule

Let’s follow below the description for a calculation rule as it is implemented in EDIT:

```
RECORD conditionalRecord {
  CONDITION isNull(VALUE);
  THEN {
    VALUE := PRICE * QUANTITY;
  }
  ELSE {
    PRICE := VALUE / $;
    QUANTITY := VALUE / PRICE;
  }
}
```

Note: For this simple rule, the ‘ERROR’ part of the rule is missing

Checking rule: let’s now see the description for a checking rule, when ‘Then’ and ‘Else’ parts are not specified anymore in the expression of the rule but they exist by default. This improves the simplicity of expressing the rules.

RECORD RC02

{ CONDITION in (WHOINTER, 1, 2, 3, -1, -2);
  ERRMSG 'WHOINTER: should be in (1, 2, 3, -1, -2) SEVERITY 'Error'
  (WHOINTER) ;
}

Explanation:

– First row, TYPE and NAME of the rule;
– Following rows: the rule body, lines 2 and 3;
– WHOINTER - a column in the dataset;
– Third line: the ERROR part, acting only if the condition fails;
3.1.4 Import a Program

Another way to create a program (once you master the EDIT Scripting Language) is to start from an existing program (previously exported) and modify it using a text editor. Then, you import the program in EDIT. To do so, go to menu Programs>>Import Program.

Select a file (.pdl) on your hard drive. Then, select option Rename existing Program if you want to import a program that you have already imported with a different name. Click on Import. Note that a format should first be created before importing a program that uses it.

After importation, EDIT displays the search program screen.

3.2 Datasets

3.2.1 Definition

A Dataset Instance (Dataset) is a collection of data rows according to the structure of a format. It is a two dimensional table composed by rows and columns. Columns correspond to the fields defined in the format and records. There is no limit on size or number. Below, is a dataset example - Table AES (Adult Education Survey):
The description of the table AES was provided in a table like the one below. The source of information may be regulation, or a gentlemen agreement.

