

A424-BASED VALIDATION RULES FOR AIXM AND NDBX DATASETS CONFIGURATION & IMPROVEMENTS OF ARC TOOL

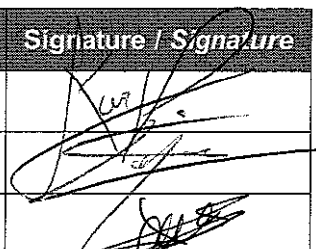
Résumé / Synopsis :

ARC (AIXM Rule Checker) Web is a tool provided by EUROCONTROL and used during the validation process of AIXM data. It provides means to perform a semantic validation of XML data, in addition to the standard syntactic XML validation against an XSD Schema. These business rules defined for the semantic validation are formulated in two contexts, the natural language understandable by human people and the Schematron language, executable by computer on XML file.

In the frame of the development of new Business rules based on the Arinc424-19 standard for EUROCONTROL, AEROCONSEIL implemented some improvements of ARC tool. The objective of this technical note is to document and justify these improvements, and to propose additional upgrades which could be implemented in ARC in the future.

Mots Clés / Key Words :

Schematron, ARC improvements

	Nom / Name	Date / Date	Signature / Signature
Rédigé par / Written by	EURIOLOGIC for AEROCONSEIL	30/04/2009	
Vérifié par / Checked by	Hubert LEPORI	30/04/2009	
Approuvé par / Approved by	Jean-Yves MORELL	30/04/2009	

Client / Customer : EUROCONTROL

Référence externe / External reference :
Cahier des Charges / Statement of Work : 08-112533-Q


Modifications / Revisions

Edition / Issue Date / Date	Réf. fiche de relecture / Proof reading form ref.	Page / Page	Objet des évolutions / Reasons for revision
Ed 1 30/04/2009	N/A	All	Creation

Diffusion / Distribution

Société / Company	Département / Department	Nom / Name	Papier / Paper	Elec. / Digital
Diffusion interne / Internal distribution				
AEROCONSEIL		Archivage - Sharepoint		X
	FMS-D	MORELL Jean-Yves		X
	FMS-D	BRUNEAUX Philippe		X
	FMS-D	RIFFLART Pierre-François		X
	FMS-D	LEPORI Hubert		
EURIOLOGIC		KARP Paul		X
Diffusion externe / External distribution				
EUROCONTROL		POROSNICU Eduard		X
		WILSON Scott		X



TABLE OF CONTENTS

REFERENCES	4
ABRÉVIATIONS / ACRONYMS	4
INTRODUCTION	5
1. ANALYSIS OF THE ORIGINAL VERSION OF ARC WEB	6
1.1. ARCHITECTURE OF ARC WEB	6
1.1.1. Web user interface	6
1.1.2. Execution engine	7
1.2. PROJECT CONFIGURATION	10
2. AEROCONSEIL IMPROVEMENTS OF ARC	11
2.1. OBJECTIVES	11
2.2. LIST OF MAIN MODIFICATIONS	11
2.2.1. Add support of new namespaces	11
2.2.2. Removal step of pre-processing input	11
2.2.3. Use of link	12
2.2.4. Execution engine modification	13
2.2.5. Extension functions	13
2.2.6. Software updating	14
2.2.7. Improvement of the error report	14
2.3. NEW PROJECT CONFIGURATION	16
2.4. LIMITATIONS, EVOLUTIONS AND RECOMMANDATIONS	16
2.4.1. Support of Schematron variables declaration	16
2.4.2. Increase profile support	17
2.4.3. Modification of output generation	17
2.4.4. Temporality management	17
A. APPENDIX: WEB USER INTERFACE TRANSFORMATION	20
B. APPENDIX: AIXM BUSINESS RULES	22
C. APPENDIX: EXTENSION FUNCTIONS - QUICK REFERENCE	24

TABLE OF FIGURES

Figure 1 – Error Report in previous ARC version	14
Figure 2 – Error report in new ARC version	14



REFERENCES

Titre / Title	Référence interne / Internal reference	Référence externe / External Reference
[1] Saxon extension functions		Web Link
[2] Saxon Java extensibility		Web Link
[3] Schematron ISO norm		ISO/IEC 19757-3:2006

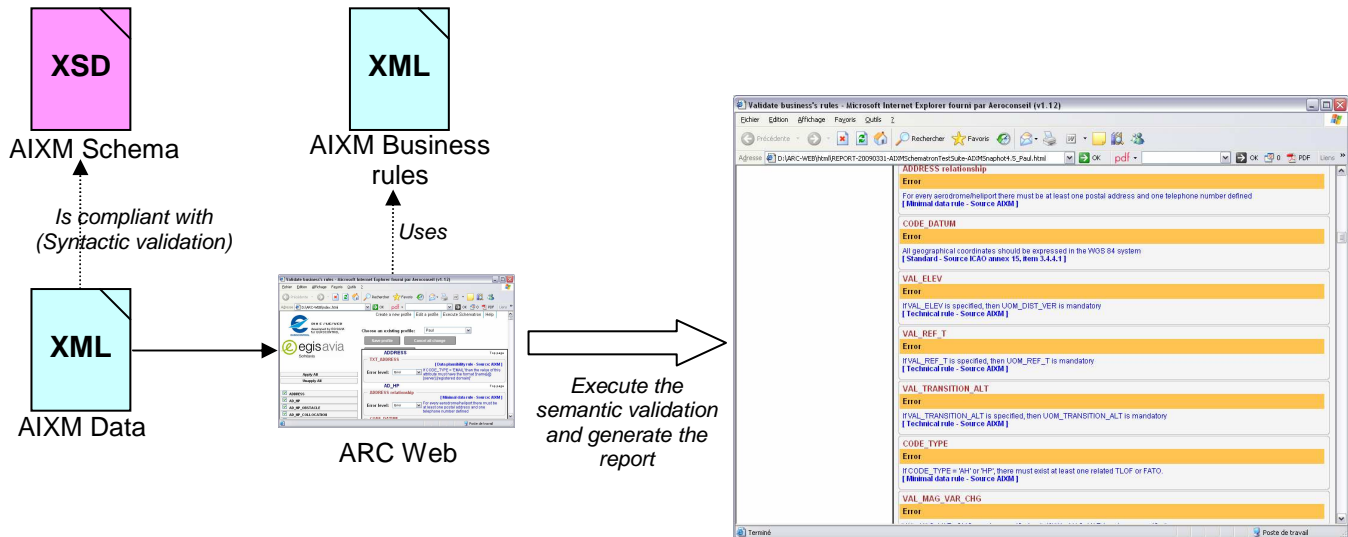
ABRÉVIATIONS / ACRONYMS

ARC	AIXM Rule Checker
AIXM	Aeronautical Information Exchange Model
XML	eXtensible Markup Language
HTML	HyperText Mark-Up Language
CSS	Cascading Style Sheets
PHP	Hypertext Preprocessor
XSLT	eXtensible Stylesheet Language Transformations
XSL	eXtensible Stylesheet Language



INTRODUCTION

ARC (AIXM Rule Checker) Web is a tool provided by EUROCONTROL and used during the validation process of AIXM data. It provides means to perform a semantic validation of XML data, in addition to the standard syntactic XML validation against an XSD Schema. These business rules defined for the semantic validation are formulated in two contexts, the natural language understandable by human people and the Schematron language, executable by computer on XML file.



The main features of ARC Web are the following:

- Creation of profiles that allow selecting a set of business rules to be used to check AIXM data (Example of profiles: AIXM 4.5, AIXM 5.0, ...)
- Display of all the business rules corresponding to the selected profile and selection of those to be used for the validation
- Execution of a Schematron validation of AIXML data using the rules corresponding to the selected profile
- Generation and display of a validation report listing all the errors identified in the AIXM data

In the frame of the development of new Business rules based on the Arinc424-19 standard for EUROCONTROL, AEROCONSEIL implemented some improvements of ARC tool. The objective of this technical note is to document and justify these improvements, and to propose additional upgrades which could be implemented in ARC in the future.



