# Java Based GUI Editor for SAML Assertion Manipulation

## Project Report

Siddharth Anand
School of Electrical Engineering and Computer Science
Oregon State University

Major Professor: Dr. Bella Bose

# Acknowledgements

I am deeply grateful to my advisor Dr. Bella Bose for his support, guidance and patience throughout my project work. I would also like to thank Dr. Timothy Budd for his support and understanding.

# Table of Contents

# List of Figures

# 1. Introduction

Security Assertion Markup Language (SAML) is an open framework for sharing security information on the Internet through XML documents. SAML was developed to provide a common language for sharing of security services between companies engaged in B2B and B2C business transactions.

SAML allows companies to securely exchange authentication, authorization, and profile information between their customers, partners, or suppliers regardless of the security systems or e-commerce platforms that they have in place today. As a result SAML promotes the interoperability between disparate security systems, providing the framework for secure e-business transactions across company boundaries.

The "Persona" Concept (as described below) envisages using SAML for the transfer/exchange of user security credentials, held in a *persona* device or pointed to by the *persona* device, between the *persona* device and the Web-Service Provider (WSP). Here, WSP is the entity that provides the services the user intends to use.

The security credentials, mentioned above, would be held in the *persona* device (or at a location pointed to by the *persona* device) in the form of SAML Assertions – a text file containing security related data in a predetermined and predefined format.

SAML Edit®, the SAML editor created as part of this project provides a Java®-based Graphical User Interface (GUI) tool to manipulate such SAML Assertion text files in a manner that does not require the user to have a command on SAML or its vocabulary. All he/she needs the various pieces of information that constitute a particular type of security credential.

# 2. Overview: Persona Concept

The "Persona Concept" is a web-based information sharing architecture supported by a personally held security device called the "Persona". This concept is aimed at preventing on-line identity theft and protecting critical infrastructures including those for emergency response and law enforcement. The aim of the Persona Concept is to protect the confidentiality and integrity of both private and enterprise data while federating information access across end-user devices [1] and across organizations.

The Persona Concept is built around the user's physical possession of a device, namely a Persona Device ("Persona" for short). This device contains electronic credentials that attest to the identity of its owner. The Persona Device is used to gain access to business systems and/or critical infrastructures containing vital information and services of both users and organizations. Agents/users from distinct organizations can use their Persona Devices to collaborate and share sensitive information to support transactions and joint operations. The Persona may be used to access a range of devices, including PCs, PDAs and cell phones that do not necessarily belong to the user.

The Persona Concept encompasses Persona Devices and credential issuing services. The Persona Device is a critical access and control component and therefore resists identity theft and tampering. The Persona Device is designed to integrate with commonly available web-clients and is supported by credentialing services for securely managing electronic credentials used by the device. The interface to the Persona Device is designed to present a common and easy-to-use human interface. The security mechanism employed by the Persona Concept offer end-to-end protection that distributes the security policy access and decision points to enhance scalability over more traditional security architectures.

Encryption keys, digital certificates, and more generalized credentials associated with a given user are distributed to the owner's Persona Device to support positive identification of the user in conjunction with server-side authentication. Possession of the device, plus password and/or biometric authentication, together with audit trails guarantees that users are who they say they are for the purposes of preventing identity theft and misusage. The Persona may be physically connected to the terminal device to enable operation and removed when the user has completed their online tasks.

Rather than allowing access based on user name or login ID, the Persona Concept makes use of electronic credentials to enable information systems to mediate access in accordance with the *roles* and attributes of the user. The user can carry their Persona Module with them and use it to instantly establish a connection with any system that recognizes Persona Devices and thereby access resources that match the user's quality attributes. Labels (quality attributes) associated with the user must match those associated with the resource – for example a file, data element or application program.

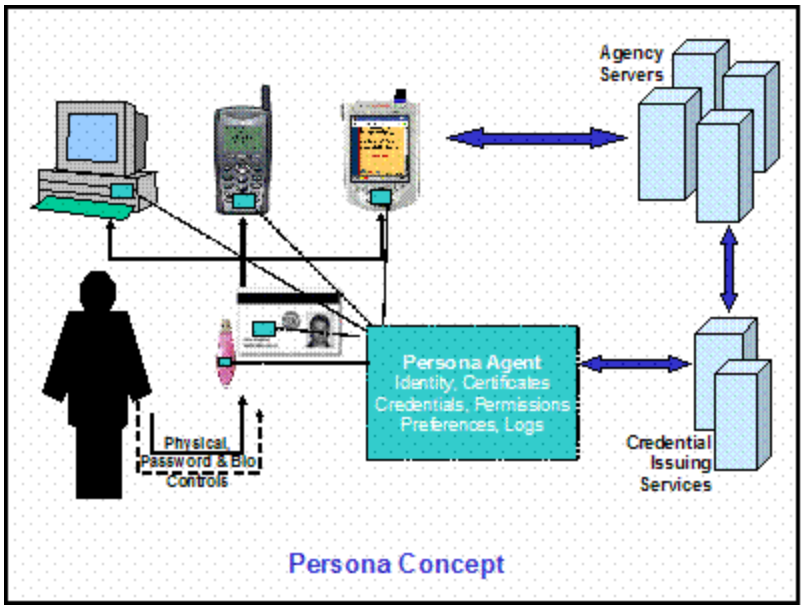



Figure 1: Persona Concept

In the context of emergency response and public safety, a Persona Device may be used to identify its owner, for example, a police officer, a fireman, an agent from the FBI, or a member of the coast guard. Role-based credentials may also be allocated to identify a Police Chief, a firefighter in a particular district, a senior manger or administrator in the CIA or even a State Governor. For enterprise, e-government, e-business and critical infrastructure applications, role-based credentials within the Persona would be used to unambiguously authenticate the user to the system and the system to the user. The user's identity is maintained within the Persona Device which is protected against loss, tampering and misuse.

Several technical papers document the initial explorations and assessments of the Persona Concept.

[1] Fixed devices such as PC workstations; mobile devices such as wireless PCs, PDAs and cell phones

# 3. Persona Concept: A Consumer-centered Identity Model

## 3.1 Motivation

In January 2002, a Jupiter Matrix* study found that almost 29 percent of online consumers were of the view that merchants could improve customer service by developing a simpler website sign-in process and that 42 percent consumers were of bothered by having to user different sign-in names and passwords to access different sites on the Web.

Given that most Web sites use a custom authentication system, this hardly comes as a surprise. Having to remember multitude of usernames and/or password makes it difficult for the users to access sites, negatively impacting both the users and the Web site/business.

Internet/electronic commerce has grown at a scorching pace over the last decade, as also the proliferation of internet access. Today, online business transactions are carried out from all kind of devices, be it he ubiquitous PC, the cell phones, PDAs and what not. In 2003, 100 billion dollars was spent on e-shopping in the United States. In fact, in an interesting survey published recently, about 70-85 percent of Americans have purchased/sold at-least one item on e-bay. These figures are an indication of the propensity towards online business transactions, even among the common day to day users. If anything, these numbers are set to grow significantly over the coming years.

Managing personal information — including, but not limited to, ones identity, credit card/other financial data — in a secure and reliable manner is turning out to be a major area of concern in such a scenario. Users end up providing sensitive information to web service providers (WSP) — a *generic* term we use for vendors/businesses. This results in a trust relationship borne more out of the need of a user to obtain a service than out of volition. Such a relationship is fragile at best. The Public Key Infrastructure (PKI) has addressed these concerns to a certain extent, through the use of *certificates* issued by *trusted* third-party organizations/corporations (credential issuing authorities). The idea behind PKI is that these credential issuing authorities are required to perform due diligence verification before issuing such certificates. Even though this has worked out good for the most part, the tremendous growth in online business transaction has added a new dimension to this whole set-up. In 2003, FBI announced a new trend called *identity-theft* where 30,000 identities where stolen from credit bureaus. The estimated loss was to the tune of millions of dollars.

