1 Business language

A business language rule (called BLR from now on) is a way to express a business rule by using a natural language like expression.
The target is to generate a JSON file that can be stored into a database and translated into a set of commands to a business rule engine and to provide a rendering engine in order to let a non-technical user to read and manage a set of rules.

The business language provides an implementation the following concepts that are used to compose the rule:

- A "dictionary" of words
- A grammar to construct the phrase
- A data structure to store the rules

1.1 Dictionary and roles:

A condition or action line is composed by different words combined using the blr "grammar".
We can identify three basic word **types** in a blr:

- **Verbs**
- **Nouns**
- **Conjunctions**

In order to further specialize the blr elements a **role** is also assigned. Where the type defines what I want to do the role defines how and where the element is used.
For this reason in the datamodel the fields will be always indicated with the combined key element+role.

1.2 Rule

At root level a blr rule is composed by a name, a list of conditions called "when" (or condition line) that contains all the checks to verify and a list of actions to execute called "then" (or action line).

```
{
        name: "name of the rule".
        when: [],
        then: []
}
```

Field	Type	Description
name	String	Name of the rule
when	Array<conditionConjuction>	List of condition groups to validate
Then	Array<actionConjuction>	List of action groups to validate

1.3 VERBS

An action that the engine must take (e.g. GET, EXISTS, COMPARE, SET). Can be used either into a condition in order to validate whether a condition line is true or into an action in order to change the context of execution of the rule.
Usually, a verb definition also requires one or more nouns as input in order to work.

1.3.1 Verbs data

To work most of the verbs (there are cases of nodata verbs like the “always valid” verb) requires context data to be ingested. This data can come either from the context of execution of the rule (e.g. the account) or from the instantiation of “variables” coming, for example, from FOR cycles.
The data structure is defined as follow:

```
{
    "value": "_1_account.ateco_category_ids"
},
{
    "isValue": true,
    "value": "123"
}
```

Field	Type	Description
value	String	Value of the data. By default the system will interprets the value as coming from a define or a variable.
isValue	Boolean	Optional. If true then the standard behaviour is overridden and the data Is treated as a manually inputted data not coming from a variable
alias	string	Required only for a verbNoun. Used to access the output of the verb as a variable in subsequent child verbs (see verbnoun section for more information).

1.3.2 Verbs Roles

1.3.2.1 prefix

A prefix verb is used in a condition line before the definition of the condition criteria and is used to indicate the noun that we want to use inside an action. The only prefix verbs available is “GET *define noun*”

```
{
    "role": "prefix",
    "type": "verb",
    "value": "get",
    "data": [],
    "define": {}
```

```
}
```

Field	Type	Description
role	String	
type	String	
value	String<”get”,”count”,”not exists”>	Type of prefix
define	nounDefine	The noun that I want to get with the condition line. The noun can then be referenced in the action or in other condition lines.
data	Array<data>	Only when the value is “count” data contains the operator and the operator value of the count (e.g. more than 1, less than 5, etc…)

1.3.2.1.1 prefix values

get

retrieve the specified noun define and save the result into a variable (see “varIndex” in the define nouns section) that can be used in subsequent conditions or actions

count

count the occurrence of the nouns. This is the only prefix verb that allows the use of the “data” array. In case of count two array values are allowed:

- Position 0: count operator (<,>,<=,>=,!=)
- Position 1: integer with count value check

The count prefix does not save any variable

not exists

Validate the condition line only if there are no facts that satisfy the condition.
No variable is saved.

1.3.2.2 Criteria

Criteria are verbs that can be used inside a condition or action line to define a check to pass to continue with the rule execution.
The list of criteria is not fixed and can be extended with new verbs and can be retrieved by using the “/api/pylon/v1/businesslanguage/verbs” web service under the “criteriaVerbs” attribute in the response.

```
{
    "data": [
        {
```

```
            "value": "_1_account.ateco_category_ids"
        },
        {
            "isValue": true,
            "value": "123"
        }
    ],
    "role": "criteria",
    "type": "verb",
    "value": "equal",
}
```

Field	Type	Description
role	String	
type	String	
value	String	Unique name of the verb. If the verb is not supported an error will be raised during the synch process
data	Data	Data to use as input of the verb

1.3.2.3 Function

Function verbs are used to execute an action on the current execution context, for example, all the basic CRUD operations (modify, delete, create, upsert).
The functions available can be retrieved by using the "/api/pylon/v1/businesslanguage/verbs" method under the "functionVerbs" attribute in the response.

```
{
    "data": [
        {
            "value": "attr.name"
        },
        {
            "isValue": true,
            "value": "13"
        }
    ],
    "role": "function",
    "type": "verb",
    "value": "modify"
}
```

Field	Type	Description
role	String	
type	String	
value	String	Unique name of the verb. If the verb is not supported an error will be raised during the synch process
data	Data	Data to use as input of the verb

1.3.2.4 Noun

With this role a verb is not checking or modifying the context but instead is generating a new set of nouns that can be used in by the actions that follow.