<table>
<thead>
<tr>
<th>No</th>
<th>CTRY</th>
<th>REG</th>
<th>URB</th>
<th>REFYEAR</th>
<th>REFMONTH</th>
<th>RESPID</th>
<th>RESPWEIGHT</th>
<th>SEX</th>
<th>BIRTHYEAR</th>
<th>BIRTHMONTH</th>
<th>HATLEVEL</th>
<th>HATYEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AT</td>
<td>AT1</td>
<td>1</td>
<td>2011</td>
<td>-1</td>
<td>1</td>
<td>5.5</td>
<td>1</td>
<td>1941</td>
<td>-1</td>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>AT</td>
<td>AT2</td>
<td>2</td>
<td>2012</td>
<td>12</td>
<td>2</td>
<td>10.6</td>
<td>2</td>
<td>1994</td>
<td>12</td>
<td>11</td>
<td>2005</td>
</tr>
<tr>
<td>3</td>
<td>AT</td>
<td>AT3</td>
<td>3</td>
<td>2011</td>
<td>-1</td>
<td>3</td>
<td>20.6</td>
<td>1</td>
<td>1959</td>
<td>12</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>AT</td>
<td>AT1</td>
<td>1</td>
<td>2011</td>
<td>5</td>
<td>5.5</td>
<td>30.5</td>
<td>1</td>
<td>1960</td>
<td>3</td>
<td>31</td>
<td>1980</td>
</tr>
<tr>
<td>5</td>
<td>AT</td>
<td>AT2</td>
<td>2</td>
<td>2011</td>
<td>8</td>
<td>10.5</td>
<td>3</td>
<td>2</td>
<td>1984</td>
<td>4</td>
<td>11</td>
<td>2005</td>
</tr>
<tr>
<td>6</td>
<td>AT</td>
<td>AT3</td>
<td>3</td>
<td>2011</td>
<td>3</td>
<td>8</td>
<td>1.2</td>
<td>2</td>
<td>1994</td>
<td>3</td>
<td>5</td>
<td>2005</td>
</tr>
<tr>
<td>7</td>
<td>AT</td>
<td>AT1</td>
<td>1</td>
<td>2011</td>
<td>5</td>
<td>10</td>
<td>6.8</td>
<td>1</td>
<td>1992</td>
<td>6</td>
<td>51</td>
<td>2003</td>
</tr>
<tr>
<td>8</td>
<td>AT</td>
<td>AT2</td>
<td>2</td>
<td>2011</td>
<td>8</td>
<td>11</td>
<td>5.4</td>
<td>1</td>
<td>1960</td>
<td>7</td>
<td>40</td>
<td>2000</td>
</tr>
<tr>
<td>9</td>
<td>AT</td>
<td>AT3</td>
<td>3</td>
<td>2011</td>
<td>3</td>
<td>21</td>
<td>8.9</td>
<td>2</td>
<td>1980</td>
<td>8</td>
<td>60</td>
<td>1999</td>
</tr>
<tr>
<td>10</td>
<td>AT</td>
<td>AT1</td>
<td>1</td>
<td>2011</td>
<td>1</td>
<td>22</td>
<td>5.5</td>
<td>1</td>
<td>1941</td>
<td>1</td>
<td>1</td>
<td>-2</td>
</tr>
<tr>
<td>11</td>
<td>AT</td>
<td>AT2</td>
<td>2</td>
<td>2011</td>
<td>12</td>
<td>23</td>
<td>10.6</td>
<td>2</td>
<td>1994</td>
<td>12</td>
<td>11</td>
<td>2005</td>
</tr>
<tr>
<td>12</td>
<td>AT</td>
<td>AT3</td>
<td>3</td>
<td>2011</td>
<td>-1</td>
<td>24</td>
<td>20.6</td>
<td>2</td>
<td>1959</td>
<td>12</td>
<td>11</td>
<td>-1</td>
</tr>
<tr>
<td>13</td>
<td>AT</td>
<td>AT1</td>
<td>1</td>
<td>2011</td>
<td>5</td>
<td>25</td>
<td>30.5</td>
<td>1</td>
<td>1960</td>
<td>3</td>
<td>31</td>
<td>1980</td>
</tr>
<tr>
<td>14</td>
<td>AT</td>
<td>AT2</td>
<td>2</td>
<td>2011</td>
<td>8</td>
<td>26</td>
<td>10.5</td>
<td>2</td>
<td>1984</td>
<td>4</td>
<td>11</td>
<td>2005</td>
</tr>
<tr>
<td>15</td>
<td>AT</td>
<td>AT3</td>
<td>3</td>
<td>2011</td>
<td>3</td>
<td>27</td>
<td>1.2</td>
<td>2</td>
<td>1994</td>
<td>5</td>
<td>32</td>
<td>2005</td>
</tr>
<tr>
<td>16</td>
<td>AT</td>
<td>AT1</td>
<td>1</td>
<td>2011</td>
<td>5</td>
<td>28</td>
<td>6.8</td>
<td>1</td>
<td>1992</td>
<td>6</td>
<td>51</td>
<td>2003</td>
</tr>
<tr>
<td>17</td>
<td>AT</td>
<td>AT2</td>
<td>2</td>
<td>2011</td>
<td>8</td>
<td>29</td>
<td>5.4</td>
<td>1</td>
<td>1960</td>
<td>7</td>
<td>40</td>
<td>2000</td>
</tr>
<tr>
<td>18</td>
<td>AT</td>
<td>AT3</td>
<td>3</td>
<td>2011</td>
<td>3</td>
<td>30</td>
<td>8.9</td>
<td>2</td>
<td>1980</td>
<td>8</td>
<td>60</td>
<td>1999</td>
</tr>
<tr>
<td>19</td>
<td>AT</td>
<td>AT1</td>
<td>1</td>
<td>2011</td>
<td>1</td>
<td>31</td>
<td>5.5</td>
<td>1</td>
<td>1941</td>
<td>1</td>
<td>1</td>
<td>-2</td>
</tr>
<tr>
<td>20</td>
<td>AT</td>
<td>AT2</td>
<td>2</td>
<td>2011</td>
<td>12</td>
<td>32</td>
<td>10.6</td>
<td>2</td>
<td>1994</td>
<td>12</td>
<td>11</td>
<td>2005</td>
</tr>
<tr>
<td>21</td>
<td>AT</td>
<td>AT3</td>
<td>3</td>
<td>2011</td>
<td>-1</td>
<td>33</td>
<td>20.6</td>
<td>2</td>
<td>1959</td>
<td>12</td>
<td>11</td>
<td>-1</td>
</tr>
<tr>
<td>22</td>
<td>AT</td>
<td>AT1</td>
<td>1</td>
<td>2011</td>
<td>5</td>
<td>34</td>
<td>30.5</td>
<td>1</td>
<td>1960</td>
<td>3</td>
<td>31</td>
<td>1980</td>
</tr>
<tr>
<td>23</td>
<td>AT</td>
<td>AT2</td>
<td>2</td>
<td>2011</td>
<td>8</td>
<td>35</td>
<td>10.5</td>
<td>2</td>
<td>1984</td>
<td>4</td>
<td>11</td>
<td>2005</td>
</tr>
<tr>
<td>24</td>
<td>AT</td>
<td>AT3</td>
<td>3</td>
<td>2011</td>
<td>3</td>
<td>36</td>
<td>1.2</td>
<td>2</td>
<td>1994</td>
<td>5</td>
<td>32</td>
<td>2005</td>
</tr>
</tbody>
</table>

