

Kansas TRCC

Kansas eCitation Validation Engine Design

Detailed Design Document

Version 1.0.0

Table of Contents

1	Intent	4
2	Overview	4
3	Project Vision/Scope	4
3.1	Vision	4
3.2	Scope of eCitation Validation Engine	5
4	Business Context.....	5
5	Design Considerations	5
5.1	Assumptions and Dependencies.....	5
5.2	Goals and Guidelines	5
5.3	Development Platform & Technologies	5
6	System Overview	6
6.1	ISO Schematron XSL Implementation	6
6.2	UML Diagram	8
6.3	Validation Engine Class Diagrams	8
7	Interfaces and Classes	9
7.1	ISchematronValidation Interface	10
7.2	IXSLTTransformer Interface	10
7.3	IXsdValidation Interface	10
7.4	ValidationProcessor Class.....	10
7.5	XSLTSchematronValidator Class	10
7.6	XsdValidator Class.....	11
7.7	XSLTTransformerBase Class	11
7.8	XSLT1Transformer Class.....	11
7.9	XSLT2Transformer Class.....	12
8	Namespaces & Assemblies	12
8.1	Validation Assembly	12
8.2	Resources Assembly.....	12
9	Resources and Configurations	13
9.1	Resources	13
9.2	Configurations	13
9.3	Schematron File	13

9.4	Pre-processed Transform Cache	13
9.5	Consuming the Validation Processor Component	14
9.5.1	As a Component in a Web Service.....	14
9.5.2	As a Component in a Native C/C++ Application	14
10	Appendix	15
11	Definitions	15

1 Intent

This document will define in detail the design of the eCitation Validation Engine which is a Schematron-based component used within the eCitation Submission Service.

This document is intended to serve as guide for implementers who wish to utilize the Schematron Validation Engine within their application.

This is a technical document and is intended for technical audience.

2 Overview

The Traffic Records Coordinating Council (TRCC) commissioned a Strategic Plan for the development and implementation of a statewide electronic traffic citation (eCitation) system, with a central traffic citation information repository (central repository) accessible by state, local, and federal agencies, and the public. This eCitation system is an integral part of the statewide Traffic Records System (TRS) program initiated in 2005 and it will require integration with the Kansas Criminal Justice Information System (KCJIS). The TRS will be a virtual data warehouse that will provide state and local agencies with the ability to efficiently access traffic data to increase the safety of the motoring public. It will bring together information that is currently housed in separate, isolated repositories at Kansas Department of Transportation (KDOT), Kansas Highway Patrol (KHP), Kansas Department of Revenue (KDOR), Kansas Bureau of Investigation (KBI), and other agencies.

As a vital component of the TRS system, the eCitation project has been initiated with the goal of implementing a statewide eCitation system through which traffic citation data can be collected, analyzed, and distributed accurately, quickly, and cost effectively for the benefit of the public and state, local, and federal agencies.

Analysts International Corporation (AIC) has been tasked to complete a design for the production eCitation system that will serve as the foundation of the TRS 2.0 architecture. The eCitation system design includes a validation component to ensure the eCitation entered comply with the rules defined by the central repository.

The goal the current project phase of the eCitation Program is to provide detailed analysis and design of key eCitation components that would enhance the statewide electronic traffic citation (eCitation) prototype constructed in Phase 1B and provide guidance for implementing a hosting environment using core Microsoft technologies within the Kansas Criminal Justice Information System (KCJIS) data center to support future deployment of the eCitation system.

3 Project Vision/Scope

3.1 Vision

The goal of the statewide eCitation system is to electronically collect, analyze and distribute traffic citation data accurately, quickly and cost effectively for the benefit of the public and state, local and federal agencies.

3.2 Scope of eCitation Validation Engine

The scope of this document is limited to the details of the eCitation Validation Engine and does not include any details pertaining to other components utilized within the eCitation System. Thus, if some general components of eCitation (e.g. Core, Utilities) are mentioned in this document, only the details significant to the validation engine are discussed. Furthermore, this document does not include discussion of any processing that happens to the citation document after validation.