Giving the user direct control over his personal information — identity, credit card/financial data, credential — would be an obvious solution to such problems. Also, this would change his role from an average user to a willing consumer.

Single Sign-On (SSO) and e-wallet (think Passport® Sign-in and Passport® Express Purchase) are a step in this direction. Liberty Alliance — involving some of the bigwigs of the industry like Sun Microsystems, IBM, Oracle, Nokia, HP. RSA, American

Express, VeriSign among others — is another such noticeable initiative towards digital identity management.

The *Persona* model intends to improve upon SSO/PKI by giving greater control to the users over their private data while distributing the task of credential issuance, allowing any organization to authorize her employee to carry out business transaction on her behalf. This would facilitate on-the-fly credential issuance.

### 3.2 Persona Data

The persona is intended to be used a single repository of the owners private data. The idea is to offer a seamless and secure way to manage ones personal information, facilitating easier online transactions. At the same time, the owner is not obligated to trust a third party corporation (like Microsoft® in case of Passport®) for the same. A typical persona might contain, among other things:

- Identity information: legal name, aliases, photo
- Authentication Data: Passwords, Certificates, Credentials
- Profile Data: contact info, Email, fax number, SMS number
- E-wallet information: Credit/Debit card numbers, bank account numbers
- Preferences: forwarding rules, filter criteria, auto-fill options

Other: favorites, air mile accounts, membership information

## 4. SAML Basics

### 4.1 Definition

SAML is a standard way of exchanging security and related data across heterogeneous, distributed systems crossing domain (geographical, namespace, temporal, spatial, organizational, *etc*.) boundaries

It is an XML-based framework for exchanging security information over the internet.

### 4.2 Assertions

Assertions are ***declarations*** of fact about a subject, which could be either a human or program, according to someone. Typically, such an assertion is produced on a need-to-know basis, as and when requested by a third party. This third party could be referred to as the Web Service Provider (WSP).

The assertion is basically is credential, encapsulating information, about the subject, needed to carry out business transaction. The entity producing the assertion is referred to as the credential issuance authority. In keeping with the distributed, open architecture envisaged as per the *persona* model, anybody — whether is a bank, a university, a corporation, or an individual — can act as the credential issuance authority. The only requirement would be for such an authority to possess its own X509 certificate/digital signature. This is to ensure that WSP can verify the

legitimacy as well as the identity of the issuance authority, if it so desires. We should realize that checking the identity of the issuance authority is an optional step for added security/reliability on part of the said WSP.

A template for a typical *Assertion* document is presented on the following pages. Clearly, this is a XML structured text file, only with standardized tags (instead of the typical XML document that can contain *custom* tags). The standardization helps make it uniform and universal, thus making it acceptable across the board by everyone involved, whether it be an individual or a corporation/organization.

- ***SAML*** Assertions CAN be *Digitally Signed.*
- ***Assertions*** are issued by "Issuing Authorities". At present, there are in general TWO types of Authorities:
    1. The First Type includes *Third-Party Service Providers*, as:
        - Microsoft through PASSPORT
        - XNSORG through Web Identity platform
        - DotGNU through Virtual Identity platform.
    2. The Second Type includes *Individual Businesses* as:
       AOL, American Express, VISA, which can serve as issuing authorities within private federations by leveraging the Liberty Alliance Technologies

```
<Assertion        Majorversion = integer MinorVersion = Integer
                  AssertionID = IDType Issuer = string
                  IssueInstant = dateTime >

        <Conditions       Notbefore = dateTime  NotAfter dateTime >
                <AudiendeRestrictionCondition>
                    <Audience>  AnyURI  </Audience>
                </AudiendeRestrictionCondition>
        </Conditions>
        <Advice>
            <AssertionIDReference>
                IDReferencetype (String)
            </AssertionIDReference>

                    Or

            <Assertion>
                :
                :<<Embedded Assertion>>
                :
            </Assertion>

                    Or

            <<Any other Content>>
        </Advice>

        Or
        <AuthenticationStatement          AuthenticationMethod = AnyURI
                                          AuthenticationInstant = dateTime>
        </AuthenticationStatement>
        Or
        <AuthorizationDecisionStatement     Resource = AnyURI
                                            Decision = Permit/Deny/Indeterminate>
        </AuthorizationDecisionStatement>
        Or

        <AttributeStatement>
        </AttributeStatement>
        <ds:Signature> XML Digital Signature for assertion
        </ds:Signature>
</Assertion>
```

The various tags have been explained below.

- **<Assertion>**

  This is the document tag, as the name suggests, that encloses all other tags embedded in the document.

| Attribute Name | Attribute Value Type | Purpose | Attribute Value Example |
|---|---|---|---|
| MajorVersion | integer | The major version of the assertion. | 0100 |
| MinorVersion | integer | The minor version of the assertion. | 0000 |
| AssertionID | IDType | A unique identifier for an assertion | {EE52CAF4-4ebe-84D3-4D372C892A5D} |
| Issuer | String | The name of the issuer of the assertion as a string. | www.example.com |
| IssueInstant | dateTime | The time instant at which the assertion has been issued. | 2003-05-01T15:00:45Z |

- **<Conditons>**

  This tag contains information about the conditions under which this particular *Assertion* document (as identified by its unique *AssertionID*) is valid.

| Attribute Name | Attribute Value Type | Purpose | Attribute Value Example |
|---|---|---|---|
| NotBefore | dateTime | Specifies the earliest time instant at which the assertion is valid. The time value is encoded in UTC. | 2003-05-01T 15:00:45Z |
| NotAfter | dateTime | Specifies the time instant at which the assertion has expired. The time value is encoded in UTC. | 2003-05-15T 15:00:45Z |

- **<AudiendeRestrictionCondition>**

  The embedded **<AudiendeRestrictionCondition>** is contains advisory information that should be heeded to when processing this assertion document. Within its embedded The embedded **<Audiende>** tag, it contains a string specifying the name of the audience. Typically, it would be a domain name or a URL.

## 4.3 Types of Assertion

As shown in the assertion document template above, every assertion is a compound document that can contain one or more embedded *assertion statements*. The *assertion statement* is the meaty part of any assertion document. This is what contains the

actual useful information/facts about the subject as stated by the credential issuance authority on request by the WSP.

Based on the use-cases designed for SAML, there are *three* kinds of *"statements"* about *"subjects"* (human or program):

**(a.) Authentication Statement**:

A credential issuing authority asserts that subject S was authenticated by means M at time T. Here the credential issuing authority declares that it has verified the identity of the subject using its custom authentication mechanism, either through username/password, or exchange of *X509* certificates. The WSP has a trust relationship with the issuance authority. Not only does it save resources for the WSP —since it does not need its very own authentication system in place —, it also frees the user from remembering multitude of username/password pairs. A most general and common scenario involves an individual (say a professor at OSU) — the *subject* — acting on behalf of the university to carry out online with a WSP transaction. The university (here, OSU) acts as the issuance authority, certifying individual and the authority bestowed upon him by the organization for such purposes.

