goAML 5.0 – XSD 1.1 ASSERT-restrictions to be applied

We aimed to create assert rules that would improve the quality of the reports without becoming prohibitive or making it too hard to fill out the web form.

NB: Assert rules are not checked while filling out the web form, but are immediately checked when the report is SUBMITTED. If goAML detects any errors, the submission will fail. The user then has the opportunity to make the necessary changes and retry the submission.

As we wait for the goAML 5.0.2 version to be published by UNODC and tested, this document explains the XSD elements that will be included in the version of the XSD compatible with goAML 5.0.2.

Asserts will be present in different parts of the XSD.

<u>The first Assert statements will be present under the ComplexType of element “report”:</u>

REJ-GEN01: Reason field must not be empty

If a reason element is present, it should not be blank.

Remark : in the goAML webforms, if left empty, the reason element will not be included in the internal XML that goAML will generate. REJ-GEN02 will apply in this case.

Assert statement:

```
<xs:assert test="if (boolean(reason)) then ((reason[text()]) ne '')
else true()" xerces:message="REJ-GEN01: Reason must not be empty" />
```

REJ-GEN02: Reason must be indicated

A reason (motivation for submitting a report) must be present. If the detailed reason is given in an attached document, at least this should be stated here – better yet if there is a short description.

Only RIRA/RIRT report types are exempt from this rule, since the reason for your submission is a request on our part (or the fact that all relevant transactions regarding your STR exceeded the maximum of 500).

Assert statement:

```
<xs:assert test="if (not(boolean(reason))) then ((report_code eq
'RIRA') or (report_code eq 'RIRT')) else true()" xerces:message="REJ-
```

```
GEN02: Reason must be indicated, except if the report type is RIRA or
RIRT" />
```

REJ-IND01: Report indicators mandatory except for RIRA/RIRT reports

Indicators must be indicated, except for RIRA and RIRT for the same reason as in the previous section.

```
<xs:assert test="if (not(boolean(report_indicators))) then
((report_code eq 'RIRA') or (report_code eq 'RIRT')) else true()"
xerces:message="REJ-IND01: Indicators must be included, except if the
report type is RIRA or RIRT" />
```

REJ-TRS01: Each transaction number in the report must be unique

```
<xs:assert test="count(transaction/transactionnumber) eq
count(distinct-values(transaction/transactionnumber))"
xerces:message="REJ-TRS01: Each transaction number in the report must
be unique" />
```

REJ-TRS02: The limit of transactions per report is set to 500

```
<xs:assert test="count(transaction) lt 501" xerces:message="REJ-TRS02:
The limit of transactions per report is set to 500" />
```

REJ-ENT01: Only one level of entity-entity relationships are allowed

```
<xs:assert test="empty(entity/related_entities)" xerces:message="REJ-
ENT01: Entity-To-Entity relation supported in schema 5.0 should allow
child entity only up to one level, and not recursively. An entity can
not be linked to an entity that is itself already linked to an entity."
/>
```

<u>Under the complexType of t_account_my_client as well as under t_account:</u>

REJ-IBAN01: an IBAN number is not correctly formatted

Remark: We want to ensure that IBAN account numbers are correctly filled out, since this is one of our merge criterions.

```
<xs:assert test="if (iban) then (matches(iban , '^([A-Z]{2}\d{2}[A-Z0-
9]{10,30})$')) else true()" xerces:message="REJ-IBAN01: an IBAN number
is not correctly formatted" />
```

REJ-ACC01: If IBAN present, then the account number must be identical to the IBAN

```
    <xs:assert test="if (iban) then (iban eq account) else true()"
xerces:message="REJ-ACC01: If an IBAN number is present, the account number
must be identical to the IBAN" />
```

REJ-ACC02: If an IBAN number is present, then a swift code should be provided

```
<xs:assert test="if (iban) then (swift) else true()"
xerces:message="REJ-ACC02: If an IBAN number is present, then a swift
code should be provided, not an institution_code." />
```

REJ-BIC01: BIC/SWIFT code is not correctly formatted

The SWIFT or BIC code has the form of 8 or 11 characters:

- 4 alphanum (business party prefix),
- 2 alpha (country code)
- 2 alphanum (business party suffix)
- optionally followed by 3 characters specifying a branch. An 'XXX'-branch (being the head office and not a branch) can be omitted.

```
<xs:assert test="if (swift) then (matches(swift , '^([A-Z0-9]{4}[A-
Z]{2}[A-Z0-9]{2}|[A-Z0-9]{4}[A-Z]{2}[A-Z0-9]{5})$')) else true()"
xerces:message="REJ-BIC01: BIC/SWIFT code is not correctly formatted"
/>
```

REJ-ACC03: If the SWIFT is 11 chars, branch must be filled out

If the SWIFT is 11 chars long (thus, indicates a branch), branch must be filled out and contain the same 11 char-SWIFT.

```
<xs:assert test="if (string-length(swift) = 11) then (branch eq swift)
else true()" xerces:message="REJ-ACC03: If the SWIFT is 11 chars long
(thus, indicates a branch), branch must be filled out and contain the
same 11 char-SWIFT" />
```

REJ-ACC04: If an institution_code is indicated, the institution_name should be omitted

Institution names, because of the variety of spellings, hinder the automatic merge process integrated in goAML and should therefore be omitted if a banking institution is identified by its institution (or Swift) code.

```
<xs:assert test="if (string-length(institution_code) gt 5 ) then
((institution_name eq institution_code) or (not(institution_name)) else
true()" xerces:message="REJ-ACC04: If an institution_code is indicated,
the institution_name should be omitted" />
```

REJ-ACC05: If a balance is given, the account currency and the date of the balance must also be indicated

Assert to be defined

Under ComplexType: t_entity_my_client as well as under t_entity

REJ-ACC01: Lux. incorporation number (numéro RCS) not correctly formatted

The RCS number should be like B123456 or F123, without padding zeroes and without spaces or other symbols.

```
<xs:assert test="if ((incorporation_country_code = 'LU') and
boolean(incorporation_number)) then (matches (incorporation_number,
'([A-Z]{1}[1-9]\d*)$')) else true()" xerces:message="REJ-ACC01: Lux.
incorporation number (numéro RCS) not correctly formatted. Must be like
B123456 or F123, no padding zeroes, no spaces." />
```

Under complexType Transaction:

REJ-TRS03: if the transaction is a Transfer, then both sides of the transfer must be accounts.

Assert to be defined

Python validation script (optional)

Since the tools we used so far to validate our XML files, do not meet the requirements for XSD 1.1 validation, here is a short python script we used to test our assert statements:

```
import xmlschema

schema = xmlschema.XMLSchema11('the path to the goaml5.0-XSD file.xsd')

print ('Schema OK')

try :

    schema.validate('The path to the report file to be tested.xml')

except xmlschema.validators.exceptions.XMLSchemaValidationError as inst:

    print ('Failed Reason: ')

    print (inst.reason)

    print (inst.validator)

    print (inst)

except:

    print ('Other error')

else :

    print('Validation OK')

print ('Validation finished')
```

The advantage of using this script over using the goAML Validation tool on goAML Web in its current version is that if your XML fails an assertion test, the printout will tell you which element of the XML caused the error.

You may of course use any tool/script you want to check your XML files with before submitting.