The activities included within the scope of this project are as defined in the Statement of Work.

4 Business Context

The eCitation Validation Engine is designed to use a standards based rules definition and processor (Schematron) to ensure that the eCitation entered comply with the defined rules. These rules are defined by the Central repository with input from the Law enforcement agencies and users. Every eCitation submitted to the Central repository system will be validated using this engine before it is stored and indexed. One of the goals of the eCitation program is to use the same validation engine within the client applications that are responsible for capturing the eCitation information. This will help in capturing valid data which in turn will aid in improving data quality and accuracy in the Central repository. The eCitation Validation engine is designed with this goal in mind.

5 Design Considerations

5.1 Assumptions and Dependencies

- The eCitation Message that are to be processed by this component are expected to be in NIEM Conformant XML format.
- The eCitation Validation Engine will run under Windows environment.

5.2 Goals and Guidelines

The eCitation Validation Engine is designed to support different client applications including but not limited to:

- BizTalk Orchestrations
- Web Services
- Windows Applications
- Web Applications

5.3 Development Platform & Technologies

The eCitation Validation Engine was developed in Visual Studio 2010 using C# .NET Framework 4.0.

The eCitation Validation Engine is currently utilizing 2 different XSLT engines to fully work with the current “ISO” version which supports different forms of XPATH queries. One form of XPATH is compatible with XSLT 1.* and another form of XPATH query is only compatible with XSLT 2.*. The out-of-the box .NET XSLT engine only supports XSLT 1.* syntax. Therefore, in order to support the full ISO Schematron standard, the design makes use of an open source XSLT engine (SAXON 9 Open Source

Home Edition parser) to do the processing, otherwise when XSLT 2.* is not required, the design will fall back to the standard .NET implementation.

As of .NET 3.5 Microsoft (MS) has not shown an interest in supporting XSLT 2 and forums about this issue indicate that it has not changed for .NET 4.0.

Below are a couple of forums regarding MS support of XSLT 2.

<http://social.msdn.microsoft.com/forums/en-US/xmlandnetfx/thread/77d7d224-0677-4c58-86df-fbc4f20f675b/>

<http://stackoverflow.com/questions/831300/what-is-the-current-state-of-xslt-2-0-availability-within-net>

<http://stackoverflow.com/questions/1525299/xpath-and-xslt-2-0-for-net>

6 System Overview

The eCitation Validation Engine is a collection of .NET classes which implement the Schematron validation engine. The implementation of the Schematron engine is based on the ISO Schematron XSLT processor found at schematron.com. This unique implementation reads a Schematron rules file, passes the rules in XML form through several XSL transformations until a validation XSL transformation is obtained and cached. Once the final validation XSL transformation is available, the NIEM XML is validated by passing the XML into the validation XSL transformation which will output validation results in XML form. This process is depicted pictorially in Section 4.1- ISO Schematron XSL Implementation and is initiated in the class ValidationProcessor. A client application may pass either an XML string or an XmlDocument as input directly to the Kansas.KBI.ECitation.Validation.ValidationProcessor class. The ValidationProcessor will perform a schema validation using the configured XSD files, followed by a Schematron rules validation.

Because ISO Schematron supports XSLT1 and XSLT2 type queries in the rules, a class called XSLTSchematronValidator pre-reads the Schematron rules to determine which version of the XSLT processor to use and adjusts accordingly. The UML diagram in Section 6.2 illustrates this process.

The different classes used in this validation engine are illustrated in Section 6.3 - Validation Engine Class Diagrams.

6.1 ISO Schematron XSL Implementation

The implementation of the eCitation Schematron validation engine is largely based on the XSLT reference implementation from Schematron.com. The ISO Schematron reference implementation, which can be found at <http://www.schematron.com/implementation.html>, takes the source Schematron rules file and applies a series of four successive XSLT transformations to the XML rules. The result of these four transforms is an entirely new “Validation XSLT” that can be applied to the eCitation XML batch document to be tested. When the Validation XSLT is applied to the eCitation batch document, the output is a series of validation errors. If the list of errors is empty, the eCitation batch document processed successfully; otherwise the errors can be reported back to the user. Further

information about the ISO Schematron XSLT pipeline can be found at <http://www.schematron.com/implementation.html>.