- Targeted towards Single Sign On uses
- *Note*: Actually checking or revoking of credentials is not in scope for SAML!
- It merely lets you link back to acts of authentication that took place previously

A template *Authentication Assertion* is presented on the next page for reference. As is clearly evident from the template, it declares the occurrence of an authentication exercise was carried out at an earlier point in time, for the benefit of the WSP.

```
<AuthenticationStatement          AuthenticationMethod = AnyURI
                                  AuthenticationInstant = dateTime>

    <Subject>

        <NameIdentifier NameQualifier = String Format = AnyURI>  </NameIdentifier >

        <SubjectConfirmation>

            <ConfirmationMethod> AnyURI </ConfirmationMethod>

            <SubjectConfirmationData>  AnyType </SubjectConfirmationData>

            <ds:KeyInfo>  </ds:KeyInfo>


        </SubjectConfirmation>

            Or

        <SubjectConfirmationData>  AnyType </SubjectConfirmationData>


    </Subject>

    <SubjectLocality IPAddress = String  DNSAddress = String>
    </SubjectLocality>

    <AuthorityBinding     AuthorityKind = Qname
                          Location = AnyURI
                          Bindings = AnyURI>
    </AuthorityBinding>

</AuthenticationStatement>
```

The various tags are explained as below.

- **[<AuthenticationStatement>**

  This is the outermost enclosing tag, declaring its an *Authentication* statement.

| Attribute Name | Attribute Value Type | Purpose | Attribute Value Example |
|---|---|---|---|
| AuthenticationMethod | AnyURI | A URI reference that specifies the type of authentication that took place. The `attribute` indicates how that authentication was done. Note that the authentication statement does not provide the means to perform that authentication, such as a password, key, or certificate. | **[1.] URI:** urn:ietf:rfc:2246 **[2.] URI: Error! Reference source not found.**am:X509-PKI **[3.] URI: Error! Reference source not found.**am:PGP **[4.] URI:** urn:ietf:rfc:3075 **[5.] URI: Error! Reference source not found.**am:XKMS |
| AuthenticationInstant | dateTime | | |

- **<Subject>**
  This tag contains information about the subject of the *Authentication* assertion statement, i.e. the person or program who was authenticated at an earlier time.
  - **<NameIdentifier>**
  - **<SubjectConfirmation>**
  - **<ConfirmationMethod>**

    It is a part of the `<SubjectConfirmation>`element, which is used to allow the relying party to confirm that the request or message came from a system entity that corresponds to the subject in the statement. The `<ConfirmationMethod>` element indicates the method that the relying party can use to do this in the future. This may or may not have any relationship to an authentication that was performed previously. Unlike the authentication method, the subject confirmation method may be accompanied by some piece of information, such as a certificate or key, that will allow the relying party to perform the necessary check.

  - **<SubjectConfirmationData>**

- **<ds:KeyInfo>**

- **<SubjectLocality>**
  This tag specifies the location of the credential issuing authority. Here the credential issuing authority is the entity (corporation/organization) that authenticated the subject of this assertion at an earlier time. Its includes the issuing authority's IP address and its DNS.
  The requesting authority has a *trust* relationship with the issuing relationship, hence accepts the issuing authority's claim of having authenticated the subject. Actually making sure if the issuing authority did carry out the claimed authentication process is beyond the *scope* of SAML.

| Attribute Name | Attribute Value Type | Attribute Value Example |
|---|---|---|
| IPAddress | String | 65.20.44.98 |
| DNSAddress | String | 65.20.44.0 |

**(b.)*Attribute Statement :***
A credential issuing authority asserts that Subject S is associated with attributes A, B, C with values "a", "b", "c". Once the WSP receives the attribute values associated with the subject, its uses it to decide whether or not the user is allowed to carry out a particular transaction.
In this case, the WSP is acting as a *policy* decision point. The WSP is using the information provided by the credential issuing authority—based on the WSP company policy, in relation to the issuing authority's policies that the user represents for the said business transaction — to perform a business deal with the user.
.

- Useful for distributed transactions and authorization services
- Typically this would be gotten from an LDAP repository
  1. *"john.doe"* in "example.com"
  2. is associated with **attribute** *"Department"*
  3. with **value** *"Human Resources"*

A template *Attribute Assertion* is presented on the next page for reference.

```
<AttributeStatement>
    <Subject>
        <NameIdentifier NameQualifier = String Format  = AnyURI>   </NameIdentifier >

        <SubjectConfirmation>

            <ConfirmationMethod> AnyURI </ConfirmationMethod>

            <SubjectConfirmationData>  AnyType </SubjectConfirmationData>

            <ds:KeyInfo>  </ds:KeyInfo>


        </SubjectConfirmation>

        Or

        <SubjectConfirmationData>  AnyType </SubjectConfirmationData>

    </Subject>

    <Attribute AttributeName = String AttributeNameSpace = AnyURI>

    <AtributeValue>
       AnyType
    </AtributeValue>

    </Attribute>
</AttributeStatement>
```

**(c.) *Authorization Statement:***

A credential issuing authority decides whether to grant the request by subject S for access type A to resource R given evidence E. In this case, the credential issuing authority acts as the *policy decision* point based on the evidence offered by the WSP. This evidence could be a reference ID of a previous assertion or an entire assertion itself, embedded in the request. The WSP just acts as the *policy enforcement* point, based on the decision returned by the credential issuing authority. Here, the security policy resides with the issuing authority.

- Useful for distributed transactions and authorization services
- The subject could be a human or a program
- The resource could be a web page or a web service, for example

A template *Authentication Assertion* is presented on the next page for reference.

<AuthorizationDecisionStatement Resource = *AnyURI*
Decision = *Permit/Deny/Indeterminate*>
<Subject>
<NameIdentifier NameQualifier = *String* Format = *AnyURI*> </NameIdentifier >

<SubjectConfirmation>

<ConfirmationMethod> *AnyURI* </ConfirmationMethod>

<SubjectConfirmationData> *AnyType* </SubjectConfirmationData>

<ds:KeyInfo> </ds:KeyInfo>

</SubjectConfirmation>

**OR**

<SubjectConfirmationData> *AnyType* </SubjectConfirmationData>
</Subject>
<Action Namespace = *AnyURI*>
*String*
</Action>
<Evidence>

<AssertionIDReference>

*IDReferenceType (String)*

</AssertionIDReference>

**Or**

<Assertion>
:
:<<*Embedded Assertion*>>
:
</Assertion>

</Evidence>

</AuthorizationDecisionStatement>

## 4.4 SAML Protocol

SAML is based on a *request-response* protocol for exchange of security information. The WSP is usually the entity that *requests* information from a credential issuing authority about a particular subject. The issuing authority then *responds* to the request with an assertion statement of the appropriate type.
There are *three* different types of request:

- Authentication Assertion Request : *" What authentication assertions are available for this subject"*
- Attribute Assertion Request : *" Return the requested attributes for this subject"*
- Authorization Decision Request: *"Is this subject allowed to access the specified resource in the specified manner, given this evidence?"*

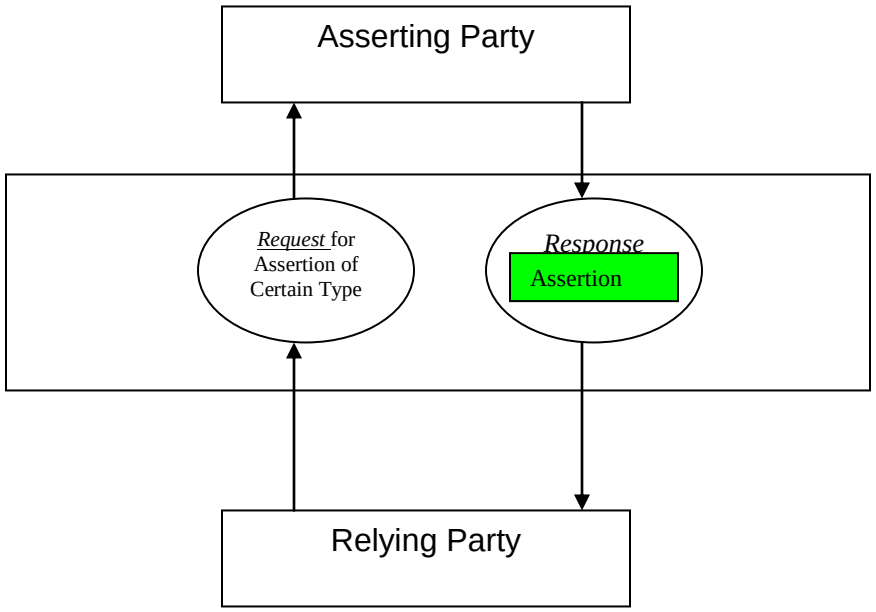



Figure 2: SAML Protocol

As the figure clearly shows, the *requester* sends a request for an assertion of a particular type. The credential issuing authority, also called the *asserting* party, responds with the corresponding assertion statement type. Since the requester is using the information returned by the issuing authority, it is acting as the *relying* party in this scenario.