1. ANALYSIS OF THE ORIGINAL VERSION OF ARC WEB

1.1. ARCHITECTURE OF ARC WEB

ARC web is composed of two main components, the web user interface and the execution engine.

1.1.1. Web user interface

The web user interface allows the user to create profiles, to display business rules, to execute a Schematron validation and to display the report containing the validation results.

ARC is based on the web technologies HTML, Javascript and CSS and is available in two modes:

- the local/offline mode
- the online mode relying also on PHP technology

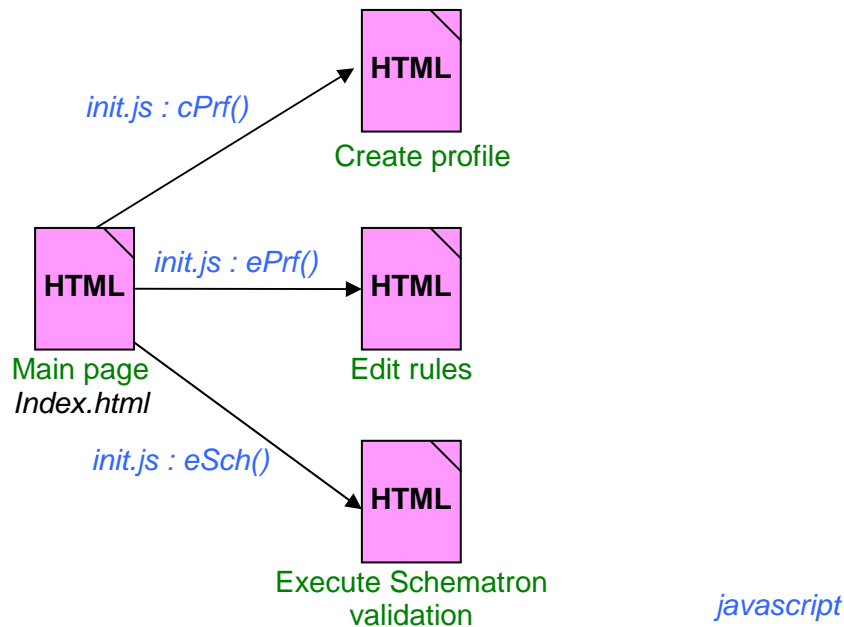
Only the local/offline mode version was analyzed and improved by AEROCONSEIL.

The web user interface consists of a set of HTML pages accessible from the main "index" page. The « index.html » page allows the user to choose between the following functions:

- o page "Create profile"
- o page "Edit rules"
- o page "Execute Schematron validation"

These 3 pages are generated by calling javascript functions "cPrf()", "ePrf()" et "eSch()" from file "init.js" which execute XSLT transformations.

The following schema depicts the overall HTML navigation:



The JavaScript function "activex.js : executeSchema" from the page "Execute Schematron validation" launches a Microsoft DOS BAT script that runs the execution engine with the appropriate parameters.

For more information see Appendix A.



1.1.2. Execution engine

The execution engine is a BAT script that runs successively a set of transformations which can be grouped in five main steps:

1. Apply specific transformations on input AIXM XML dataset (data pre-processing)
2. Extract the rules corresponding to the selected profile from the AIXM Business rules file and generate the Schematron rules file
3. Generate the Schematron XSL Validation file from Schematron rule file
4. Validate the input AIXM XML dataset file with the Schematron XSL Validation file
5. Extract the errors from the SVRL file and generate the HTML report

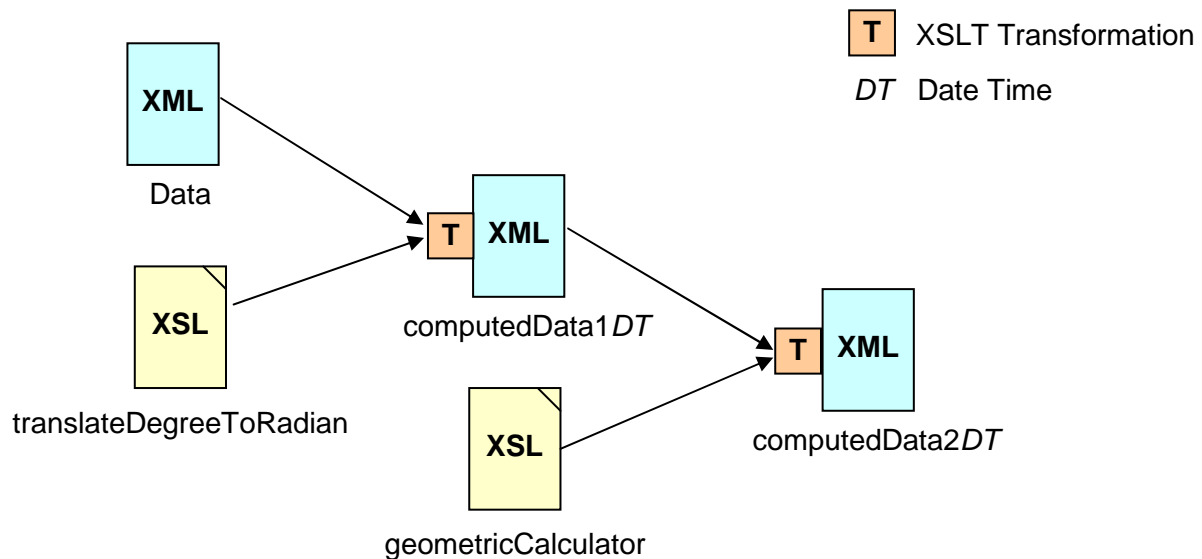
All these transformations are performed by calling the SAXON 8 XSLT engine in command line.

1.1.2.1. Specifics transformations

These transformations are used to pre-process the AIXM input file. This means that the transformations are executed before the validation. For AIXM, the following operations are performed:

- o The conversion of all angles from degree to radian
- o The computation of the all distances between geographical points

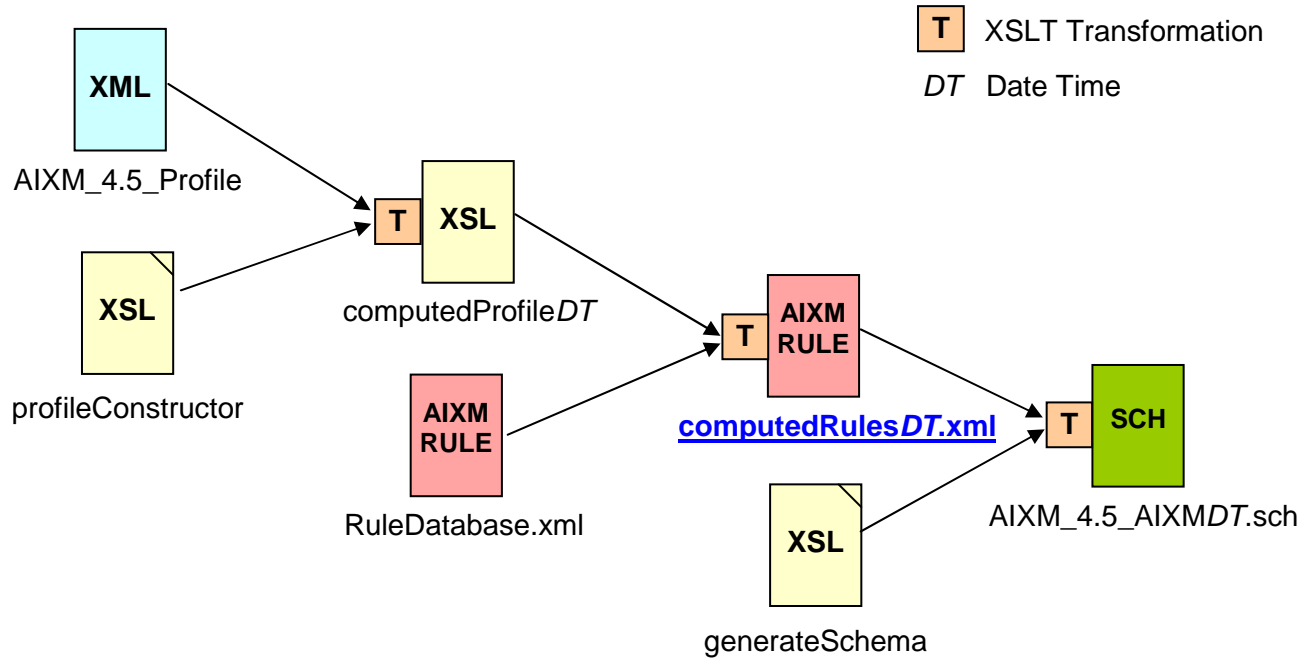
These computed values are directly added in the XML file before the validation.