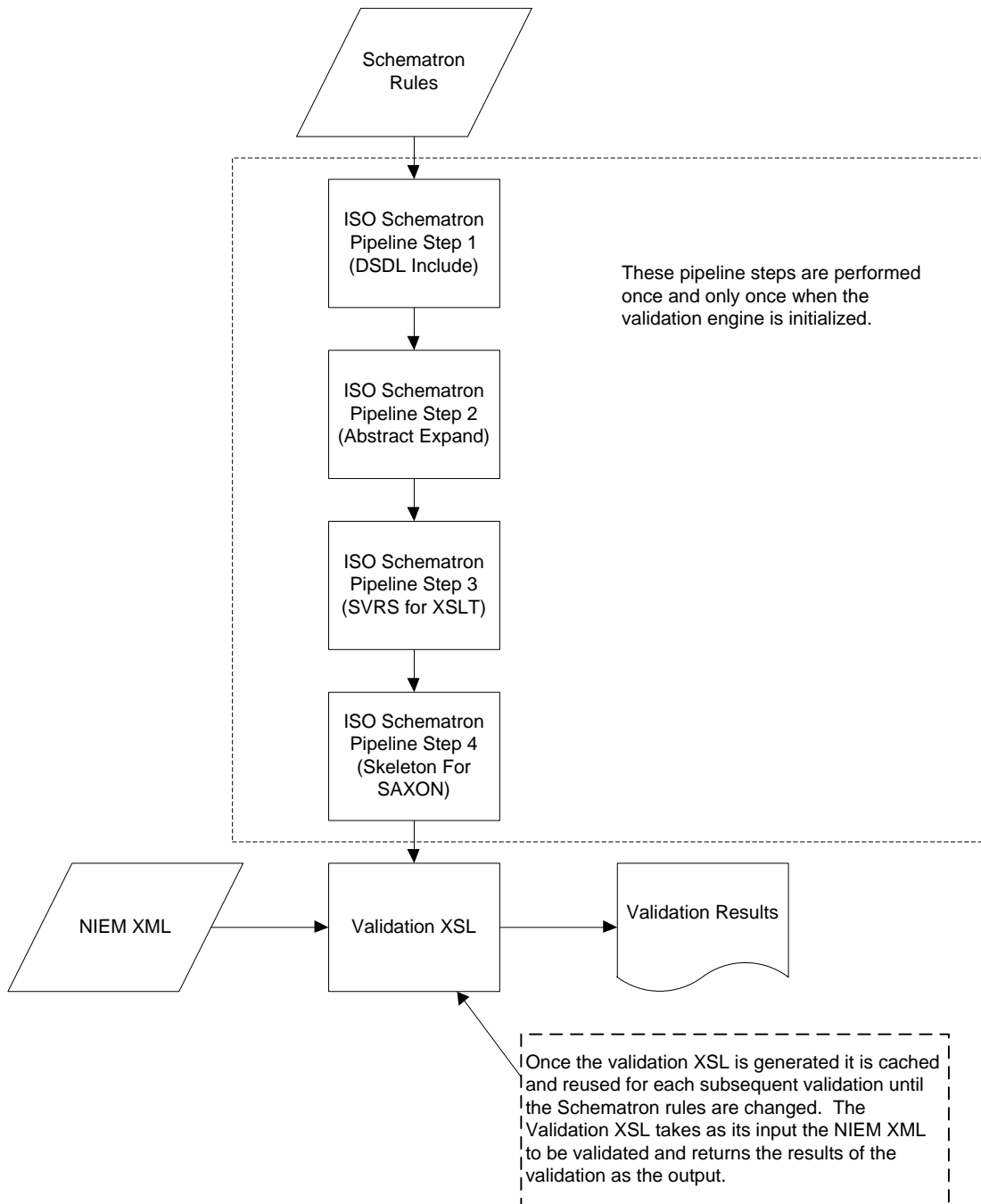


Figure 1. ISO Schematron Pipeline.