## 4.5 Request Assertion Document Template

A template for a typical *request* assertion is presented below. As shown, the request document can contain one of the three possible types of assertion request type. The **<RespondWith>** tag tells the asserting party about the type of assertion statement that the relying party expects in response. This tag can have one of three possible values — **AutheticationStatement, AttributeStatement, AuthorizationDecisionStatement**

The **<AssertionIDReference>** tag contains the ID for the assertion document that the issuing authority can user as reference (evidence) when issuing the requested assertion statement.

**<Request** RequestID = *IDType* MajorVersion = *Interger*
MinorVersion = *Interger* IssueInstant = *dateTime*>

<RespondWith>
*Qname*
</RespondWith>

**<ds:Signature>**
***XML Digital Signature for assertion***
**</ds:Signature>**

<AuthenticationQuery AuthenticationMethod = *AnyURI*>

</AuthenticationQuery>

**Or**

<AuthorizationDecisionQuery Resource = *AnyURI*>

</AuthorizationDecisionQuery>

**Or**

<AttributeQuery Resource = *AnyURI*>

</AttributeQuery>

<AssertionIDReference>

*IDReferenceType (String)*

</AssertionIDReference>

<AssertionArtifact>
*String*
</AssertionArtifact>
**</Request>**

### 4.5.1 Authentication Query template

The relying party requests the asserting party about the authentication method the latter used to authenticate the subject at an earlier time. As shown, the query only contains the subject.
The *response* to this query is an *authentication statement* embedded inside a response document. The authentication statement, as expected, contains information about the authentication method used — like username/password, exchange of *X509 certificates*.

A sample *Authentication* query template is presented below for reference.

```
<AuthenticationQuery AuthenticationMethod = AnyURI>

    <Subject>
        <NameIdentifier NameQualifier = String Format  = AnyURI>   </NameIdentifier >

        <SubjectConfirmation>

          <ConfirmationMethod> AnyURI </ConfirmationMethod>

          <SubjectConfirmationData>  AnyType </SubjectConfirmationData>

          <ds:KeyInfo>  </ds:KeyInfo>


        </SubjectConfirmation>

        OR

        <SubjectConfirmationData>  AnyType </SubjectConfirmationData>

    </Subject>

</AuthenticationQuery>
```

### 4.5.2 Authorization Decision Query template

The relying party requests the asserting party whether is allowed to carry out a particular business transaction on the latter's behalf. Again, as shown, the query only contains the subject.
The *request* contains either an assertion ID or an actual embedded assertion as evidence — **<Evidence>** tag — that the asserting/issuing party can use to make its decision.

The *response* to this query is an *authorization decision statement* embedded inside a response document. The authorization decision for an action can have one of three possible values — *permit/deny/execute*.

A sample *Authorization Decision* query template is presented below for reference.

```
<AuthorizationDecisionQuery Resource = AnyURI>
    <Subject>
        <NameIdentifier SecurityDomain="xyz.org" Format= ”#X509SubjectName”>
            Email=user@xyz.org CN=Portland ST=OR C=USA
        </NameIdentifier >

        <SubjectConfirmation>
        <ConfirmationMethod> AnyURI </ConfirmationMethod>
        <SubjectConfirmationData>  AnyType </SubjectConfirmationData>
        <ds:KeyInfo>  </ds:KeyInfo>
        </SubjectConfirmation>

            OR

        <SubjectConfirmationData>  AnyType </SubjectConfirmationData>

    </Subject>
    <Action Namespace = “http://www.wisenet.org”>
            Execute
    </Action>
    <Evidence>
        <AssertionIDeference> IDReferenceType (String)  </AssertionIDReference>
                Or
        <Assertion>
            : <<Embedded Assertion>>
            :
        </Assertion>

    </Evidence>
</AuthorizationDecisionQuery>
```

### 4.5.3 Attribute Query template

The relying party requests the asserting party for attribute values for a set of attribute for the subject. The subject represents the asserting party in the business transaction. Typically, the asserting party is an organization (like OSU) and the subject is its employee. On receiving such a request, the asserting party — also, the credential issuing authority — responds with the values for the requested attributes. Based on the values for the attributes, the *relying* party uses its security policy to decide whether or not the user has necessary rights to carry out the transaction on behalf of his/her organization.

The *response* to this query is an *attribute statement* embedded inside a response document.

A sample *Attribute* query template is presented below for reference.

```
<AttributeQuery Resource = AnyURI>
    <Subject>
        <NameIdentifier NameQualifier = String Format  = AnyURI>   </NameIdentifier >

        <SubjectConfirmation>

          <ConfirmationMethod> AnyURI </ConfirmationMethod>

          <SubjectConfirmationData>  AnyType </SubjectConfirmationData>

          <ds:KeyInfo>  </ds:KeyInfo>


        </SubjectConfirmation>

        OR

        <SubjectConfirmationData>  AnyType </SubjectConfirmationData>

    </Subject>

    <AttributeDesignator AttributeName = String AttributeNameSpace = AnyURI>
    </AttributeDesignator >

</AttributeQuery>
```

## 4.6 Response Assertion Document Template:

A response document contains the particular assertion statement type requested by the relying party.

The *<<ResponseType>>* is a place holder. It can be one of three possible assertion statements: *AuthenticationStatement*, *AttributeStatement*, or *AuthorizationDecisionStatement*

A template for a *response* document is presented below for reference.

```
<Response       ResponseID = IDType
                InResponseTo = IDReference
                MajorVersion = Interger
                MinorVersion = Interger
                IssueInstant = dateTime
                Recipient = AnyURI>

    <ds:Signature>

    </ds:Signature>

    <Assertion  MajorVersion="1"
                MinorVersion="0"
                AssertionID="124.456.7456"
                Issuer="wisenet.org">

        <Conditions     NotBefore="2003-5-14T10:00:20Z"
                        NotBefore="2003-5-14T10:00:50Z">
        </Conditions>
    <ResponseType> ….. </ResponseType>

    </Assertion>

</Response>
```

# 5. Actual Assertion Examples

A hypothetical authentication transaction between www.thewisenet.com and http://owl.een.orst.edu is used for the following example assertions. Here the www.thewisenet.com is the credential issuing authority (*a.k.a* asserting party) and http://owl.een.orst.edu acts as the relying party.