1.1.2.2. Generation of “Schematron rules” from “AIXM Business Rules”

The original file “RuleDataBase.xml” contains all the AIXM Business rules whatever the targeted AIXM version (e.g.: rules for AIXM 4.5 are mixed with rules for AIXM 5.0). A XML element “version” is used to define the applicability of each business rule. The first step towards the production of Schematron rules is to extract the rules which correspond to the selected profile. This is done by an XSLT transformation which produces as output the file “computedRulesDT.xml” in which all the rules that are not applicable to the profile are erased. The following schema depicts this operation:

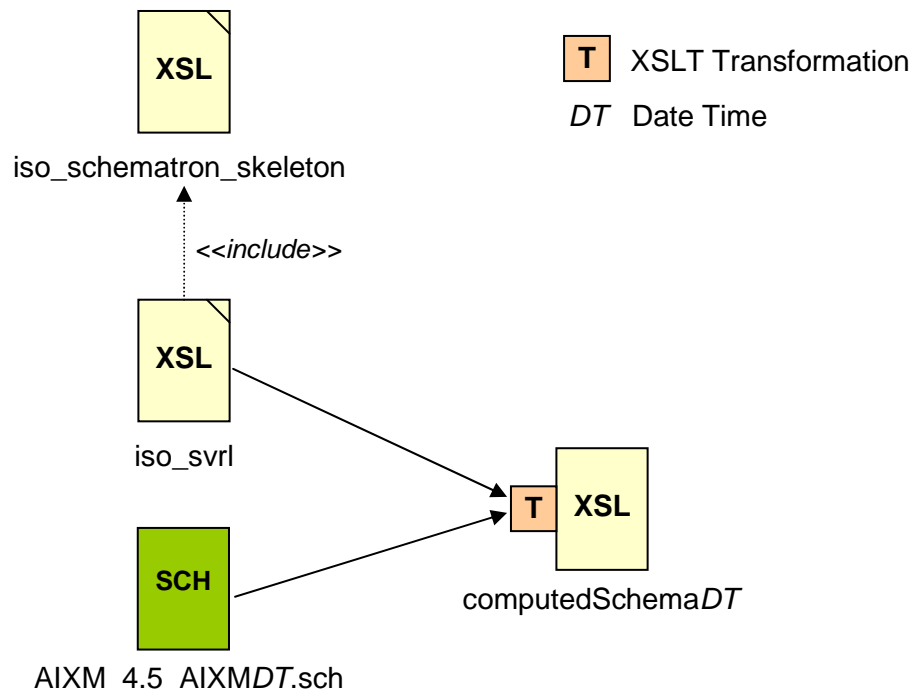




1.1.2.3. Generation of the Schematron validator file

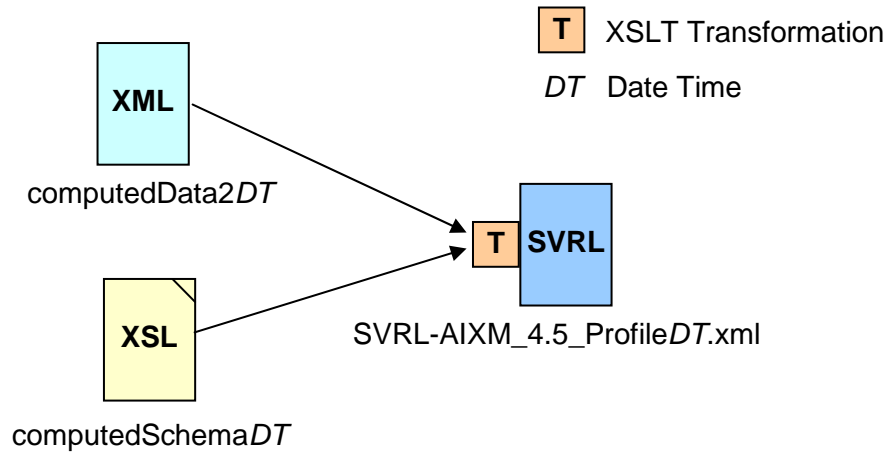
Schematron validator is an XSL file used to validate the AIXM input data file by executing an XSL transformation. This file is generated from the Schematron rules file and the software used to create it is the official Schematron engine. This software is composed of two files:

- o "iso_schematron_skeleton.xml", the base of the engine
- o "iso_svrl.xml" which includes "iso_schematron_skeleton.xml" and enables to format the output file in SVRL (Schematron Validation Report Language)



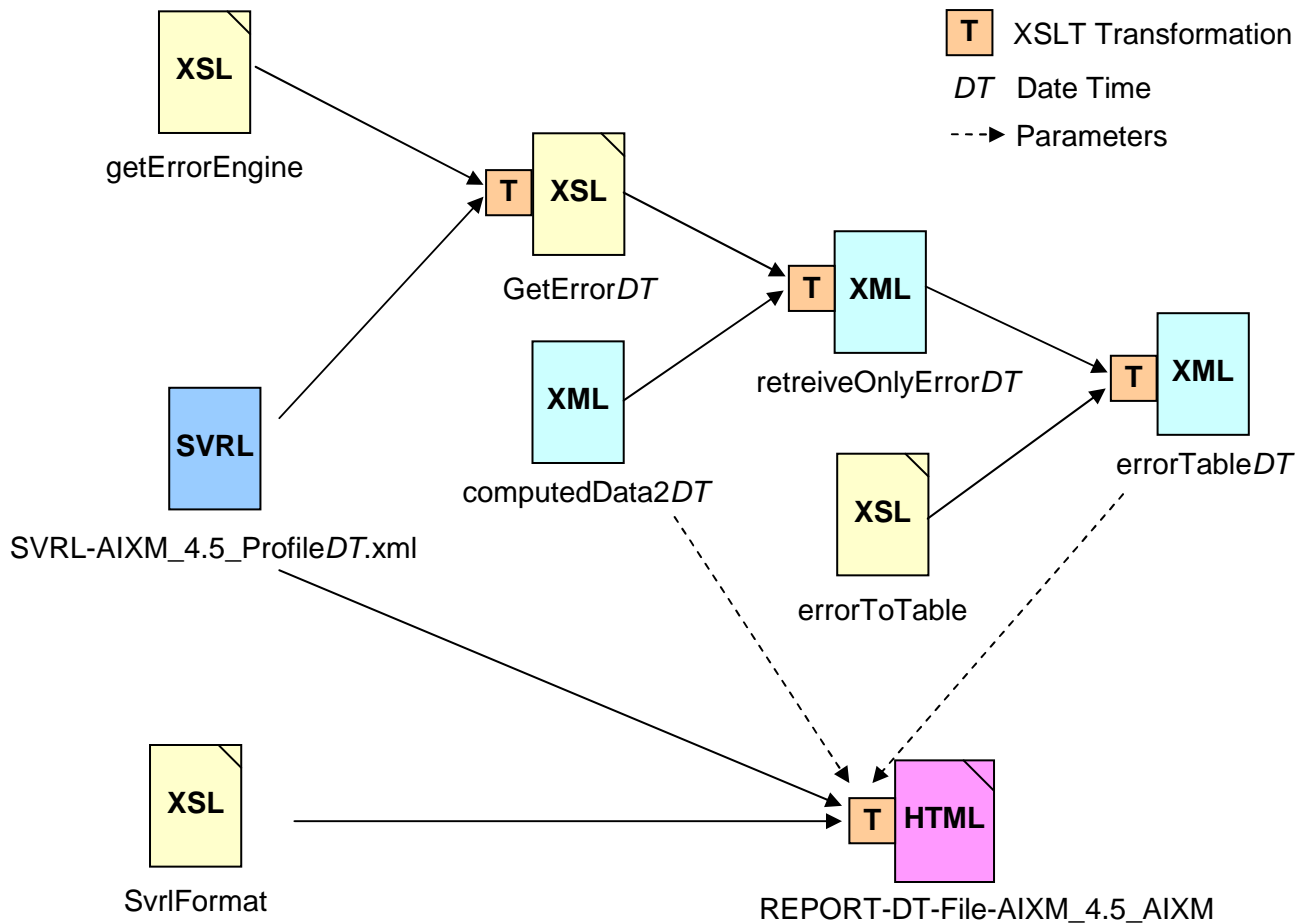
1.1.2.4. Validation of the AIXM input file