6.2 UML Diagram

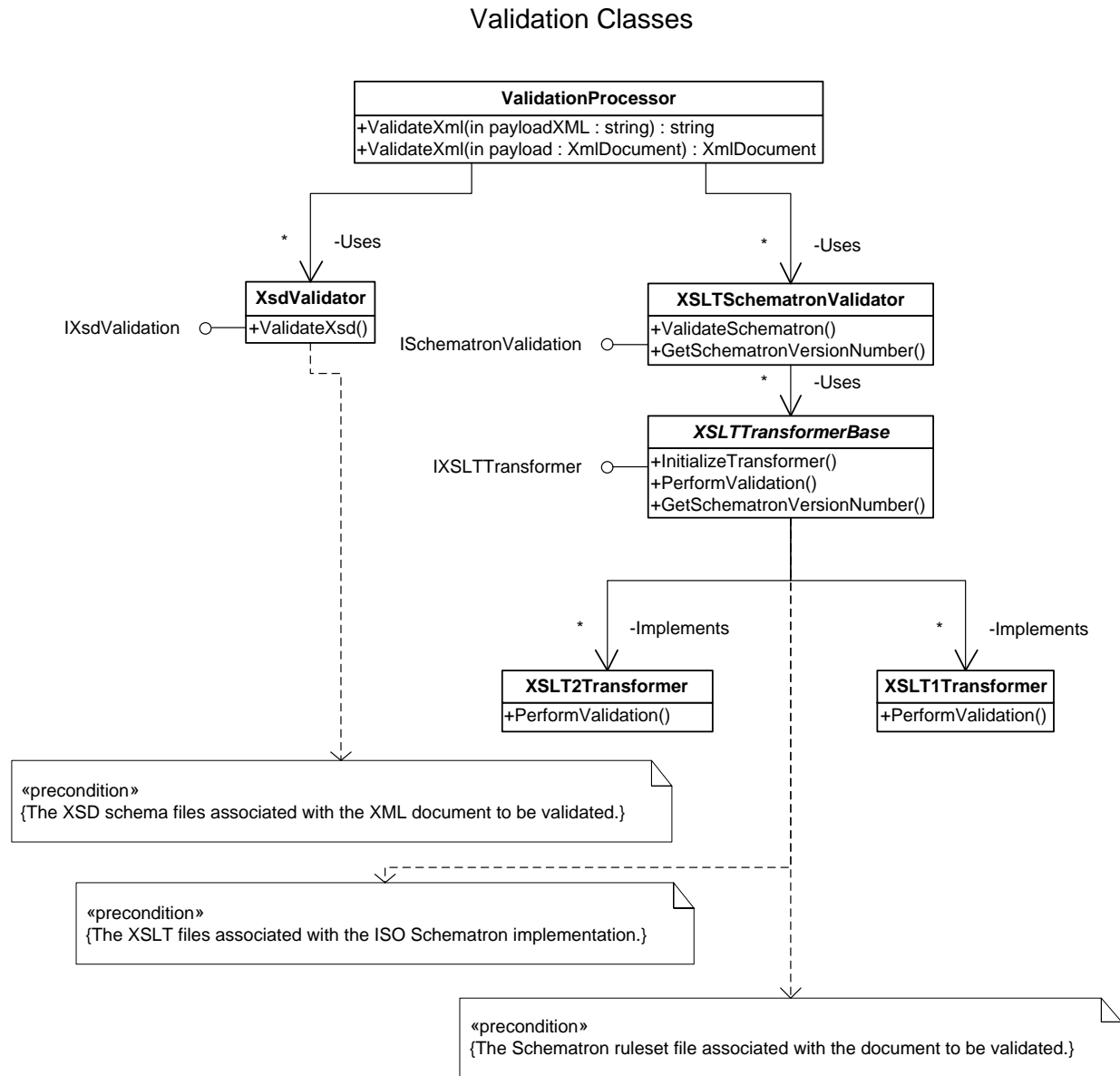


Figure 2. Validation Processor UML Diagram.

6.3 Validation Engine Class Diagrams

Following diagram illustrates a detailed definition of classes within the Schematron validation engine.