## 5.1 Authentication Assertion Example

```
<Assertion      Majorversion = 1
                MinorVersion = 1.1
                AssertionID = "186CB370-5C81-4716-8F65-F0B4FC4B4A0B"
                Issuer = "www.thewisenet.com"
                IssueInstant = "2003-05-19T18:26:00" >

    <Conditions  Notbefore = "2003-05-19T18:26:00 "
                 NotOnOrAfter = "2003-05-19T18:40:00"/>

        <AudienceRestrictionCondition>

            <Audience> http://owl.een.orst.edu  </Audience>

        </AudienceRestrictionCondition>

    </Conditions>
    <AuthenticationStatement
                    AuthenticationMethod                                    =
                    "urn:oasis:names:tc:SAML:1.1:am:X509-PKI"
                    AuthenticationInstant = "2003-05-19T18:20:00">

        <Subject>

       <NameIdentifier   NameQualifier = "thewisenet.com"
                         Format  = "#X509SubjectName" >
       Email = ktoth@thewisenet.com CN = thewisenet.com ST =   C =
       </NameIdentifier >

            <SubjectConfirmation>
              <ConfirmationMethod>
                urn:oasis:names:tc:SAML:1.1:am:X509-PKI

              </ConfirmationMethod>
              <SubjectConfirmationData>
              </SubjectConfirmationData>

              <ds:KeyInfo> </ds:KeyInfo>
            </SubjectConfirmation>
        </Subject>
        <SubjectLocality IPAddress = "198.162.32.16" DNSAddress =
        "192.162.32.0">
        </SubjectLocality>

        <AuthorityBinding     AuthorityKind = "thewisenet.com"
                              Location = "www.thewisenet.com/signon"
                              Bindings = anyURI>
        </AuthorityBinding>
    </AuthenticationStatement>
    <ds:Signature>
            XML Digital Signature for assertion
    </ds:Signature>
</Assertion>
```

## 5.2 Attribute Assertion Example

```
<Assertion      Majorversion = 1
                MinorVersion = 1.1
                AssertionID = "186CB370-5C81-4716-8F65-F0B4FC4B4A0B"
                Issuer = "www.thewisenet.com"
                IssueInstant = "2003-05-19T18:26:00" >

    <Conditions  Notbefore = "2003-05-19T18:26:00 "
                 NotOnOrAfter = "2003-05-19T18:40:00"/>

        <AudienceRestrictionCondition>
            <Audience> http://owl.een.orst.edu  </Audience>
        </AudienceRestrictionCondition>

    </Conditions>

    <AttributeStatement>
        <Subject>

            <NameIdentifier NameQualifier = "thewisenet.com" Format  =
                    "urn:oasis:names:tc:SAML:1.1:nameid-format:
            X509SubjectName">
            Email = ktoth@thewisenet.com CN = thewisenet.com ST =   C =
            </NameIdentifier >

            <SubjectConfirmation>

              <ConfirmationMethod>
                urn:oasis:names:tc:SAML:1.1:am:X509-PKI
              </ConfirmationMethod>

              <SubjectConfirmationData>
              </SubjectConfirmationData>

              <ds:KeyInfo> </ds:KeyInfo>
            </SubjectConfirmation>
        </Subject>

        <Attribute          AttributeName = SecurityLevel
                            AttributeNameSpace = "http://owl.een.orst.edu">

            <AttributeValue>
               TopSecret
            </AttributeValue>

        </Attribute>
     </AttributeStatement>

<ds:Signature>
  XML Digital Signature for assertion
</ds:Signature>
</Assertion>
```

## 5.3 Authorization Decision Assertion Example

```
<Assertion      Majorversion = 1
                MinorVersion = 1
                AssertionID = <some random generated string used for the transaction>
                Issuer = "www.thewisenet.com"
                IssueInstant = "2003-05-19T18:26:00" >

    <Conditions  Notbefore = "2003-05-19T18:26:00 "
                 NotOnOrAfter = "2003-05-19T18:40:00"/>

        <AudienceRestrictionCondition>
            <Audience> http://owl.een.orst.edu  </Audience>
        </AudienceRestrictionCondition>
    </Conditions>

    <AuthorizationDecisionStatement      Resource = "http://owl.een.orst.edu/buy.html"
                                         Decision = "Permit">

        <Subject>

       <NameIdentifier  NameQualifier = "thewisenet.com" Format =
       "X509SubjectName">
                Email = ktoth@thewisenet.com CN = thewisenet.com  ST = OR  C = USA
       </NameIdentifier >
            <SubjectConfirmation>
              <ConfirmationMethod>
                urn:oasis:names:tc:SAML:1.1:am:X509-PKI
              </ConfirmationMethod>

              <SubjectConfirmationData>  … </SubjectConfirmationData>

              <ds:KeyInfo> </ds:KeyInfo>
            </SubjectConfirmation>
        </Subject>

        <Action Namespace = http://owl.een.orst.edu>
                buy
        </Action>
    </AuthorizationDecisionStatement>
    <ds:Signature>
       XML Digital Signature for assertion
     </ds:Signature>

</Assertion>
```

# 6. A Sample Scenario

A complete *sample* scenario – derived from the ***Persona Project*** – with SAML fragments *et. Al.* :

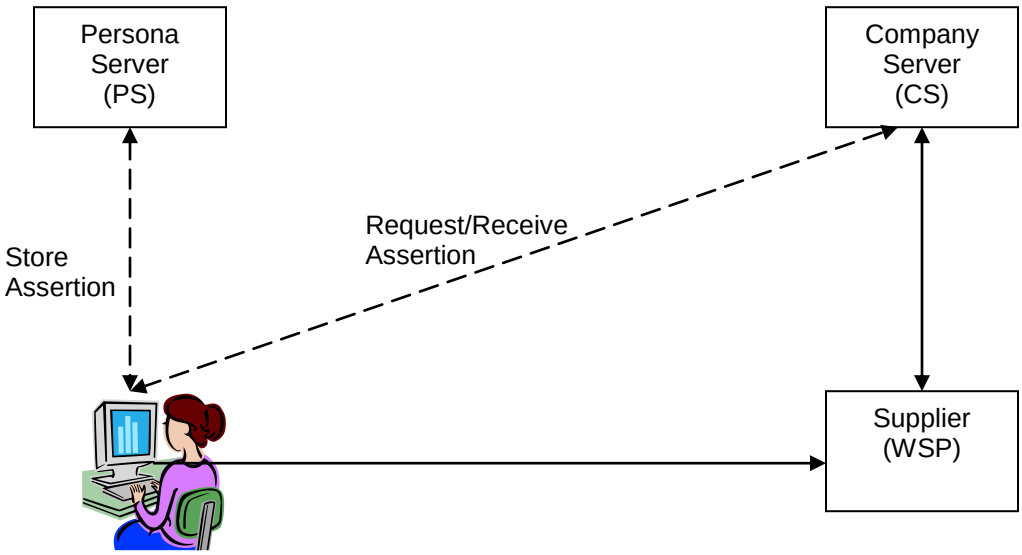



Figure 3: A Sample Persona Scenario

In-band Communication

Out-of-band Communication

**Out-of-band Communication**

**Step 1:** *User* goes to the company server (CS), identifies himself (either through a password mechanism or through the use of *X509* certificates.)

**Step 2:** *User* requests for an assertion (*Authorization Decision Assertion*) from the *CS*.

**(a)** To obtain an *"Assertion"* that would allow him/her to perform monetary transactions (purchases *et. al.*) on behalf of the company, he/she must provide the necessary information.
**Note:** The required information can also be obtained automatically by the SAML application/processor from the database record for the employee concerned. Remember, as the employee has been authenticated, his identity is known.

**(b)** Based on the information provided (by the user) – or obtained from the database-- , as the case may be, the *CS* will create an *"Assertion"* that would, among other things, assert the following:

**(I)** Buying Power
**(II)** Credit History
**(III)** Buying rights
**(IV)** Validity period for the *"Assertion"*

**(c)** The *"Assertion"* produced would be ***digitally signed*** by the CS.

**(d)** In effect, this *"Assertion"* is like a certificate that certifies the "financial" status of an employee vis-à-vis the company for which he/she works.

