

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.

For submitted xml files, you need to go to the Submitted Reports menu and the Status column will indicate Failed Validation if there are assert violations. The details thereof can be found by clicking on the Failed Validation hyperlink.

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.

The first assert statements will be present under complexType transaction:

REJ-TRS03: If the transaction type is Transfer, then both sides of the transaction must be accounts

```
<xs:assert test="(if
(concat(t_from/from_funds_code,t_from_my_client/from_funds_code) eq
'Q') then ((t_from/from_account) or (t_from_my_client/from_account))
else true())
and (if (concat(t_to/to_funds_code,t_to_my_client/to_funds_code) eq
'Q') then ((t_to/to_account) or (t_to_my_client/to_account)) else
true())"
```

REJ-TRS04: Transaction type (funds_code) must be the same for 'from' and 'to' sides

```
<xs:assert
test="concat(t_from/from_funds_code,t_from_my_client/from_funds_code)
eq concat(t_to/to_funds_code,t_to_my_client/to_funds_code)"
```

Under complexType report:

REJ-REA01: Reason 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.

```
<xs:assert test="if (boolean(reason)) then reason/node() else true()"
```

REJ-REA02: Reason must be indicated, except if the report type is RIRA or RIRT

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()"
```

REJ-IND01: Indicators must be included, except if the report type is RIRA or RIRT

Report 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()"
```

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

```
<xs:assert test="count(transaction/transactionnumber) eq count(distinct-
values(transaction/transactionnumber))"
```

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

```
<xs:assert test="count(transaction) lt 501"
```

Under complexType t_account my_client as well as t_account:

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()"
```

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

```
<xs:assert test="if (iban) then (iban eq account) else true()"
```

REJ-ACC02: If an IBAN number is present, then a BIC/SWIFT code should be provided, not an institution_code

```
<xs:assert test="if (iban) then (swift) else true()"
```

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()"
```

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

```
<xs:assert test="if (string-length(swift) = 11) then (branch eq swift) else true()"
```

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()"
```

REJ-ACC05: If an account balance is indicated, the account currency must be specified

```
<xs:assert test="if (balance) then (currency_code) else true()"
```

NB : The date of the balance is required by a minOccurs 1 in the corresponding sequence.

REJ-ACC06: Branch should only be indicated for accounts that have a SWIFT with a length of 11 characters

```
<xs:assert test="if (branch) then ((string-length(swift) = 11) or not(swift)) else true()"
```

Under complexType t_entity my_client as well as t_entity:

REJ-ENT02: Lux. incorporation number (numéro RCS) not correctly formatted. Must be like B123456 or F123, no padding zeroes, no spaces

```
<xs:assert test="if ((incorporation_country_code = 'LU') and boolean(incorporation_number)) then (matches (incorporation_number, '([A-Z]{1}[1-9]\d*)$')) else true()"
```

Under ComplexType entity related entity type:

RJ-ENT01: An entity cannot be linked to an entity that is itself already linked to an entity

```
<xs:assert test="empty(entity/related_entities)"
```

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.