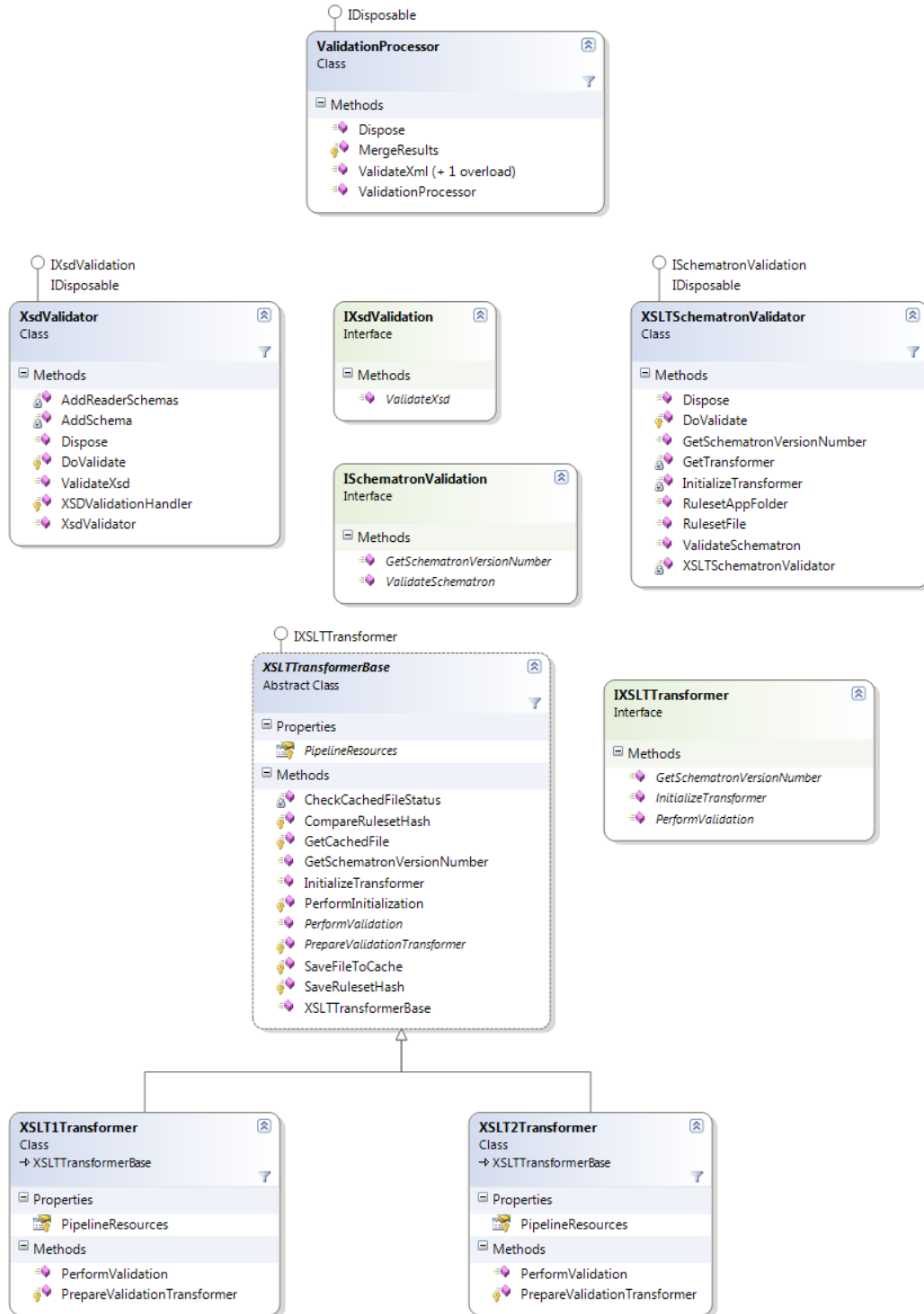


Figure 3. Validation Processor Class Diagram.

7 Interfaces and Classes

Below are interfaces and classes used within the validation engine. Only the public properties and methods are included in the details of these interfaces and classes.