This step validates the pre-processed input file with the generated Schematron validator by executing an XSL transformation. The generated SVRL output contains the list of detected errors and the path of the invalid elements in the input XML data file.



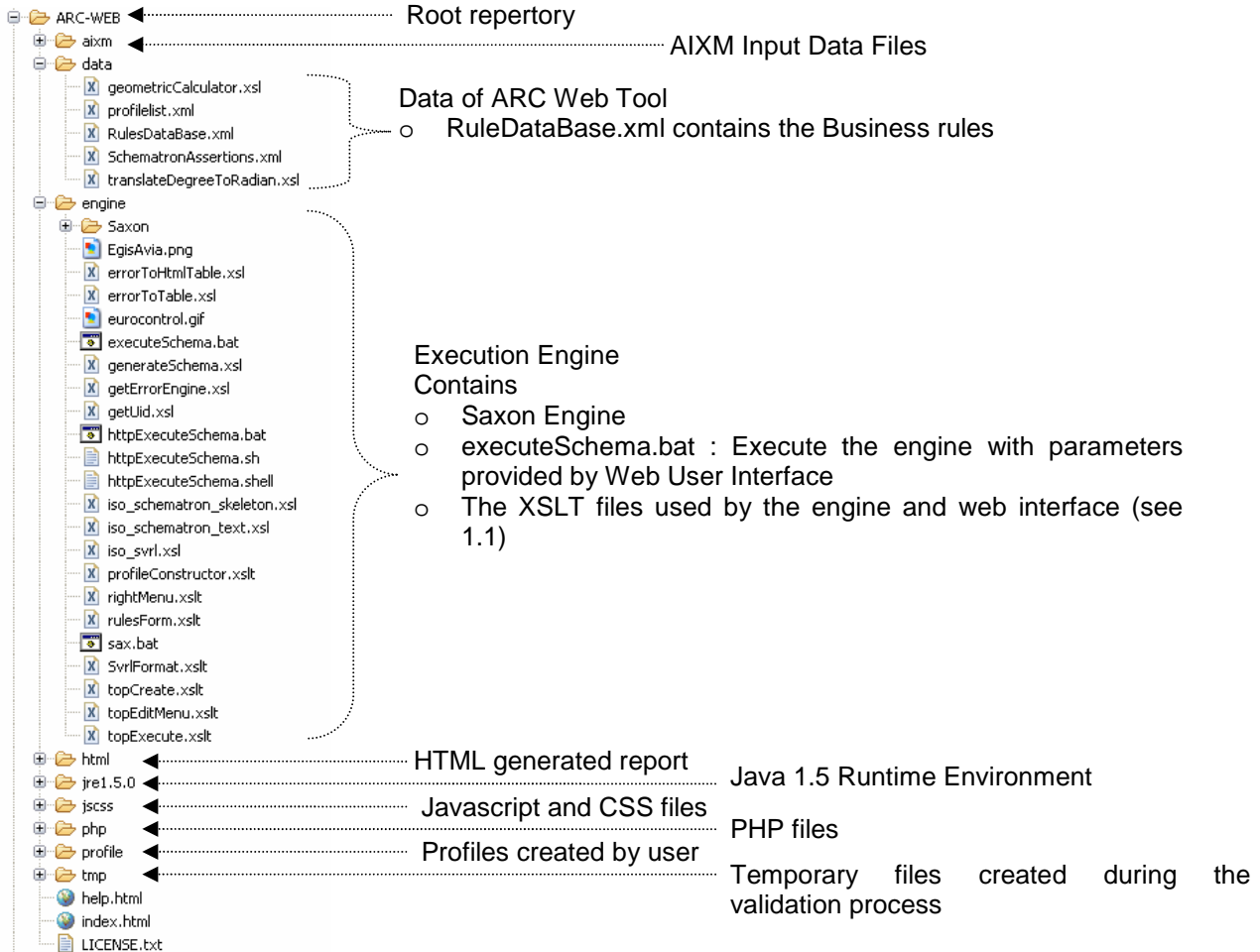
1.1.2.5. Generation of the HTML report

This step generates an HTML report from the SVRL file.



1.2. PROJECT CONFIGURATION

The following picture provides an overview of ARC-WEB content.



2. AEROCONSEIL IMPROVEMENTS OF ARC

2.1. OBJECTIVES

The main objective was to adapt the original version of ARC web, developed for the validation of AIXM data, for NDBX use. The main issues to be dealt with were the following:

- Support the XML namespaces
- Find a solution to handle the links between XML features
- Perform the geographical computation in WGS84 system (distance, course ...) directly during the validation and not using data pre-processing, so that the input XML data remains unchanged.
- Adapt the output presentation
- Ensure that these ARC improvements are not only valid for the validation of NDBX data but also for the validation of AIXM data.

2.2. LIST OF MAIN MODIFICATIONS

2.2.1. Add support of new namespaces

The original version of ARC tool did not support the XML namespace and consequently could not locate the appropriate XML elements inside the NDBX XML input data file during the Schematron validation. To correct this limitation, the following lines were added in the file generateSchema.xsl:

```
<!-- SAXON native Extension functions support -->  
<ns prefix="saxon" uri="http://saxon.sf.net/" />  
<!-- NDBX support -->  
<ns prefix="ndbx" uri="http://www.arinc.com/aec/xmlschema/xxx" />  
<!-- ARC Web Extension functions support -->  
<ns prefix="arcext" uri="java:com.aeroconseil.ARCExtFunctions" />  
<!-- GML support (for AIXM identifiers) -->  
<ns prefix="gml" uri="http://www.opengis.net/gml/3.2" />
```

The above list of namespaces is not exhaustive: any additional namespace that should be supported by ARC can be declared at this level.

2.2.2. Removal step of pre-processing input

The objective is to perform the geographical computations during the Schematron validation without modifying the input XML data. To do so, the AIXM-specific transformations have been removed and the files "translateDegreeToRadian.xsl" and "geometricCalculator.xsl" have been deleted. These operations are now realized at the execution time by using the extension functions.

