Purpose

Purpose of this section is to introduce the feedback Studio returns when OCL code is invalid. And to realise that it is always a good thing when there is a build error in the code, as this prevents invalid validation rules being visible in normal validation users are able to access.

Description

Following steps are done here:

1. Modifying OCL in an existing validation rule to be erroneous
2. Compiling and investigating erroneous input
3. Further build errors

Instructions

1: Modifying OCL in an existing validation rule to be erroneous

We can modify the rule we have made in section 1. When opening the rule, we should see the following definition:

self.Id.matches("Id1") or self.Id.matches("Id2")

We can intentionally break this by breaking the syntax.

Lets change the code by removing the last bracket

self.Id.matches("Id1") or self.Id.matches("Id2"

2: Compiling and investigating erroneous input

When rebuilding the project, we should be receiving an error. To access the error message, we'll have to find the rule which has been highlighted with red and to open it. Technical error message is displayed there, which will indicate what has happened and why the compilation was not successful. In this case the actual error message is:

The syntax of the ocl expression is invalid. Problem detected at '(' Location: line 1. Position 41

Compile Error with rule Transaction_Id_valueRestrictions. Error found on line '1'. no viable alternative at input '<EOF>'

These may sound technical and confusing at start, but it usually helps to navigate to the position indicated in the message and paying attention to the characters stated to be erronoeus. Here character (is stated, which should help us to realise that the ending bracket is missing.

3: Further build errors

We should also investigate other erroneous cases. At least following OCL and their corresponding error message should be checked:

self.Id.matches("Id1") or self.Id.matches("Id2)

self.Id.matches("Id1") self.Id.matches("Id2")

self.Id.matches("Id1") orself.Id.matches("Id2")

self.Id.matches("Id1") or self.matches("Id2")

self.Id.matches("Id1") nor self.Id.matches("Id2")

We can also change the context to be something else instead of TransactionType1. Example of another context in the schema is: PartyIdentification