7.1 ISchematronValidation Interface

ISchematronValidation Interface has one virtual method and one property to be implemented by any class inheriting from it:

- `XmlDocument ValidateSchematron(XmlDocument payload, string rulesetName);`
- `string GetSchematronVersionNumber(string rulesetName);`

This interface is designed to allow future implementation of the Schematron Validation for a different version of Schematron.

7.2 IXSLTTransformer Interface

IXSLTTransformer Interface has two virtual methods to be implemented by any class inheriting from it:

- `void InitializeTransformer(string rulesetName, string schematronVersion);`
- `XmlDocument PerformValidation(XmlDocument payload, string rulesetName);`
- `string GetSchematronVersionNumber(string rulesetName);`

This interface is designed to allow future implementation of the XSLTTransformer for a different version of XSLT.

7.3 IXsdValidation Interface

IXsdValidation Interface has one virtual method to be implemented by any class inheriting from it:

- `XmlDocument ValidateXsd(XmlDocument payload);`

This interface is designed to allow future implementation of the XSD Validation for a different schema.

7.4 ValidationProcessor Class

ValidationProcessor Class contains the following methods that are available to external classes or components.

- [constructor] `ValidationProcessor(string rulesetName)`

The ValidationProcessor class accepts one parameter in the class constructor which is the name of the ruleset file. This parameter must point to the name of the Schematron file (without file path or file extension as those are assumed) used for validation.

- `string ValidateXml(string xml)`

This method is the main method that is invoked from client applications that can pass a citation XML in the form of string. It validates the XML by invoking the ValidateXSD() method of XsdValidator Class and the ValidateSchematron() of XSLTSchematronValidator Class.

- `public XmlDocument ValidateXml(System.Xml.XmlDocument payload)`

This method is the main method that is invoked from client applications that can pass a citation XML in the form of XmlDocument. It validates the XML by invoking the ValidateXSD() method of XsdValidator Class and the ValidateSchematron() of XSLTSchematronValidator Class.

7.5 XSLTSchematronValidator Class

SchematronValidator Class inherits from IXSLTSchematronValidation Interface and implements the following methods that are available to external classes or components:

- `public XmlDocument ValidateSchematron(XmlDocument payload)`

This method checks the result string returned by validation methods and returns an XmlDocument containing any of the following result code:

- SUCCESS
- ERROR
- WARNING

- `public string GetSchematronVersionNumber(string rulesetName)`

This method returns the schematron version number used by the class that inherited from IXSLTSchematronValidation Interface.

7.6 XsdValidator Class

XsdValidator Class inherits from IXsdValidation Interface and implements the following methods that are available to external classes or components:

- `XmlDocument ValidateXsd(XmlDocument payload);`

This method validates the citation xml against the citation exchange document schema. It returns an XML-based result information.

7.7 XSLTTransformerBase Class

XSLTTransformerBase Class inherits from IXSLTTransformer Interface and implements the following methods that are available to external classes or components:

- `public void InitializeTransformer(string rulesetName, string schematronVersion)`

This method is invoked from the XSLTSchematronValidator class at the initiation of the validation. It uses the rulesetName to identify the Schematron file and prepare the validator for processing.

- `public abstract XmlDocument PerformValidation(XmlDocument payload, string rulesetName)`

This method is an abstract definition for performing XSL transformation, to be implemented by base class implementations XSLT1Transformer and XSLT2Transformer classes.

7.8 XSLT1Transformer Class

XSLT1Transformer Class inherits from IXSLTTransformer Interface and implements the following methods that are available to external classes or components:

- `public void PrepareValidationTransformer(Stream xsltStream)`

This method is invoked inside `InitializeTransformer()` method of the base class. It compiles the specified xsltStream using XSLCompiledTransform class which supports XSLT 1.0 syntax.

- `public XmlDocument PerformValidation(XmlDocument payload, string rulesetName)`

This method finally performs validation XSL transformation using XSLCompiledTransform class which supports XSLT 1.0 syntax.

