Description

OCL statement is the part of the rule containing a formal description of a business rule, defined in OCL language.

Following items are related to OCL statement in XMLdation service:

- Context - defines which part of the XML message is being checked by the OCL statement. simpleType or complexType
- Property within context (e.g. XML element, attribute)
- Manipulating qualities of a property (e.g. arithmetic operations, comparisons)
- Logical operators between sections in the statement (and, or, xor, not, implies)

Boolean returns and OCL statement example

Rule is written so that it matches the business rule it is depicting.

For example, when a business rule is:

> Debtor name must be "The initiator"

OCL-restriction is written in same form, so that the statement depicts a valid case, and thus valid case will not give an error in the validation phase. Error is only returned when the check does not pass.

Context:	HeaderType1
OCL:	self.Debtor.Name = "The initiator"
Description:	Rule defines a specific value for Debtor/ Name.
Message	Debtor / Name is not The initiator

This rule locates element Debtor / Name under HeaderType1 and compares it's value to another value.

Property within context

For reference, below is an image representation of type *HeaderType1.* Note that context of the above example rule is not PartyIdentification nor Max140Text. HeaderType1 is the first unique mandatory type present in XML for referencing Debtor content.

In OCL statement, keyword "self" is used to refer the type defined in the context, *Header* in this case. Then dot notation is used to traverse the tree to locate the specific element.

Schema element	Type	Occ.
Message	Message	1...1
Header	HeaderType1	1...1
Id	Max35Text	1...1
TimeStamp	DateTime	1...1
ControlSum	decimal	1...1
NumberOfTransactions	integer	1...1
Debtor	PartyIdentification	0...1
Name	Max140Text	0...1

Manipulating qualities of property

In the example above, property's (self.Debtor.Name) value is compared against another string, depicted inside quotation marks. Notation used to depict comparisons are: =, <, >, =>, =<, <>. Explanations for these can be found in comparison wiki page.

Arithmetic operators may be used when applicable as well.

Logical operators

Within the statement, logical operators (and, or, xor, not) may be used to alter the behaviour of the rule or to present a condition which has to pass before something else is checked (implies).

Examples

Let's have a more thorough look at the business rule depicted above, 'Debtor name must be "The initiator"'. This business rule looks simple, but it contains two different rules. Following example scenarios are possible to be given:

The XML snippet below would the pass check.

```
<Header>
  <Id>1.0</Id>
  <TimeStamp>2015-07-03T12:17:50</TimeStamp>
  <ControlSum>2</ControlSum>
  <NumberOfTransactions>1</NumberOfTransactions>
  <Debtor>
    <Name>The initiator</Name>
  </Debtor>
</Header>
```

However, there are two different possible cases which would not pass the rule

```
<Header>
  <Id>1.0</Id>
  <TimeStamp>2015-07-03T12:17:50</TimeStamp>
  <ControlSum>2</ControlSum>
  <NumberOfTransactions>1</NumberOfTransactions>
</Header>
```

```
<Header>
  <Id>1.0</Id>
  <TimeStamp>2015-07-03T12:17:50</TimeStamp>
  <ControlSum>2</ControlSum>
  <NumberOfTransactions>1</NumberOfTransactions>
  <Debtor>
    <Name>Something else</Name>
  </Debtor>
</Header>
```

Two cases being:

- Debtor name is not present at all
- Debtor name is something else than defined in the business rule

To separate these two different cases it makes sense to create two different OCL-rules. Creating multiple rules instead of one first sounds like making things more complicated, but it's rather vice versa. There are multiple benefits from this: each individual rule will be simple and clear, there are no hidden behavior inside any of the individual rules, error messaging in the validation can be made very accurate, documentation of the rules also becomes very accurate. So in this case we need these two OCL-rules:

- Debtor name is mandatory
- When Debtor name is given, its value must be "The initiator"

Context:	HeaderType1
OCL:	self.Debtor.Name->size() = 1
Description:	Rule mandates Debtor name

Context:	HeaderType1
OCL:	self.Debtor.Name->size() = 1 implies self.Debtor.Name = "The initiator"
OCL-Query	self.Debtor.Name
Description:	Rule restricts the value of Dbtr Nm when it is given

Note that in the second rule, OCL query is defined as well.

In addition, the schema contains two occurrences of Debtor, under Header and under Transaction. In this example an assumption is made that checking the Debtor under Header is enough.

Collections

Elements which have maximum multiplicity higher than 1 need a more detailed way to access them. As the multiplicity is higher than one, OCL code has to tell which element and in what conditions specifically is relevant for the rule.

Description and usage of collections in ocl is described in the collections wiki page.