**Data Enrichment and Validation Step**

1. Data enrichment and validation – input and output

The data enrichment and validation procedure (DEAV) logically requires as input the following information:

1. **InputTable:** A reference to a table containing the input data which will be enriched/validated by the procedure. This means that the table will be also used as output for the enrichment, that is, the input table will be *updated* by adding the additional columns.
2. **ErrorTable:** A reference to a table which will contain the list of errors and warnings occurred during the enrichment/validation step. This table has the following structure:

|  |  |
| --- | --- |
| Column | Description |
| ROW\_ID | Id which univocally identifies a row of the *inputTable* (not in this table)  This is used to identify the whole set of errors related to a single row. |
| ERR\_CODE | A code identifying the current error |
| ERR\_TYPE | Type of error, as “warning” or “error”. |
| ERR\_MESSAGE | The error message explaining the issue |
| ERR\_COLUMN | The name of the column of the *inputTable* involved in the error. If multiple columns are involved in the same error, then one row per column is added to this table. In particular, each row will have the same ROW\_ID, ERR\_CODE, ERR\_TYPE, ERR\_MESSAGE but different ERR\_COLUMN and ERR\_VALUE. |
| ERR\_VALUE | The value of the column specified in ERR\_COLUMN |

1. **IdColumns:** A list of columns which identify a row in the InputTable. This is needed to univocally identify inside the error table the rows affected by the issues.
2. **[Additional parameters depending on the requested actions]:** TheDEAV-specific set of parameters required to make the procedure work properly.

At the end of the process, the required additional columns will be added to the input table. If errors/warnings were detected (either by the enrichment step or by the validation), then they will be inserted into the table specified in the second input parameter.

For example, the FoodEx2 validation procedure needs as specific parameter the name of the column of the input dataset which contains the FoodEx2 codes to validate. Therefore an example of its use would be:

%validateFoodex2(inputTable, errorTable, idColumns, foodex2Column);

For the complete list of available DEAV, check the documentation.

1. Enrichment/validation for Business Rules

The Business Rules Engine (BRE) needs more and more sophisticated checks, in order to perform a suitable and useful validation of the reported data. This leads to extend the engine in order to accommodate new validations which have a complex structure. In particular, some validations steps of the engine require having:

* External/computed additional information, in order to assess the correctness of the data more precisely.
* The possibility to delegate the validation to a less structured system if the validation is too complex to be managed by the engine (e.g. a check which cannot be expressed by the engine syntax).

The enrichment and validation step achieves these two points in a single procedure callable directly within the Business Rules Engine. More precisely, the DEAV procedure is wrapped inside another process whose implementation is Business Rules specific. In particular, the process flow is the following:

1. The BRE calls the DEAV\_ENRICH\_AND\_VALIDATE macro specifying:
   * The required enrichment/validation procedure;
   * The input dataset (name of the dataset which contains the data to process);
   * The error dataset (name of the dataset which will contain the errors);
   * The additional DEAV-specific parameters;
   * The name of the column inside the input dataset which contains the DCF id;
   * The name of the column inside the input dataset which contains the dataset id.
2. The specific DEAV procedure is then called by the DEAV\_ENRICH\_AND\_VALIDATE macro passing as idColumns the DCF id and the dataset id.
3. If errors/warnings occurred in the DEAV process/validation, then the error dataset is converted to a format which is suitable for the business rules. More precisely, the FAILED\_RECORDS table will be created.
4. If the DEAV step added new columns to the input dataset, then the columns are transposed and inserted into an Oracle table (ODS.DEAV\_ENRICHMENT\_TABLE) in order to be usable inside the ETL processes. The structure of the table is the same as the structure of the tables with suffix \_A contained in the DCFDATA library. In particular, the structure of the Oracle table is the following:

|  |  |
| --- | --- |
| Column | Description |
| ID | Row Id (primary key) |
| F\_ID | Dcf id |
| F\_C\_NAME | Name of the enriched column |
| ATTR\_NAME | Attribute name (for compound values, otherwise empty) |
| VALUE | Value of the column for single cardinality columns, otherwise value of the attribute |
| SORTING | Order of application of the attributes |
| DATASET\_ID | The id of the dataset |

**Important note:** The procedure does not support yet enrichment procedures to create compound fields in Oracle, because it would be needed to split the repeated values into multiple rows (it is a transposed table). In case of need, it will be necessary to edit the source code of the DEAV\_ENRICH\_AND\_VALIDATE macro stored in MSTORE. The source code is available on GitHub at the following link:

<https://github.com/openefsa/SAS-stored-functions-and-macros/blob/master/data-enrichment-and-validation/src/storedDeavInterface.sas>

This means that the current version of the algorithm will always put an empty value for the ATTR\_NAME column and a value of 1 for the SORTING column.