7.9 XSLT2Transformer Class

XSLT2Transformer Class inherits from IXSLTTransformer Interface and implements the following methods that are available to external classes or components:

- `public void PrepareValidationTransformer(Stream xsltStream)`
This method is invoked inside `InitializeTransformer()` method of the base class. It compiles the specified `xsltStream` using `Saxon.Api.Processor` class which supports XSLT 2.0 syntax.
- `public XmlDocument PerformValidation(XmlDocument payload, string rulesetName)`
This method finally performs validation XSL transformation using `Saxon.Api.Processor` class which supports XSLT 2.0 syntax.

8 Namespaces & Assemblies

Namespaces are used to organize components and group related functionalities into different assemblies. The following are the different assemblies used within the Kansas eCitation Validation Engine.

8.1 Validation Assembly

The Validation Assembly contains all functionalities for the entire eCitation Validation component. The primary namespace called `Kansas.KBI.ECitation.Validation`, contains classes used in the validation engine. Additional sub-namespaces `Kansas.KBI.ECitation.Validation.Schematron`, `Kansas.KBI.ECitation.Validation.Utility`, `Kansas.KBI.ECitation.Validation.XSD` and `Kansas.KBI.ECitation.Validation.XSLT` contain classes used by the validation engine to perform its operation. This assembly is compatible with .NET 2.0 and was built and compiled in Visual Studio 2008.

Namespace: `Kansas.KBI.ECitation.Validation`

Assembly Name: `Kansas.KBI.ECitation.Validation.dll`

8.2 Resources Assembly

The Resources Assembly contains all of the non-code files associated with the validation engine. These files include the XSLT files associated with the ISO Schematron implementation, the XSD files associated with the eCitation IEPD and the SCH files associated with any Schematron rules. These files are prepared as a separate resource assembly so that they can be updated and deployed independently from the validation code components.

Namespace: `Kansas.KBI.ECitation.Validation.Resources`

Assembly Name: `Kansas.KBI.ECitation.Validation.Resources.dll`

9 Resources and Configurations

9.1 Resources

The following external files are used by the Validation Engine:

- iso_dsdl_include.xml - used to preprocess Schematron schema. This is a macro processor to assemble the schema from various parts. If your schema is not in separate parts, you can skip this stage. This stage also generates error messages for some common XPath syntax problems.
- iso_abstract_expand.xml - used to preprocess the output from the first preprocessing with iso_dsdl_include.xml. This is a macro processor to convert abstract patterns to real patterns.
- iso_svrl_for_xslt1.xml - used to compile the schema into an XSLT script which in turn invoke iso_schematron_skeleton_for_xslt1.xml
- iso_svrl_for_xslt2.xml - used to compile the schema into an XSLT script which in turn invoke iso_schematron_skeleton_for_saxon.xml
- iso_schematron_skeleton_for_saxon.xml - invoked in Iso_svrl_for_xslt1.xml
- iso_schematron_skeleton_for_xslt1.xml - Iso_svrl_for_xslt2.xml
- ComplexCitationRules.sch - the Schematron validation file containing business rules to be used by the validation engine.
- CitationExchange-DocumentSchema.xsd - used for validation of the XSD schema. Also involves numerous included XSD files associated with the extension and subsets of NIEM.

9.2 Configurations

The current version of the validation engine does not require any configuration parameters or configuration file to be present for operation. Any parameters required for configuration are to be provided in the ValidationProcessor constructor parameters.

9.3 Schematron File

The Schematron rules files are expected to be installed in a sub-folder of the assembly dlls. The assembly installer program will automatically place the file in the appropriate location for use at runtime. The proper assembly filename is provided at the time that the validation processor class is instantiated in the constructor parameter.