An example of a noun verb is the “forEach” command that get an array as input and cycle it.

```
{
    "data": [
        {
            "alias": "attr",
            "value": "fam.attributes"
        }
        "role": "noun",
        "type": "verb",
        "value": "forEach"
 }
```

In this case I will cycle the fam.attributes array, creating a noun called “attr” for each iteration.

Field	Type	Description
role	String	
type	String	
value	String	Unique name of the verb. If the verb is not supported an error will be raised during the synch process
data	Data	Data to use as input of the verb. In case of a noun verb the “alias” attribute is required and will be used to access the output of the verb as a variable in subsequent child verbs.

1.4 Nouns

Express everything that can be retrieved with a condition or manipulated with an action.

1.4.1 Nouns roles

1.4.1.1 Define

A noun can have only one role that is “define”. A define is a data structure that the rule can access or manipulate. From the define is possible to know:

- The objects available (e.g. Account, Contact, Cartitem, etc..)
- From each object the available fields (e.g. Account.name, Contact.address, etc..)
- In case of a hierarchy the field that is used to solve the parent/child relationship (e.g. the list of attributes that can be configured is stored on the Family.attributes fields)

The list of nouns can be retrieved using the “/api/pylon/v1/businesslanguage/nouns” service.

```
{
    "role": "define",
    "type": "noun",
    "value": "cartitem",
```

```
    "varIndex": 1,
    "criteria": []
}

{
    "role": "define",
    "type": "noun",
    "value": "family"
    "from": "_1_cartitem.families",
    "varIndex": 2,
    "criteria": []
}
```

Field	Type	Description
role	String	
type	String	
value	String	Name of the define to use from this value depends on the list of fields available for the criteria
from	string	Some defines are not available directly, but they exist only as children of other defines. For example, a family exists as a child of a cartitem. The “from” attribute contains the name of the attribute on the parent define that contains the children.
varIndex	Integer	varIndex is an integer used to generate the name of the variable that can be used in subsequent criteria and verbs (both action and condition). For example, a varIndex=1 and value=cartitem will generate a _1_cartitem variable name. For this reason, the combination of value+varIndex must be a unique key in the rule.
Criteria	Array<verbCriteria | criteriaConjunction>	The logic to be validated to retrieve the data from the execution context. If the check fails and the conditionConjuction is “and” the rule is not executed

1.5 Conjunctions

Conjunctions are used to connect a group of two or more elements inside the blr rule, usually two verbs with the same role or a verb with a conjunction in order to generate complex logic (verb1 AND verb2).

1.5.1 Conjunctions roles

1.5.1.1 Condition

Connects different condition lines (see verbs-prefix), is used to specify if a group of condition lines must be in AND or in OR.

```
{
    "role": "condition",
    "type": "conjunction",
    "value": "and",
    "conditions": []
}
```

Field	Type	Description
role	String	
type	String	
value	String <“and” | “or”>	Type of conjunctions only AND/OR is supported.
conditions	Array<verbPrefix>	List of prefix verbs

1.5.1.2 Action

Connects different action lines (see conjunction-enabler)

```
{
    "role": "action",
    "type": "conjunction",
    "enablers": []
}
```

Field	Type	Description
role	String	
type	String	
enablers	Array<conjuctionEnabler>	List of enablers for the action group

1.5.1.3 Enabler

Enabler conjunctions relates a set of actions to be executed to a list of criteria conditions to pass.
This can be useful to limit the number of rules by aggregating common conditions and defining different behaviour based on the context directly in the action.

You can define multiple sub-enablers from an action conjunction of from a function verb, but it is advised to avoid going more than three levels deep in order to keep the rule readable.

```
{
    "criteria": [],
    "role": "enabler",
    "type": "conjunction",
    "verbs": []
}
```

Field	Type	Description
role	String	
type	String	
Criteria	Array<verbCriteria | criteriaConjunction>	The logic to be validated to execute the related verbs. If the check fails only the related verbs are not executed.
verbs	Array<verbFunction | verbnoun>	

1.5.1.4 Criteria

Used to define the entry criteria inside a single condition, for example defining a logical operator between two verbs (AND/OR) or by defining groupings in the criteria with parentheses.

Field	Type	Description
role	String	
type	String	
Value	String<"(","")","and","or">	Logical grouping or operators

1.6 NL Result

Even if not required for the generation of the rule, storing the rule in a natural language is useful for readability and searching purposes.
Also is a good way to ensure feedback to the user and to an AI system in order to establish a NL<->BLRJSON conversion engine.
The NL result is not managed by the conversion engine but will be automatically generated by the manager tool.

Verify the following conditions:

- **If all of the following conditions are true:**
    - 1. **Get** all the **account**

Execute the following actions:

- If always
    - change the **1.account.ateco_category_ids** field to **1.account.current_ids**

check the "examples" section for a list of requirement/nl/blrjson