A special attention should be paid to the difference between a format and a dataset instantiation. A format can be seen as a container only and one can think that a large diversity of data can be accommodated in. Example: 'Format' - 'Data set instantiation'.
## Format - as a container

<table>
<thead>
<tr>
<th>No</th>
<th>CTRY</th>
<th>REG</th>
<th>LURB</th>
<th>REYEAR</th>
<th>REPNUM</th>
<th>RESPID</th>
<th>RESPWEIGHT</th>
<th>SEX</th>
<th>BIRTHYEAR</th>
<th>BIRTHMONTH</th>
<th>HATLEVEL</th>
<th>HATYEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AT</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2011</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2011</td>
<td>1</td>
<td>2011</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>AT</td>
<td>2</td>
<td>2</td>
<td>2012</td>
<td>10.6</td>
<td>2</td>
<td>1994</td>
<td>12</td>
<td>11</td>
<td>2005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>AT</td>
<td>3</td>
<td>3</td>
<td>2011</td>
<td>1</td>
<td>2</td>
<td>1993</td>
<td>1</td>
<td>21</td>
<td>-1</td>
<td>2005</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>AT</td>
<td>1</td>
<td>1</td>
<td>2011</td>
<td>5.5</td>
<td>1</td>
<td>1990</td>
<td>3</td>
<td>21</td>
<td>1990</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>AT</td>
<td>2</td>
<td>2</td>
<td>2011</td>
<td>15.5</td>
<td>2</td>
<td>1994</td>
<td>4</td>
<td>11</td>
<td>2005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>AT</td>
<td>3</td>
<td>3</td>
<td>2011</td>
<td>1</td>
<td>6.2</td>
<td>1934</td>
<td>5</td>
<td>12</td>
<td>2005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>AT</td>
<td>1</td>
<td>1</td>
<td>2011</td>
<td>10.6</td>
<td>1</td>
<td>1992</td>
<td>6</td>
<td>51</td>
<td>2003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>AT</td>
<td>2</td>
<td>2</td>
<td>2011</td>
<td>1</td>
<td>7.4</td>
<td>1980</td>
<td>7</td>
<td>40</td>
<td>2004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>AT</td>
<td>1</td>
<td>1</td>
<td>2011</td>
<td>1</td>
<td>8.0</td>
<td>1980</td>
<td>8</td>
<td>60</td>
<td>1999</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>AT</td>
<td>1</td>
<td>1</td>
<td>2011</td>
<td>1</td>
<td>5.5</td>
<td>1941</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>AT</td>
<td>2</td>
<td>2</td>
<td>2012</td>
<td>10.6</td>
<td>2</td>
<td>1994</td>
<td>12</td>
<td>11</td>
<td>2005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>AT</td>
<td>3</td>
<td>3</td>
<td>2011</td>
<td>-1</td>
<td>20.6</td>
<td>2059</td>
<td>-1</td>
<td>21</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>AT</td>
<td>1</td>
<td>1</td>
<td>2011</td>
<td>5.5</td>
<td>1</td>
<td>1940</td>
<td>5</td>
<td>51</td>
<td>1990</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>AT</td>
<td>2</td>
<td>2</td>
<td>2011</td>
<td>8</td>
<td>10.5</td>
<td>2</td>
<td>1994</td>
<td>4</td>
<td>11</td>
<td>2005</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>AT</td>
<td>1</td>
<td>1</td>
<td>2011</td>
<td>2</td>
<td>1</td>
<td>1941</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>AT</td>
<td>2</td>
<td>2</td>
<td>2012</td>
<td>10.6</td>
<td>2</td>
<td>1994</td>
<td>12</td>
<td>11</td>
<td>2005</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Data set instantiation