9.4 Pre-processed Transform Cache

The XSLTSchematronValidator supports the ability to cache the pre-processed intermediate XSLT file after it has been processed through the ISO implementation XSLT pipeline. Once the Validation XSLT (see the diagram in section 6.1) is prepared, it is cached to the Microsoft Windows Common Application Data folder to the following path:

`%CommonApplicationData%\KBI\Validation\%RulesetName%\%RulesetName%.xslt`

where `%RulesetName%` is the name of the Schematron ruleset file that is used and `%CommonApplicationData%` is one of the following folders:

- Win XP/Server 2003 - C:\Documents and Settings\All Users\Application Data
- Win Vista/Win 7/Server 2008 - C:\ProgramData

Once the file is cached and the Schematron file remains unchanged, the XSLTSchematronValidator will use the cached version of the file each time the validator is initialized, rather than processing the Schematron ruleset file from scratch.

9.5 Consuming the Validation Processor Component

9.5.1 As a Component in a Web Service

When used as a component in a Web Service, the deployment of this component will be integral to the deployment of the Web Service. The component will be integrated with the Web Service code during development and packaging and therefore there will be no additional deployment packages required for the Validation Processor components.

9.5.2 As a Component in a Native C/C++ Application

The Validation Processor component is built in .NET Framework version 2.0 and exposed to native C/C++ applications through the use of a COM Callable Wrapper (CCW) which makes it accessible from a native C/C++ application as a COM object. The following Microsoft documentation describes the operation of the CCW:

When a COM client calls a .NET object, the common language runtime creates the managed object and a COM callable wrapper (CCW) for the object. Unable to reference a .NET object directly, COM clients use the CCW as a proxy for the managed object.

The runtime creates exactly one CCW for a managed object, regardless of the number of COM clients requesting its services. As the following illustration shows, multiple COM clients can hold a reference to the CCW that exposes the INew interface. The CCW, in turn, holds a single reference to the managed object that implements the interface and is garbage collected. Both COM and .NET clients can make requests on the same managed object simultaneously.

COM callable wrappers are invisible to other classes running within the .NET Framework. Their primary purpose is to marshal calls between managed and unmanaged code; however, CCWs also manage the object identity and object lifetime of the managed objects they wrap. [Source: Microsoft.NET documentation “COM Callable Wrapper” <http://msdn.microsoft.com/en-us/library/f07c8z1c.aspx>]

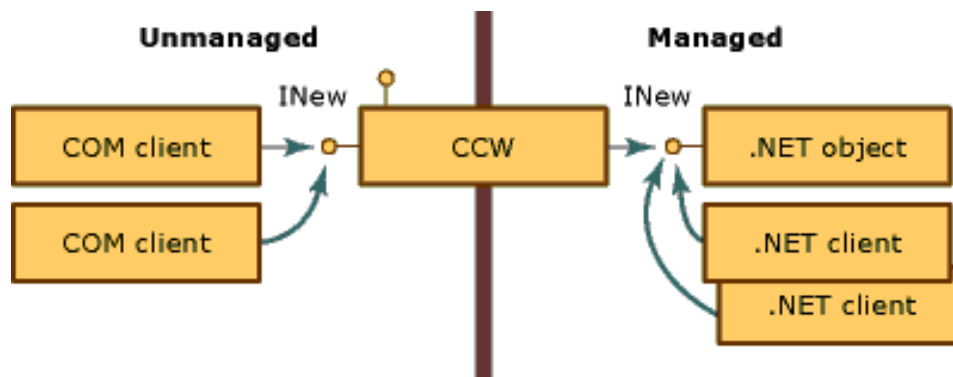


Figure 4. Accessing .NET objects through COM callable wrapper. [Source: Microsoft.NET documentation]

The components associated with the Validation component will be provided to third-party developers as a Visual Studio Installer Merge Module. In this way, the deployment of the Validation component can be seamlessly integrated into a unified installation package with other applications.

10 Appendix

- www.Schematron.com - please visit this site for more information on Schematron Validation
- <http://saxon.sourceforge.net/> - please visit this site for more information on the Open source SAXON XSLT processor

11 Definitions

Acronym/Term	Definition
NIEM	National Information Exchange Model
XML	eXtensible Markup Language
XSL	eXtensible Stylesheet Language
XSLT	eXtensible Stylesheet Language Transformations
XPath	XML Path Language
ISO	International Organization for Standardization
DSDL	Document Schema Definition Language
SVRL	Schematron Validation Report Language
XSD	XML Schema Definition