Transformation Rule Specification Language

This document presents the Extended Backus-Naur Form (EBNF) of the transformation rule specification language.

```
<transformation_specification> ::= <configuration_element> <transformation_rule>+
    <condition_element>* <search_element>*
```

Listing 1 – General Structure

```
<configuration_element> ::= 'config_element ' <config_body>
<config_body> ::= '{' <body_specification> '}'

<body_specification> ::= <baseIRI_spec> (',' <syntax_spec>)? (',' <header_spec>)?
    (',' <namespace_spec>)

<baseIRI_spec> ::= '"default_baseIRI" = "' string '"'

<syntax_spec> ::= '"export_syntax" =' <syntax>
<syntax> ::= '"Turtle"' | '"RDF/XML"' | '"N-Triples"'

<header_spec> ::= '"has_non_processable_header" = ' <boolean>
<boolean> ::= '"true"' | '"false"'

<namespace_spec> ::= '"namespace" = "' <namespace> '" refers_to ' <IRI>
<namespace> ::= string
<IRI> ::= '"' string '"'
```

Listing 2 – Configuration Element

```
<transformation_rule> ::= <rule_header> '{' <rule_body> '}'

<rule_header> ::= <rule_type>' '<rule_identifier>' '<subject_type>
<rule_type> ::= 'column_based_rule' | 'row_based_rule'
<rule_identifier> ::= string

<subject_type> ::= '[' ( (<SSD_column_header> <flag>*) |
    (<SSD_column_header> <flag>* '/;') )+ 'is_equivalent_to' <ontology_class> ']'
<SSD_column_header> ::= '"' string '"'
<ontology_class> ::= ('"' string '"') | ('"' <namespace> ':' <string> '"')

<rule_body> ::= <rule_statement> (',' <rule_statement> )*
<rule_statement> ::= 'links_to ' <triple_object> ('/;' <triple_object>)*
    ' using ' <triple_predicate>
<triple_object> ::= (<SSD_column_header> | <rule_identifier>) <flag>*
<triple_predicate> ::= '"' string '"'
```

Listing 3 – Transformation Rule

```
<condition_element> ::= 'condition_element ' <condition_element_identifier>
    <condition_body>
<condition_element_identifier> ::= string

<condition_body> ::= '{' <condition_statement> (',' <condition_statement>)* '}'
<condition_statement> ::= <SSD_column_header> <operator> <value>
<operator> ::= '==' | '!=' | '<' | '<=' | '>' | '>='
<value> ::= '"' string '"'
```

Listing 4 – Condition Element

```
<search_element> ::= <search_header> '{$' <sparql_query> '$}'

<search_header> ::= 'search_element' <search_element_identifier> <search_endpoint>
<search_element_identifier> ::= string
<search_endpoint> ::= '["' <URI> '"]'
<URI> ::= '"' string '"'

<sparql_query> :: string
```

Listing 5 – Search Element

```
<flag> ::= <defined_flag> (' ' <defined_flag>)+
<defined_flag> ::= <NM_flag> | <SP_flag> | <BaseIRI_flag> | <FX_flag> | <DT_flag>
    |
    <CustomID_flag> | <Col_flag> | <SE_flag> | <Condition_flag> | <NODE_flag>

<NM_flag> ::= '/NM'

<SP_flag> ::= '/SP(' <character> (',' <position>)? ')'
<character> ::= '"' string '"'
<position> ::= integer

<BaseIRI_flag> ::= '/BaseIRI(' <IRI> ',' <namespace> ')'
<namespace> ::= '"' string '"'

<DefaultValue_flag> ::= '/DefaultValue(' <content> ')'
<content> ::= '"' string '"'

<DT_flag> ::= '/DT(' <XSD_datatype_name> ')'
<XSD_datatype_name> ::= '"' string '"'

<CustomID_flag> ::= '/ID("' string '")'

<Col_flag> ::= '/COL(' <filename> ',' <column_number> ')'
<filename> ::= '"' string '"'
<column_number> ::= integer

<SE_flag> ::= '/SE(' <search_element_identifier>, <SPARQL_variable> ')'
<SPARQL_variable> ::= string

<Condition_flag> ::= '/CE(' <condition_element_identifier> ')'

<NODE_flag> ::= '/NODE(' <ontology_class> ')'
```

Listing 6 – Flags