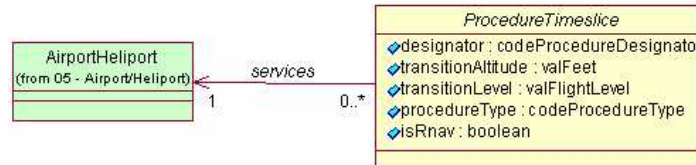
See chapter 2.2.5 for details on the new geographical computation mechanisms.



2.2.3. Use of link

XLink technology is used to create links between the NDBX features (Similarly for AIXM). XLink uses the xml element *href* to point to a portion of XML code located indifferently in the same or a separate file. The general syntax is **href = 'Mydocument.xml#Xpointer(xpathExpression)'**

Example: in NDBX, a *ProcedureTimeslice* services an *AirportHeliport*.



```

<!-- Reference to the AirportHeliport that has an identifier equal to 2 in the
Airport.xml file -->
<services xlink:href="Airport.xml#xpointer(//AirportHeliport[@identifier = '2'])"/>

<!-- Reference to the AirportHeliport that has an identifier equal to 3 in the same
file-->
<services xlink:href="#xpointer(//AirportHeliport[@identifier = '3'])"/>
  
```

When writing Schematron rules, in order to get the value of an *AirportHeliport* attribute from a *ProcedureTimeslice*, it is necessary to open the file that contains the corresponding data and to evaluate the XPath expression of *href*. This is realized directly by Saxon 9.0 XSLT engine which provides as an extension the functions **saxon:evaluate()** and **fn:doc()**.

The following examples explain how a static expression is evaluated:

```

<!-- Return the AirportHeliport that has an identifier equal to 2 in Airport.xml -->
saxon:evaluate("//AirportHeliport[@identifier = '2']", doc('Airport.xml'))

<!-- Return the AirportHeliport that has an identifier equal to 3 in the current file
-->
saxon:evaluate("//AirportHeliport[@identifier = '3']")
  
```

The objective is to evaluate dynamically a link during the validation. To do so, *href* must be split as described above:

```

<!-- Evaluate the XPath expression in the given file -->
<!-- The statement "substring-before(substring-after(@xlink:href, '#xpointer('), ''))"
return the value of XPath expression -->
<!-- The statement "substring-before(@xlink:href, '#)" return the document -->
saxon:evaluate(substring-before(substring-after(@xlink:href, '#xpointer('), '')),
doc(substring-before(@xlink:href, '#')))

<!-- Evaluatate the XPath expression in the current file -->
saxon:evaluate(substring-before(substring-after(@xlink:href, '#xpointer('), '')))
  
```

To simplify the reading and writing of Schematron rules, the extension function **arcext:getXPath(\$XPathExpression)** was developed (see 2.2.5), based on the assumption that all the XML features are gathered in the same file. The resulting code used in the Schematron rules definitions is:

```
saxon:evaluate(arcext:getXPath(@xlink:href))
```



2.2.4. Execution engine modification

The Execution engine was ported to the Java platform: all the transformations using the SAXON XSLT engine, which were initially run by the original BAT script, are now performed by a java program provided as a JAR file named "ARCEngine.jar". From now on, the original BAT script now contains the following line

```
"java -jar %dir%engine/ARCEngine.jar %profil% %rule% %datas% %result%"
```

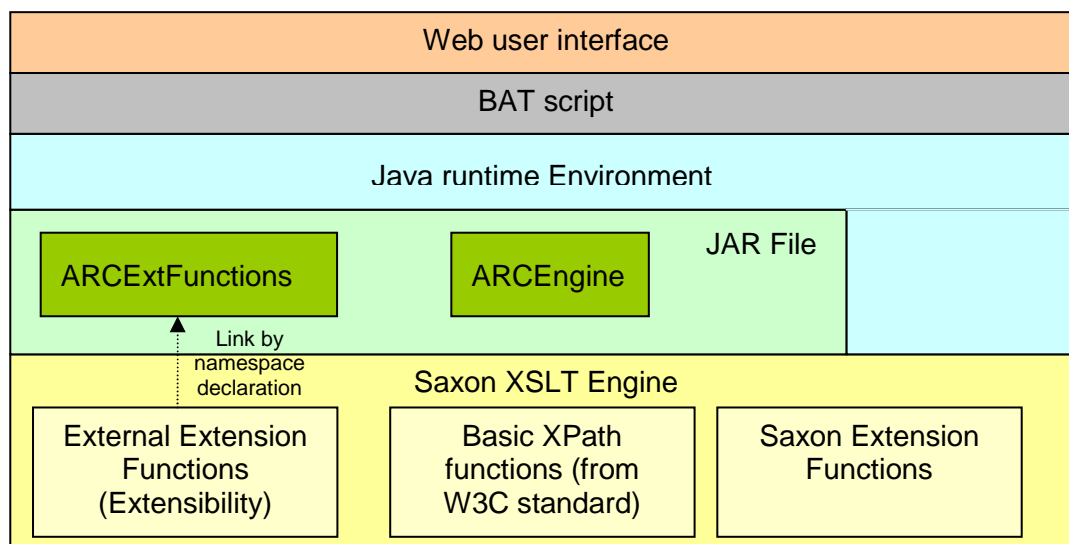
...which runs the Java execution engine and contains also the Java extensions functions described in chapter 2.2.5.

The java sources are composed of two files located in the same Java package "com.aeroconseil":

- o ARCEngine.java: The java execution engine
- o ARCExtFunctions.java: The extension functions

They are included in the JAR file.

The schema below summarizes the new architecture of ARC Web using the Java execution engine and the new Java extension module.



2.2.5. Extension functions

Saxon [2] includes by default some mechanisms allowing the user to write his own functions/methods in another environment (Java language for instance). These "custom" functions can then be called in the Schematron test assertion.

Example:

Declaration of the namespace in "generateSchema.xsl" file (Extensibility configuration)

```
<ns prefix="arcext" uri="java:com.aeroconseil.ARCExtFunctions"/>
```

Use of a custom function in Schematron rule

```
<!-- Test if the referenced Navaid (in ProcedureLeg) is NDB type -->
boolean(saxon:evaluate(arcext:getXPath(./reference_NavaidChoice/reference_NavaidSystem/@xlink:href))/NavaidSystemTimeslice[@type = 'NDB'])
```

Several extension functions were coded for the validation of NDBX data, especially for the geographical computing. A Quick reference is available in Appendix C and more details can be found in Javadoc.



2.2.6. Software updating

The version of JAVA has been updated from v.1.5 to v.1.6 and SAXON has been updated from v.8 to v.9.

2.2.7. Improvement of the error report

The original error report used the GML identifier to identify the invalid AIXM features. As GML is no longer used for NDBX, this data could not be used to identify invalid NDBX features. AEROCONSEIL noted that the standard file **iso_svrl.xml**, used to format the output SVRL file, had been updated with AIXM-specific code to extract the GML Id of the AIXM features, and, as a consequence, ARC was no longer a generic Schematron validation tool.

AEROCONSEIL has modified the file **iso_svrl.xml** which now gets the xpath expression of the invalid elements (and no longer the GML ID). This update works in both AIXM and NDBX contexts.

By clicking on this identifier, a copy of the XML record (without formatting modification) is now displayed instead of the original table summarizing the feature's properties.

