Purpose

Purpose if this sections is to introduce concept of collections and to state why they need extra handling.

Collections are anything where the amount item we are looking at has to possibility of being more than 1. For example, element with an occurrence of 1..1 does not need to be handled as a collection (although nothing prevents us), but element with occurrence of 0..n requires us to treat it as a collection

Desription

Collections within Studio are explained in the following page: https://wiki.xmldation.com/User_guide_for_OCL_rules_in_XMLdation_environment/5._Collections

Before going to the tasks, it is a good idea to glimpse over the above page. However, doing these tasks do not explicitly require complete understanding of above page.

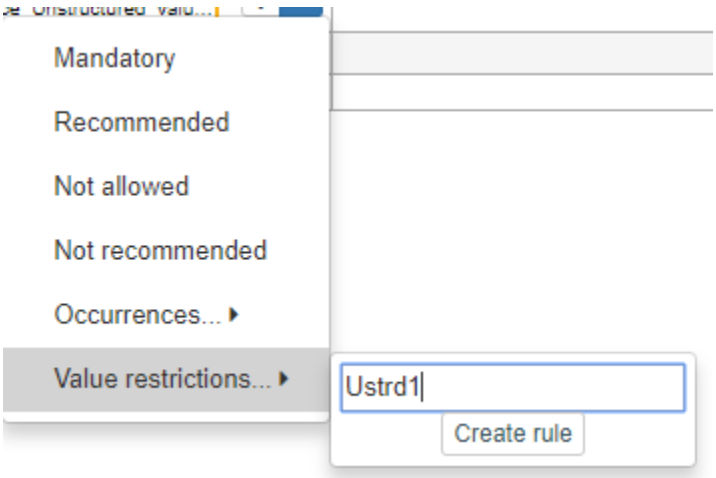
In this section, we shall be going two tasks

1. Creating a rule which requires us to restrict a values for Unstructured
2. Testing

Instructions

1: Creating a rule which requires us to restrict a values for Unstructured to either "Ustrd1" or "Ustrd2"

We can use the automatical rule generation to create basis for this, by finding Unstructured from the schema and selecting value restriction from the dropdown menu. We can first insert the value "Ustrd1" into the input field



When opening the code, following code should be present in the definition:

self.Unstructured->forAll(q | q.matches("Ustrd1"))

(however, as of writing this there is a bug present and no ->forAll() exists. If this is the case, the above

snippet should be copied into the definition)

Having a further look at the element and the code, we can see the following:

Remittance	RemittanceType	0...1
Unstructured	Max140Text	0...*
Structured	PartyIdentification	0...*
Name	Max140Text	0...1

Because Unstructured is stated to be 0..n, there may be more than 1 Unstructured given in the XML file according to schema. For this reason, the generated code includes forAll() syntax. This means that for each of the Unstrucuted found in the XML file, the code within forAll() brackets is executed. We have two different sections within the bracket, separated by a pipe character | . Left side of the char introduces a variable we can use when referring to an individual element inside a collection. Right side contains the expression we want to execute for each individual item.

self.Unstructured->forAll(q | q.matches("Ustrd1"))

Therefore, for changing this validation rule to the purpose we want (either Ustrd1 or Ustrd2 are valid codes), we must do the following

self.Unstructured->forAll(q | q.matches("Ustrd1") or q.matches("Ustrd2"))

2: Testing

Above rule should now be tested. Worth noting is that rule message will be confusing unless changed (as we were modifying a pre-existing rule) and error message is not the best possible.

In next section, we shall focus on improving the error message location.