As an example that different data can enter the same format, see below, the same format and two different actual datasets:
3.2.2 Import/Export Datasets

To access to the Import/Export History in EDIT, go to menu Datasets>>Imports/Exports. In section Dataset Import/Export History Search, enter your search criteria to filter the Dataset Import/Export History list. In the list you will see any kind of dataset imported or exported. The type of dataset is displayed in the Format column.

In the Action column you can do the following:

- View task details
- Delete task
- Download data file
- Download log file

3.2.3 Import Code Lists

EDIT considers Code lists as datasets. They are named in the Scripting language as lookup tables. In EDIT a look-up is an auxiliary dataset containing a list of values to be used for validating codes. Code lists are usually lookup tables that refer to code lists. One can use several code lists inside the same program - as many as needed for the given data sets -
'COUNTRY', NACE, NUTS. Several versions of the same code list can be used from within the same program, if needed.

To import a list of codes and to use it for validation, you need to follow these steps:

1. Add (or import) a specific format;
2. Import the code list (CSV file) -> import dataset using the format previously defined;
3. Define the rules in a program.

This is an example of format for a classical code list (two fields: CODE and LABEL):

```
# EXPORTED BY EDIT V01
FORMAT COUNTRIES_LIST
  DESCRIPTION "Description for e.g. countries";
  FIELDS {
    CODE {
      DESCRIPTION "CODE";
      CAPTION "CODE";
      TYPE STRING;
      LENGTH 2 MANDATORY;
    }
    LABEL {
      DESCRIPTION "LABEL";
      CAPTION "LABEL";
      TYPE STRING;
      LENGTH 150;
    }
  }
```

First import the format above, then import the CSV file (two columns) containing the list of codes via the menu Datasets>>Import Dataset.

The next figure illustrates the example:
Select the input file
Select the CSV format
Select the starting row, decimal separator, field delimiter etc.
Select the format created in step 1 (e.g. COUNTRY_LIST)
Select both fields (CODE and LABEL)
Provide a name for the lookup (e.g. CL_COUNTRY)
Click to import
4 FREQUENTLY ASKED QUESTIONS (FAQ)

1. **Question**: Is the field specificity ‘Length’ compulsory when creating a new Format?
   **Answer**: EDIT always requires filling in this component.

2. **Question**: Is the layout and the information of the error report always the same?
   **Answer**: While the information included is exactly the same, upon request made by some ESTAT units, the error report’s layout may be different for some domains.

3. **Question**: Who will get a user account?
   **Answer**: There’s currently an implementation team working on developing specific validation rules which are domain-based. If everyone follows the tested-rules we could all benefit from harmonised reports. However, if a person wants to develop their own validation rules they will be allowed to do so.

4. **Question**: Does EDIT connect to the EURO-SDMX REGISTRY?
   **Answer**: Yes.

5. **Question**: If a DSD has been modified, do we have to import it to EDIT as a new FORMAT?
   **Answer**: It depends on the number of changes and the account used to access EDIT. As a programmer one can modify/create FORMATs but it is not the case if the system is entered with a user account.

6. **Question**: How developed is the scripting language used by EDIT?
   **Answer**: EDIT scripting language tries to be as simple as possible and still flexible enough to fit the requirements of any existing domain. There’s a specific document entitled *Building Blocks Evolutionary and Corrective Maintenance - EBB Scripting Language Specification* for more detailed information.

7. **Question**: Does EDIT support importing several data files at once?
   **Answer**: They have to be imported one by one. However, if EDAMIS is used, the system can import several data files as long as they are located in a specific folder.
8. **Question:** Can you import data from EUROBASE?

**Answer:** This is possible through the DATASET tab/ IMPORT DATASETS DB.

9. **Question:** If a DSD is modified and as a result we have a modified DSD, does the system update the changes automatically?

**Answer:** It has to be imported manually. FORMATS, PROGRAMs can be imported at once in a zip file.

10. **Question:** If you change a rule and that very same rule is used in another PROGRAM, does it change automatically?