AERO_GND_LGT - XXC SIG Top page

Agl	
AglUId	
txtName	XXC
codeType	SIG
txtDescrCharact	GP FLG (3) W EV 10 SEC Type and intensity (1 000 Candelas) : Marine W 500
geoLat	552200N
geoLong	0336900W
codeDatum	WGE
valGeoAccuracy	5
valElev	46
valElevAccuracy	5
valGeoidUndulation	12
Agt	
codeWorkHtr	HN

VAL_GEO_ACCURACY

Error

If VAL_GEO_ACCURACY is specified, then UOM_GEO_ACCURACY is mandatory
[\[Technical rule - Source AIXM \]](#)

VAL_ELEV

Error

If VAL_ELEV is specified, then UOM_DIST_VER is mandatory
[\[Technical rule - Source AIXM \]](#)

Figure 1 – Error Report in previous ARC version

```

ApproachTransitionTimeslice - :A829Database[namespace-uri]=http://www.arinc.com/aec/xmlschema/xxx[1]HDBXRecord[1]ApproachTimeslice[1]ApproachTransition[1]ApproachTransitionTimeslice[1]
<ApproachTransitionTimeslice xmlns:ndbx="http://www.arinc.com/aec/xmlschema/xxx" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001-01-01/xsi" type="ndbx:TerminalSegmentPointType">
  <startOfValidity>2001-01-01</startOfValidity>
  <endOfValidity>2001-01-01</endOfValidity>
  <ProcedureLeg>
    <ProcedureLegTimeslice pathAndTermination="IF">
      <startOfValidity>2001-01-01</startOfValidity>
      <endOfValidity>2001-01-01</endOfValidity>
      <SegmentLegFixUsage type="IAF" xsi:type="ndbx:TerminalSegmentPointType">
        <Fix>
          <reference_Waypoint xlink:href="#xpointer(//*[@identifier = '18'])"></reference_Waypoint>
        </Fix>
      </SegmentLegFixUsage>
    </ProcedureLegTimeslice>
  </ProcedureLeg>
  <ProcedureLeg>
    <ProcedureLegTimeslice pathAndTermination="TF" course="287.9" altitude="2000.0" altitudeDescription="4" routeOrHoldingDistance="14.8">
      <startOfValidity>2001-01-01</startOfValidity>
      <endOfValidity>2001-01-01</endOfValidity>
      <SegmentLegFixUsage xsi:type="ndbx:TerminalSegmentPointType">
        <Fix>
          <reference_Waypoint xlink:href="#xpointer(//*[@identifier = '10'])"></reference_Waypoint>
        </Fix>
      </SegmentLegFixUsage>
    </ProcedureLegTimeslice>
  </ProcedureLeg>
  <startsAt>
    <reference_Waypoint xlink:href="#xpointer(//*[@identifier = '18'])"></reference_Waypoint>
  </startsAt>
</ApproachTransitionTimeslice>

```

APP_TRANS_FINISH_WITH_XF_LEG

Warning

If an Approach Transition finish with XF leg, it is recommended to reference the same fix that the FAF or if exist, the FACF
[\[Recommended practice - Source ARINC 424-19 Attach. 5 - 6.3.1 \]](#)

```

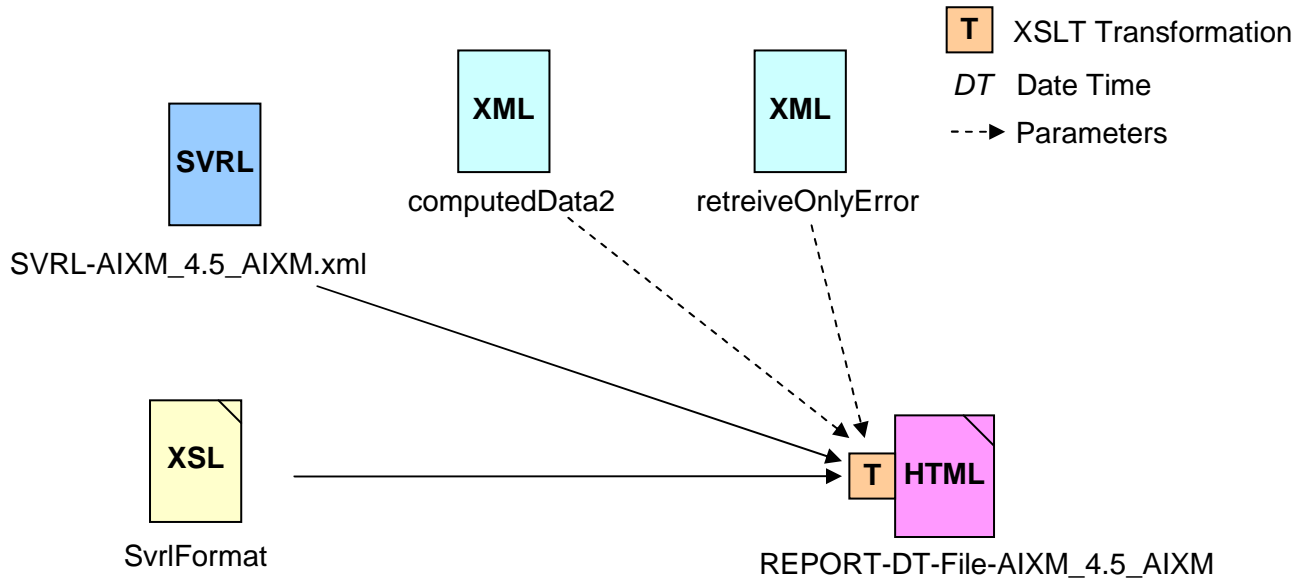
FinalTransitionTimeslice - :A829Database[namespace-uri]=http://www.arinc.com/aec/xmlschema/xxx[1]HDBXRecord[1]ApproachTimeslice[1]FinalTransition[1]FinalTransitionTimeslice[1] Top page
CF_LEG_IN_RNAV_WITHOUT_GBN
Warning
TF leg is the preferred to CF leg type in RNAV Terminal Procedures that are not using ground-based navaid references.
[ Recommended practice - Source ARINC 424-19 Attach. 5 - 1.3 ]
FINAL_TRANS_FACF_OR_FAF_IN_FIRST

```

Figure 2 – Error report in new ARC version



The transformation "errorToTable.xsl" of the HTML generation was deleted and the file "errorTable.xml" used for the "SvrlFormat.xsl" transformation was replaced by "retrievOnlyError.xml". Changes were also made in the files "SvrlFormat.xsl" and style sheet "report.css" to update the web interface.



2.3. NEW PROJECT CONFIGURATION



2.4. EVOLUTIONS AND RECOMMENDATIONS

This paragraph provides a list of potential evolutions or technical recommendations to go on improving ARC tool in the future.

2.4.1. Support of Schematron variables declaration

It could prove useful to update the Business rule schema (RuleDatabase.xml schema) and ARC Web tool in order to support the Schematron variables declaration. These variables are declared with the markup "let" in Schematron file:

```
<rule context="myContext">
  <!-- Variable declaration -->
  <let name="myVariable" value="XPathExpression"/>
  <assert test="$myVariable >= 10000 and $ myVariable <= 1000000">
  </assert>
</rule>
```

The main advantage of this evolution would be to reduce the complexity of the Schematron rules by declaring only once a complex variable. At the present time, the complete code corresponding to the variable is always repeated each time this variable must be referred to.



2.4.2. Increase profile support

All the business rules matching one given profile are executed during validation. It may be interesting to select only a group of rules, among the ones matching the selected profile, to be used for the validation.

Example: for the profile AIXM 5.0, run only the validation using only the rules applicable to Procedures.

The implementation of this upgrade was already initiated in the original ARC version, but it is not working (it is completely disabled).

2.4.3. Modification of output generation

Currently, some changes were made in the official Schematron engine “iso_svrl.xml”, which has some consequences:

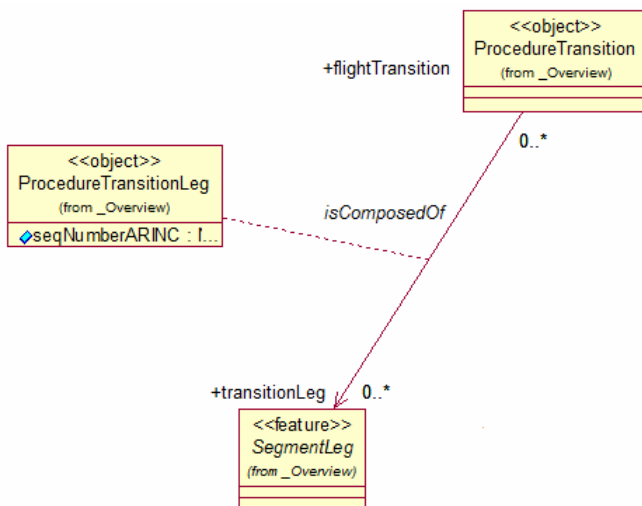
- The generated SVRL is not compliant with ISO Schematron [3] – see Annex D
- The Schematron engine can not be updated directly; the specific code included in “iso_svrl.xml” must be copied.