**(e)** The ***validity period*** on the said *"Assertion"* is a means to unsure that an assertion, once obtained by an employee (and subsequently stored in his ***Persona Server***, with a copy stored on the *CS*) cannot be used forever.

**Step 3:** *User* obtains the requested *"Assertion"* as a SAML document, (mostly) embedded in a SOAP message.

**Step 4:** *User* stores this *"Assertion"* in his ***Persona Server*** (PS) for later. A copy of the assertion is also stores by the ***Company Server*** (CS) – stateful server.

**Online Transaction**

**Step 1:** *User* goes to the supplier (**WSP**) to perform purchases on behalf of his company.

**Step 2:** *User* and *supplier* identify each other by exchanging their ***X509*** certificates.

**Step 3:** With their respective identities established, the supplier can do one of the following:

**(a)** Request user to provide a ***URL*** to his *"Assertion"*, assuming the WSP is ***persona- aware.***

**(b)** Pops-up a webpage, asking the user to provide information that would be used by the WSP to check for the user's credentials in order to allow ( or disallow) access to protected resources.

**Step 4:** In case of 3(a) above, the WSP simply fetches (pulls) the user *"Assertion"* from his/her ***Persona Server***.

**Step 5:** In case of 3(b) above, the WSP may create an *"Authorization Decision Assertion"* request and send it to the ***Persona Server***. The ***PS***, then, responds with a response assertion.

**Step 6:** In either case, it is, then, up to the supplier (WSP) to contact the CS and establish the validity of the *"Assertion"* (using the CS's ***digital signature*** on it).

> **Note:** This is an optional step. If the user is a returning-user, and the supplier trusts him/her, he can allow user access to protected resources (equivalently, allow him/her to perform monetary transactions/purchases) and its own risk.

**Step 7:** Finally, the supplier (WSP) provides the user with access to the protected resources and allows the transaction to go through.

For the above scenario, the *request* document sent by the WSP to the Persona Server is as follows:

```
<Request        Majorversion = 1   MinorVersion = 1.1
                AssertionID = "186CB370-5C81-4716-8F65-F0B4FC4B4A0B"
                Issuer = "www.thewisenet.com"  IssueInstant = "2003-05-19T18:26:00" >

        <RespondWith>  AttributeStatement   </RespondWith>
            <ds:Signature>
                <SignedInfo>
                <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2000/..." />
                <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                <Reference URI="#PurchaseOrder">
                <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
                <DigestValue>qZk+nkcGcWq6piVxeFdcbJzQ2JO</DigestValue>
                </Reference>
                </SignedInfo>

                <SignatureValue>
                IWijxQjUrcXBYc0ei4QxjWo9Kg8Dep9tlWoT4SdeRT87GH03dgh
                </SignatureValue>
                <KeyInfo>
                  <X509Data>
                  <X509SubjectName>
                  CN=Alice Smith, STREET=742 Park Avenue, L=New York, ST=NY, C=US
                  </X509SubjectName>
                  </X509Data>
                </KeyInfo>
            </ds:Signature>

    <AttributeQuery Resource="supplier.wsp.org/protected.html">
                <Subject>
                    < NameIdentifier        SecurityDomain="www.xyz.org"
                                            Format="#X509SubjectName">
                            Email=user@xyz.org CN=Portland ST=OR C=USA
                     </NameIdentifier>
                < SubjectConfirmation>
                    < ConfirmationMethod> urn:ietf:rfc:1510  </ ConfirmationMethod>
                    < SubjectConfirmationData> A X.509 certificate    </SubjectConfirmationData>
                     <ds:KeyInfo>
                        <ds:X509Data>...</ds:X509Data>
                      </ds:KeyInfo>
                </ SubjectConfirmation>
            </ Subject>

        <AttributeDesignator  AttributeName="Buying Power"  AttributeNamespace="www.xyz.org"/>
        <AttributeDesignator AttributeName="Credit History"   AttributeNamespace="www.xyz.org"/>
        <AttributeDesignator AttributeName="Buying Rights"   AttributeNamespace="www.xyz.org"/>
    </AttributeQuery>
</Request>
```

**Request from the *Supplier* (WSP) to the Persona Server.**

Next, the *two* assertion statement document statements produced by the CS and stored on the user's *Persona* server. The *CS* provides the user with a *Authentication* assertion, encapsulating the authentication method employed to verify the said user's identity. Also, the *CS* produces an *Attribute* assertion document — including information about his/her *credit history, buying power, buying rights*.
When the user contacts the WSP to perform a business transaction on behalf of his/her organization/company, the WSP requests for and receives the *attribute assertion* statement as part of the response document.

The two assertion documents are presented below for reference.

- Authentication Assertion statement document

```
<Assertion      MajorVersion="1"
                MinorVersion="0"
                AssertionID="156.56.104.10.1037385546507"
                Issuer="XYZ Inc."
                IssueInstant="2002-11-15T13:39:06.507-05:00">

    <Conditions     NotBefore="2002-11-15T13:34:06.518-05:00"
                    NotOnOrAfter="2002-11-15T13:49:06.518-05:00">
        <AudienceRestrictionCondition>
            <Audience>http://www.xyz.org/agreement.xml</ns1:Audience>
        </AudienceRestrictionCondition>
    </Conditions>

    <AuthenticationStatement        AuthenticationMethod="urn:ietf:rfc:1510"
                                    AuthenticationInstant="2002-11-15T13:39:06.558-5:00">
        <Subject>
            <NameIdentifier  SecurityDomain="www.gatewayportal.org"
                             Format="#X509SubjectName">
                    Email=user@xyz.org CN=Portland ST=OR C=USA
            </NameIdentifier>
        <SubjectConfirmation>
            <ConfirmationMethod> urn:ietf:rfc:1510 </ConfirmationMethod>
        <SubjectConfirmationData>
                    A X.509 certificate
        </SubjectConfirmationData>
                Or
            <ds:KeyInfo>
                <ds:X509Data>...</ds:X509Data>
            </ds:KeyInfo>
        </SubjectConfirmation>
    </Subject>
    <SubjectLocality    IPAddress="125.125.125.105"
                        DNSAddress="persona.serv.org"/>
    <AuthorityBinding  Location="156.56.104.10" >
</AuthenticationStatement>
</Assertion>
```

- Attribute Assertion statement document

```
<Assertion    MajorVersion="1"
              MinorVersion="0"
              AssertionID="156.56.104.10.1037385546507"
              Issuer="XYZ Inc."
              IssueInstant="2002-11-15T13:39:06.507-05:00">

    <Conditions    NotBefore="2002-11-15T13:34:06.518-05:00"
                   NotOnOrAfter="2002-11-15T13:49:06.518-05:00">

        <AudienceRestrictionCondition>
          <Audience>http://www.xyz.org/agreement.xml </ Audience>
        </AudienceRestrictionCondition>
    </Conditions>
  <AttributeStatement   AuthenticationMethod="urn:ietf:rfc:1510"
                        AuthenticationInstant="2002-11-15T13:39:06.558-05:00">
      <Subject>
          <NameIdentifier  SecurityDomain="www.xyz.org"
                           Format="#X509SubjectName">
                  Email=user@xyz.org CN=Portland ST=OR C=USA
          </NameIdentifier>
      <SubjectConfirmation>
          <ConfirmationMethod> urn:ietf:rfc:1510  </ConfirmationMethod>
      <SubjectConfirmationData>
            A X.509 certificate
      </SubjectConfirmationData>
          <ds:KeyInfo>
            <ds:X509Data>   ...   </ds:X509Data>
          </ds:KeyInfo>
    </SubjectConfirmation>
    </Subject>
    <SubjectLocality IPAddress="125.125.125.105"
                     DNSAddress="persona.serv.org"/>
    <AuthorityBinding  Location="156.56.104.10" />
        <Attribute AttributeName="Buying Power" AttributeNamespace="www.xyz.org">
            <AttributeValue> Platinum </AttributeValue>
    </Attribute>
    <Attribute AttributeName="Credit History" AttributeNamespace="www.xyz.org">
            <AttributeValue>  Excellent  </AttributeValue>
    </Attribute>
    <Attribute AttributeName="Buying Rights" AttributeNamespace="www.xyz.org">
            <AttributeValue> Manager  </AttributeValue>
    </Attribute>
  </AttributeStatement>
</Assertion>
```