**Answer:** It needs to be done one by one. It is probably quicker to use a text editor such as WordPad. First you export the PROGRAM, then change whatever needs to be modified and finally you import the PROGRAM back to the system.

11. **Question:** Does EDIT support to stop a JOB?

**Answer:** There is a function named ABORT to stop the validation process.

12. **Question:** Can you give more insight into the EDIT rules?

**Answer:** Rules can be very specific or flexible depending on the programmer.

13. **Question:** If I want to compare historical data, can it be uploaded by ESTAT and checked by MSs?

**Answer:** Data can be uploaded by MSs as well as ESTAT.

14. **Question:** Is EDIT able to read DSD 2.0?

**Answer:** Overall the answer is yes. However, some specificities may be lost when the system creates the format.

15. **Question:** Can you validate 28 files coming from all MSs at once or do you have to validate one by one?

**Answer:** EDIT can perform dataset operations such as merge. This functionality would allow you to create a merged file containing the data from all MSs and hence, allow you to run a specific program on the merged file.
16. **Question**: If I run a program and get an error report. Can it be seen by national users?

**Answer**: Error reports are saved by the system together with databases. This place can be accessed by users and domain managers of the application so the answer is yes, provided they have access to the same space.

17. **Question**: Is it possible to use a lookup file just to add some new data?

**Answer**: Lookups are objects intended mainly for codelists. New data should be imported to the system in the form of a new/updated dataset.

18. **Question**: I see in the error reports that a kind of formal language is used to specify the rule that was broken. Is there a formal language to specify the rules?

**Answer**: The "message" included in the error report is to be decided by the person (programmer) who creates the set of validation rules. It does not follow any specific language. However, validation rules do have/follow a specific structure. Please look at the script language guide.

19. **Question**: We are in the process of migrating to windows 8 … Will there be any problems?

**Answer**: EDIT is supported by Windows XP and 7. Regarding Windows 8 - it hasn't been tested so far and there are no plans to do so since it is not officially supported by the European Commission.

20. **Question**: Is EDIT multi-language able?

**Answer**: The translation of the tool into other languages is included within the framework of future development of the tool. The multilingual version is under test now.

21. **Question**: How does one define multiple datasets for type C validation?

**Answer**: If you are to create a hierarchical rule, this rule will include the required information on the datasets that are covered by it. So in the example of EU SILC, it will include the Households and the Persons (living in those households) datasets. Another possibility would be merging the two datasets and creating vertical/column(s) validation rules instead.

22. **Question**: Have I good understood, it is not yet possible to use a SDMX DSD to import code list in lookup file?

**Answer**: You are able to use SDMX ML format-like to upload a database or lookup. However, version 2.1 is not fully readable by the system yet.
23. **Question:** Is there a limit on the size of dataset files?

**Answer:** There is no limit on the number of records.

24. **Question:** Is CSV the only format to export a dataset or is it possible do export in SDMX ML too?

**Answer:** EDIT allows user to export using CSV and FLR formats for the time being.

25. **Question:** In this concept of Editing where is the place for users Data entry? Is it possible to add some columns in the data set with free editing and after submit jobs, programms etc.?

**Answer:** The Edit does not provide a data entry facility. You would have to include the column/s on the dataset before importing it/them to the system. However there are different operations available on datasets which have already been imported to the system. For more detailed information please take a look at the document entitled *EDIT Scripting Language Specification* located on [http://ec.europa.eu/eurostat/edit/](http://ec.europa.eu/eurostat/edit/)

26. **Question:** How was the e-mail address, which you said at the beginning of the presentation?

**Answer:** the functional email address to be used in order to get access to the tool is as follows ESTAT-VALIDATION@ec.europa.eu

27. **Question:** The language used to describe the rules. Is it standardized? Is it VALS, or something else?

**Answer:** The language used to create format definition files and program definition files is called the Edit scripting language. Please check document entitled EDIT scripting language (same as in answer to question 25).

28. **Question:** Is it possible to directly correct errors in EDIT or do we have to export the data, correct the error and then import the data again?

**Answer:** No for security reasons. If we alter the data manually we will have no trace of the change.

29. **Question:** Is there a manual with mathematical background about outliers and imputation in EDIT?

**Answer:** For imputation no (we impute through formulas only, not things as hot deck, but for outliers thee system is based on the description of the outliers building block. We will provide the useful part.