In future ARC versions, it could be useful to move these changes from “iso_svrl.xml” file to “GetError.xml” and to update the Schematron engine with the latest version, especially the Schematron version optimized for Saxon.

2.4.4. Temporality management

Temporality management is not implemented yet in the AIXM/NDBX business rules but is supported by Schematron.

Here is a simple example that explains how the temporality between two linked features could be managed. This example relates to the association ProcedureTransition → SegmentLeg defined in the AIXM Model. The business rule considered for this example is “*The first leg of a Final Approach Transition must be IP*”.



Original Schematron rule (without temporality)

Context: `//aixm:ProcedureTransition`

Test:

```
not(/parent::* /parent::* /aixm:codingStandard = 'ARINC_424_18')
or
not(/aixm:type='FINAL' and ./parent::* /parent::* /aixm:InstrumentApproachProcedureTimeSlice)
or
saxon:evaluate(arcxct:getXPath((./aixm:transitionLeg/aixm:ProcedureTransitionLeg/aixm:theSegmentLeg)[1]/@xlink:href))/*[aixm:legTypeARINC='IF']
```

The objective of the temporality management is to check the appropriate Timeslice instances.

The image displays two Schematron rule validation results. The left tree shows a valid instance (OK) and an invalid instance (KO). The right tree shows a valid instance (IF) and an invalid instance (TF).

Left Tree (OK): Shows a valid instance of `aixm:InstrumentApproachProcedure` with `gml:id=8`. It contains a `aixm:timeSlice` with `gml:id=1_1` and `gml:validTime` from 2009-04-01 to 2009-04-30. The `aixm:flightTransition` is `BASELINE`. The `aixm:ProcedureTransition` is `FINAL`. The `aixm:legTypeARINC` is `IF`.

Right Tree (IF): Shows a valid instance of `aixm:ArrivalLeg` with `gml:id=2`. It contains a `aixm:timeSlice` with `gml:id=2_1` and `gml:validTime` from 2009-04-01 to 2009-04-30. The `aixm:legTypeARINC` is `IF`.

Left Tree (KO): Shows an invalid instance of `aixm:InstrumentApproachProcedure` with `gml:id=1_2` and `gml:validTime` from 2009-05-01 to 2009-05-31. The `aixm:legTypeARINC` is `TF`.

A new test that checks the validity dates was added in the Schematron rule as follows:

Context: `//aixm:ProcedureTransition`

Test:

```
not(/parent::* /parent::* /aixm:codingStandard = 'ARINC_424_18')
or
not(/aixm:type='FINAL' and ./parent::* /parent::* /aixm:InstrumentApproachProcedureTimeSlice)
or
saxon:evaluate(arcxct:getXPath((./aixm:transitionLeg/aixm:ProcedureTransitionLeg/aixm:theSegmentLeg)[1]/@xlink:href))/*[xs:date(/gml:validTime/gml:TimePeriod/gml:beginPosition) >=
xs:date(current()/parent::* /parent::* /gml:validTime/gml:TimePeriod/gml:beginPosition)
and
xs:date(/gml:validTime/gml:TimePeriod/gml:endPosition) <=
xs:date(current()/parent::* /parent::* /gml:validTime/gml:TimePeriod/gml:endPosition)]
[aixm:legTypeARINC='IF']
```

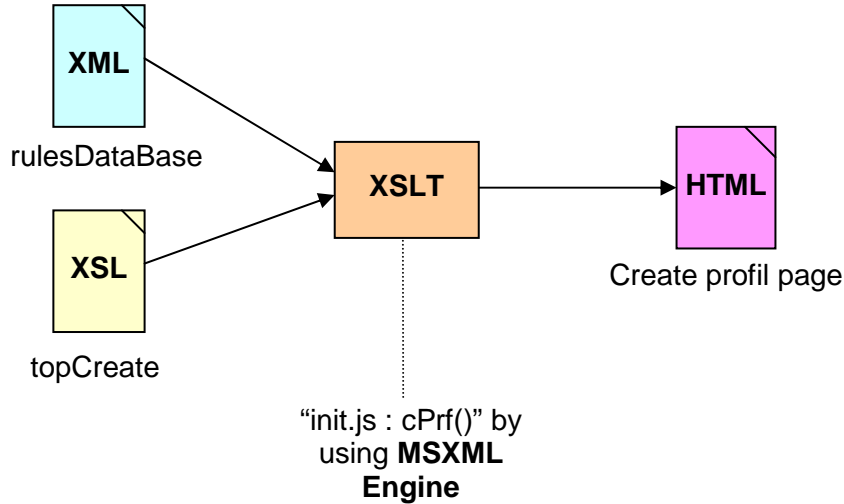


ANNEXES / APPENDICES



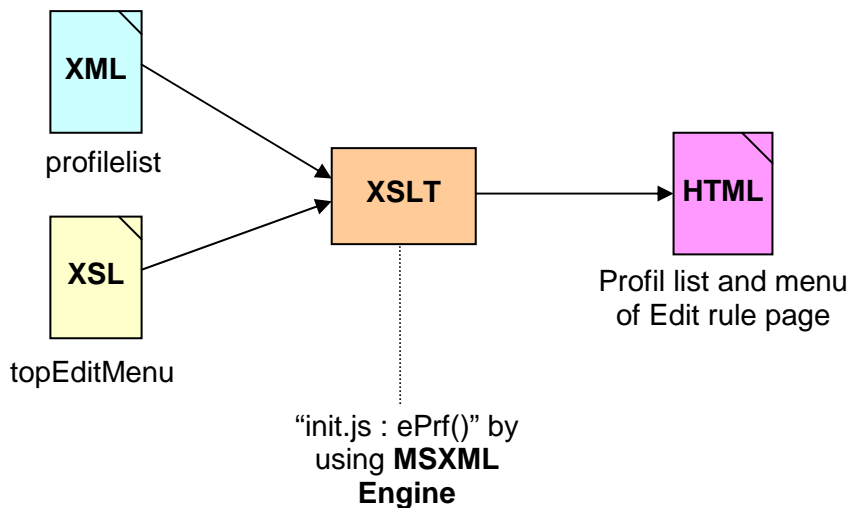
A. APPENDIX: WEB USER INTERFACE TRANSFORMATION

topCreate



The transformation “topCreate.xsl” generates the HTML page “create profile” by extracting the existing “rules versions” from the file “ruleDataBase.xml”.