# 7. SAML Edit® -- SAML Editor

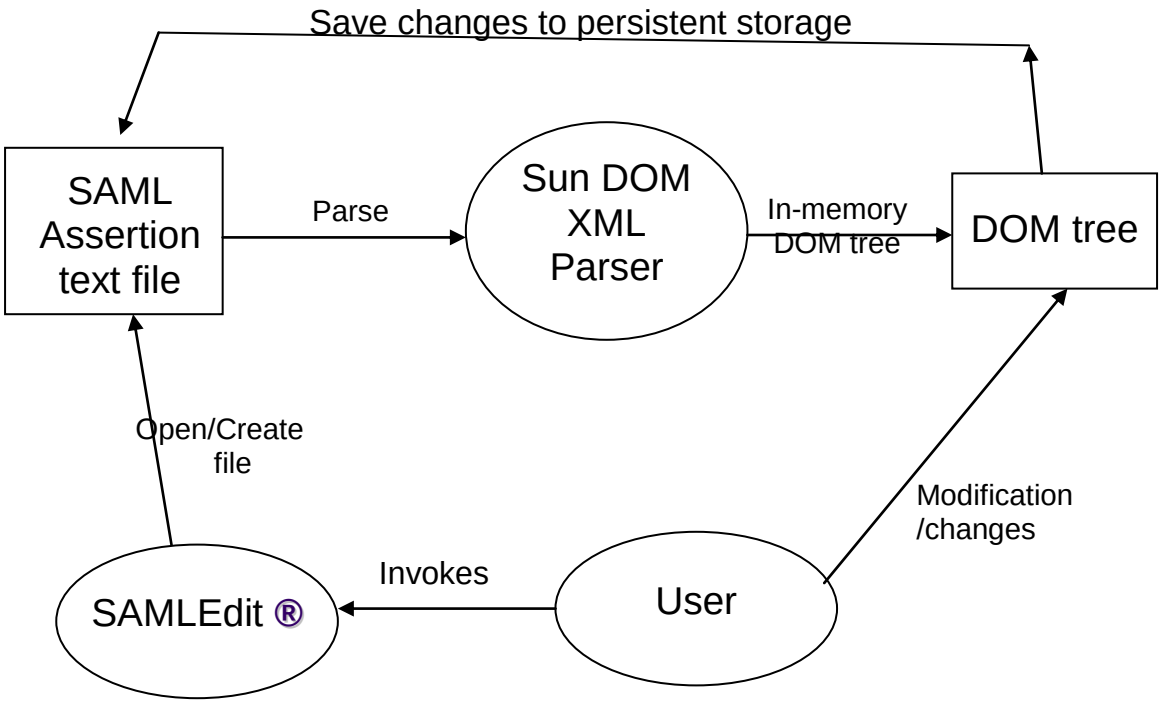
## 7.1 Introduction

The SAML Editor is a Java-based GUI editor that uses the DOM (Document Object Model) parser provided by Sun Microsystems.

## 7.2 Purpose

SAML is an up and coming industry standard for the transfer of security information over the internet. The various types of Assertions offer standard template for this purpose. The editor provides a user friendly interface to the user to manipulate/modify and/or create new SAML Assertions. The User Interface is designed in a way that accommodates both SAML literate and non-SAML literate users.

## 7.3 Design




This is a schematic representation of how a user may invoke and use **SAMLEdit®**. As is evident from the figure, the user invokes the editor, opens an existing (or creates a new file, as the case may be). The file is parsed by the underlying Sun Microsystems® DOM parser. The parsed file is stored in memory as a DOM tree. Any modification made by the user is directly and instantly reflected in the in-memory DOM tree.

The editor offers multiple views to the underlying data in an SAML file:

- The tree-based view in the left pane
- The individual field values are displayed in the adjoining text-pane on the top-right
- The table in the bottom-right pane displays all the attribute/value pairs for the selected tag in the tree-pane
- The menu provides an additional option to view and/or modify the underlying SAML file directly.

## 7.4 Editor Details

The SAML Editor provides a very intuitive interface to the data/information stored in the underlying SAML file. The focus, while designing this editor, was to create a tool that allows both SAML and non-SAML literate users to manipulate SAML Assertion with relative ease.

The following pages provide a detailed look at the various features of the said Editor.