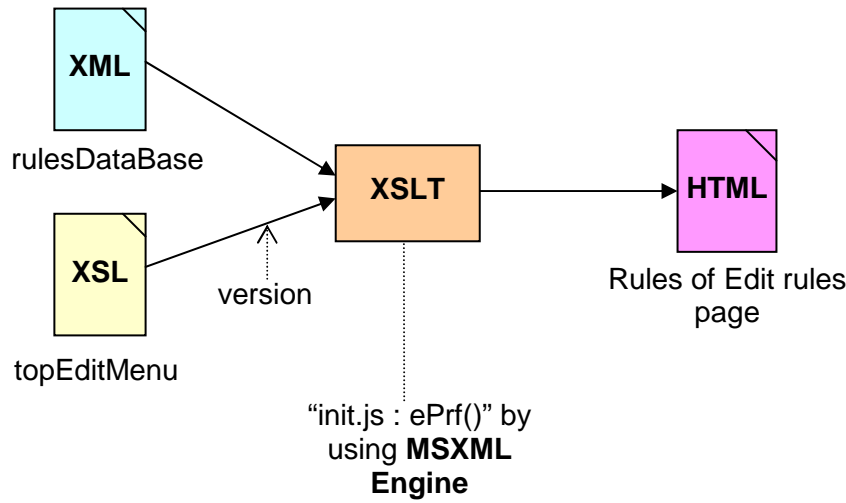
topEditMenu



The transformation « topEditMenu.xsl » generates the menu of the “edit rules” HTML page. It extracts the list of profiles from « profilelist.xml ».

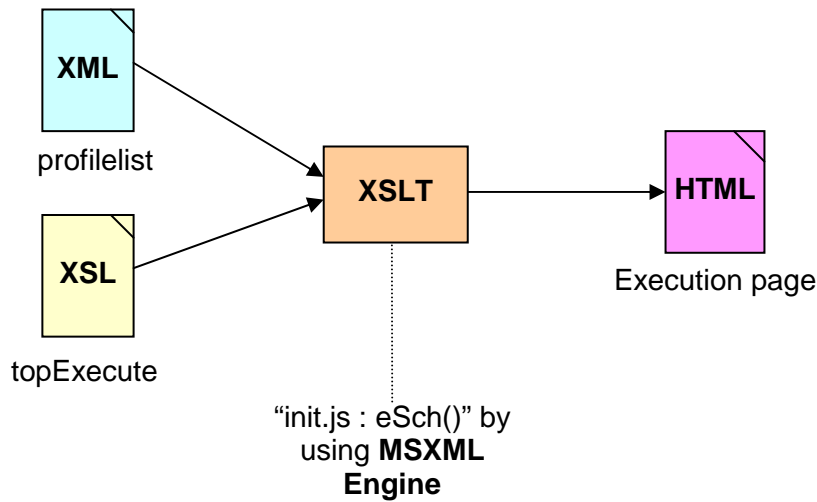


ruleForm



The transformation “ruleform.xsl” generates the list of rules the edit rule HTML page. It extracts them from “ruleDataBase.xml” according to the selected profile.

topExecute



The transformation “topExecute.xsl” generates the HTML execution page that allows launching the execution of Schematron rules. It extracts the list of profiles from « profilelist.xml ».



B. APPENDIX: AIXM BUSINESS RULES

Simple AIXM Business rules file

```

<?xml version="1.0" encoding="UTF-8"?>
<root xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:fo="http://www.w3.org/1999/XSL/Format"
      xmlns:fn="http://www.w3.org/2005/02/xpath-functions"
      xmlns:xdt="http://www.w3.org/2005/02/xpath-datatypes">
  <title>AIXM-Snapshot document validation</title>

  <rule id="12" context="/AIXM-Snapshot/Ahs" class="Ahs" name="AD_HP_GND_SER">
    <assert id="12_1" version="AIXMSnaphot4.5"
      test="(./AhsUid/codeType != 'FIRE') or boolean(codeCat)"
      name="CODE_TYPE"
      diagnostics="std_err">
      <comment uval="" type="Consistency rule" src="ICAO Annex 15, Appendix 1, AD
2.6">
        If the value of CODE_TYPE = 'FIRE' , then CODE_CAT is mandatory
      </comment>
    </assert>
    <assert id="12_2" version="AIXMSnaphot4.5"
      test="not(boolean(codeCat)) or (codeCat!= 'A10') or
(codeCatReference='NATO')"
      name="CODE_CAT_REFERENCE"
      diagnostics="std_err">
      <comment uval="" type="Consistency rule" src="AIXM">
        If CODE_CAT = 'A10' then CODE_CAT_REFERENCE must have the value
        'NATO'
      </comment>
    </assert>
  </rule>
  <rule id="6" context="/AIXM-Snapshot/Ag1" class="Ag1" name="AERO_GND_LGT">
    <assert id="6_1" version="AIXMSnaphot4.5"
      test="not(boolean(codeDatum)) or (codeDatum='WGE')"
      name="CODE_DATUM"
      diagnostics="std_err">
      <comment uval="" type="Standard" src="ICAO Annex 15, item 3.7.1.1">
        All geographical coordinates should be expressed in the WGS 84
        system
      </comment>
    </assert>
    <assert id="6_2" version="AIXMSnaphot4.5"
      test="not(boolean(valGeoAccuracy)) or boolean(uomGeoAccuracy) "
      name="VAL_GEO_ACCURACY"
      diagnostics="std_err">
      <comment uval="" type="Technical rule" src="AIXM">
        If VAL_GEO_ACCURACY is specified, then UOM_GEO_ACCURACY is mandatory
      </comment>
    </assert>
  </rule>
</root>

```



AIXM Business rules Schema

```

<xs:schema targetNamespace="http://www.arinc.com/aeec/xmlschema/AIXMBusinessRules"
  xmlns="http://www.arinc.com/aeec/xmlschema/AIXMBusinessRules"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  version="1.0">

  <xs:element name="assert">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="comment" />
      </xs:sequence>
      <xs:attribute name="version" type="xs:string" use="required" />
      <xs:attribute name="diagnostics" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="std_err" />
            <xs:enumeration value="std_war" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="name" type="xs:string" use="required" />
      <xs:attribute name="test" type="xs:string" use="required" />
      <xs:attribute name="id" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>

  <xs:element name="comment">
    <xs:complexType mixed="true">
      <xs:attribute name="uval" type="xs:string" use="required" />
      <xs:attribute name="type" type="xs:string" use="required" />
      <xs:attribute name="src" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>

  <xs:element name="root">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="title" />
        <xs:element ref="rule" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="rule">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="assert" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required" />
      <xs:attribute name="context" type="xs:string" use="required" />
      <xs:attribute name="class" type="xs:string" use="required" />
      <xs:attribute name="id" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>

  <xs:element name="title">
    <xs:complexType mixed="true" />
  </xs:element>

</xs:schema>

```



C. APPENDIX: EXTENSION FUNCTIONS - QUICK REFERENCE

double round(**double** d, **int** decimalPlace)
Round half down at decimal

double additionAngle(**double** angle, **double** additionValue)
Add the value to the angle

double substractAngle(**double** angle, **double** substractValue)
Subtract the value to the angle

double courseDiff(**double** inputCourse, **double** outputCourse)
Get the difference between two courses

double courseBetweenPoints(String latlong1, String latlong2)
Get the course between two points

boolean isConsistentICAO_AREA(String icao, String area)
Check the ICAO - AREA consistency
The table is given in the XML file "ICAO_Table.xml" located in the execution path :

```
<ICAO_Table>
  <ICAO name="AG">
    <Area name="SPA"/>
  </ICAO>
  <ICAO name="AN">
    <Area name="SPA"/>
  </ICAO>
  ...
</ICAO_Table>
```

boolean collocated(String latlong1, String latlong2)
Check if 2 nav aids are collocated: the lat/longs tolerance is 1/10 arc minutes

double distanceBetweenPoints(String latlong1, String latlong2)
Get the distance between two points

String getOppositeRunway(String runwayID)
Get the opposite runway of the runway
Ex. RW22L -> RW04R

int MAP_position(**double** course, String mapLatlong, String runwayLatlong)
Get the MAP position beside the runway

String getXPath(String xlinkExpr)
Get XPath expression contains in the xlink expression

double getAltitudeDiff(String fromPoint, **double** angle, String toPoint)
Get the altitude difference between the from point and to point with the angle at from point

double convertToNM(**double** value, String uom)
Convert the value to Nautical Mile according to this unit of measurement (support of AIXM uomDistanceType)