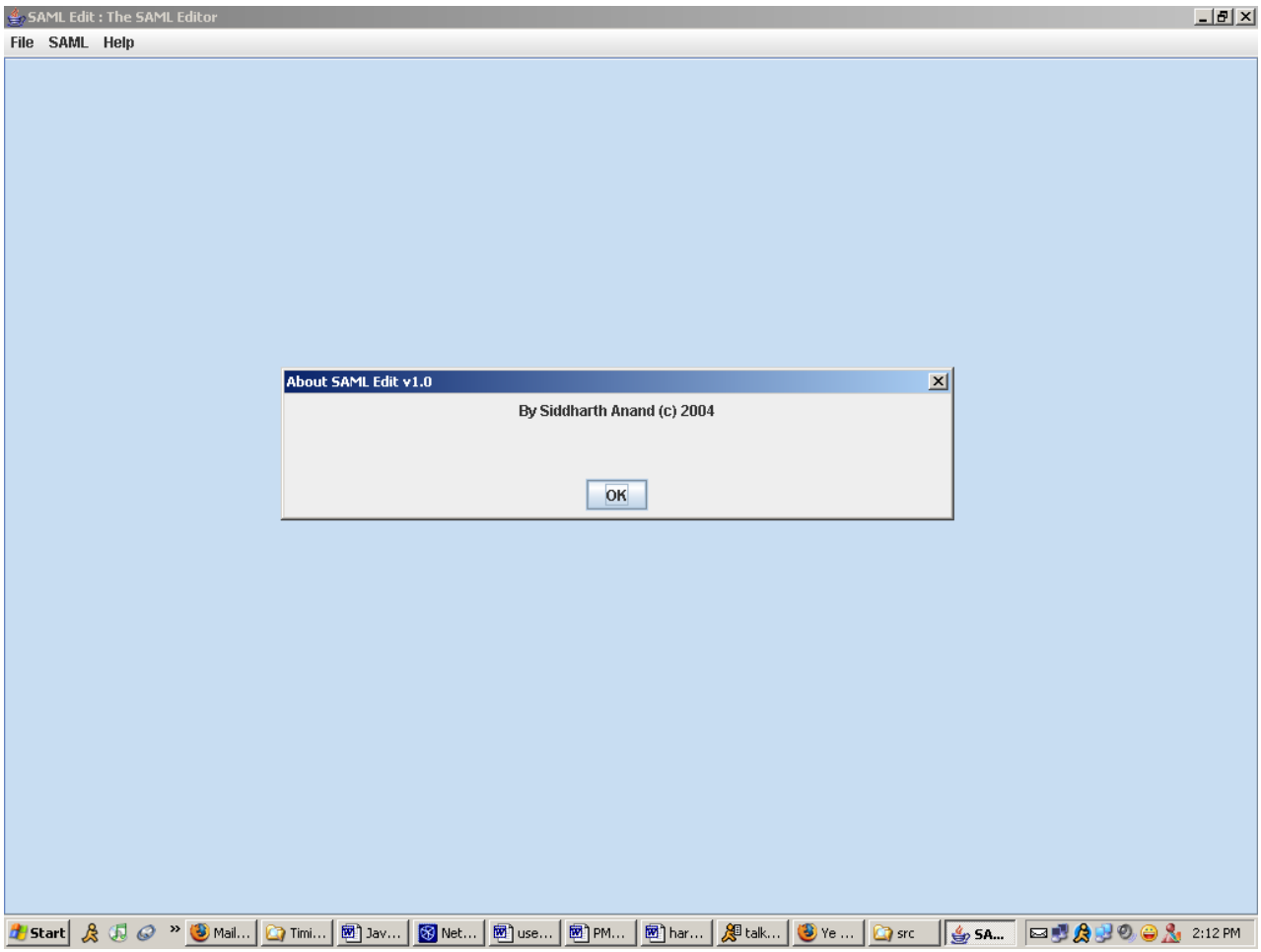
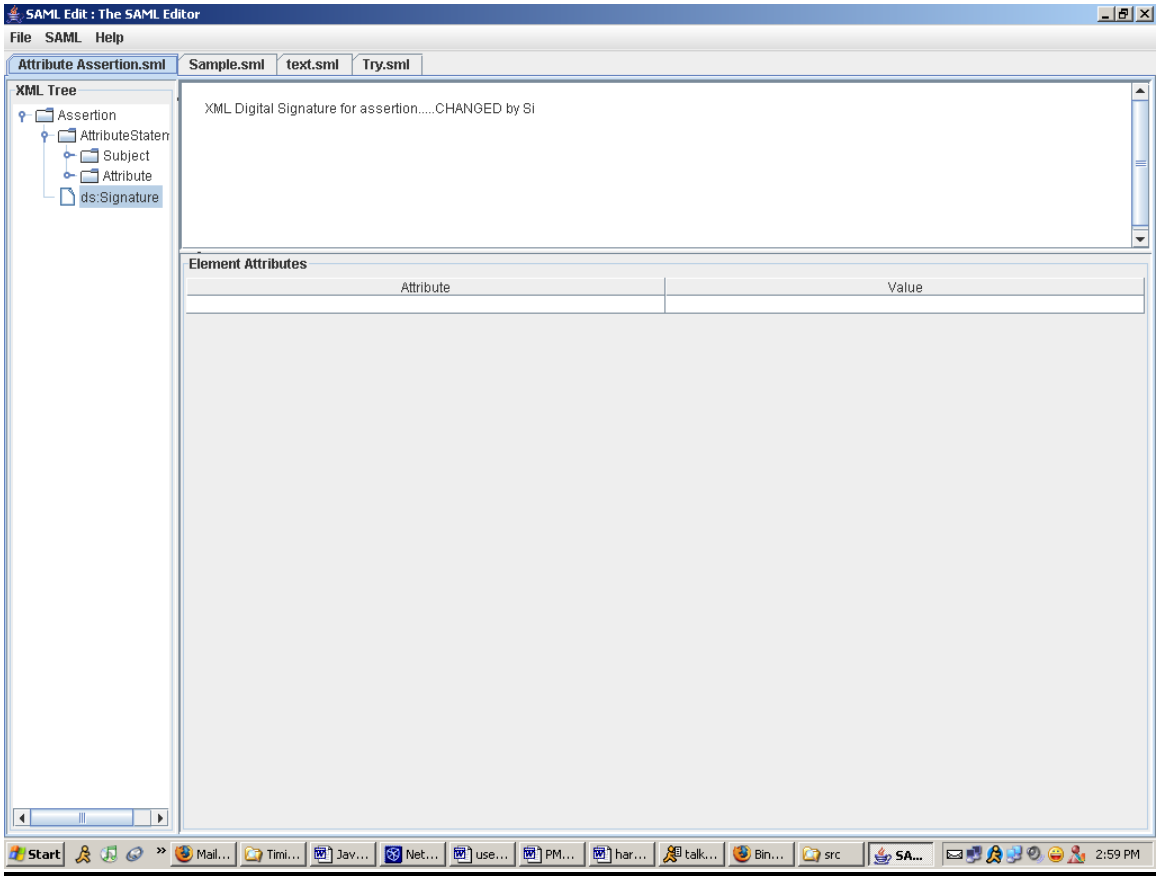
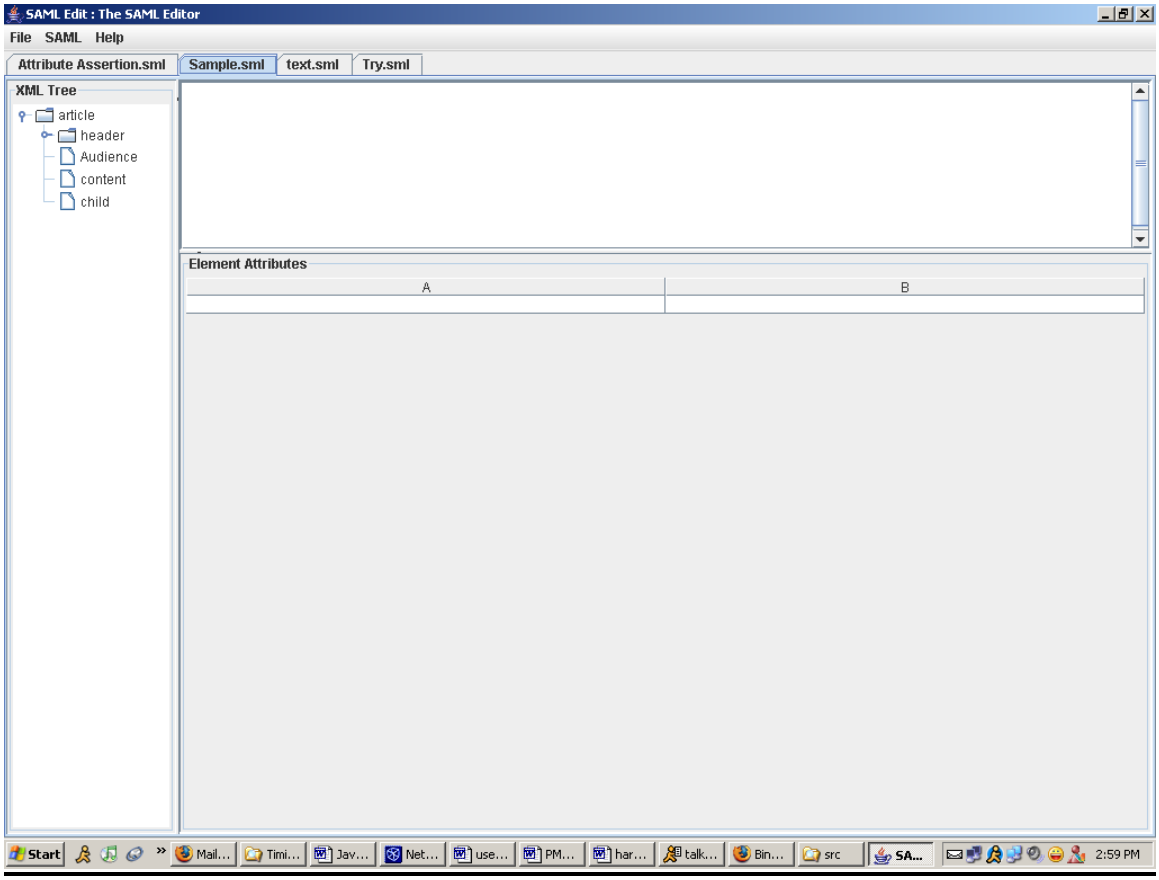


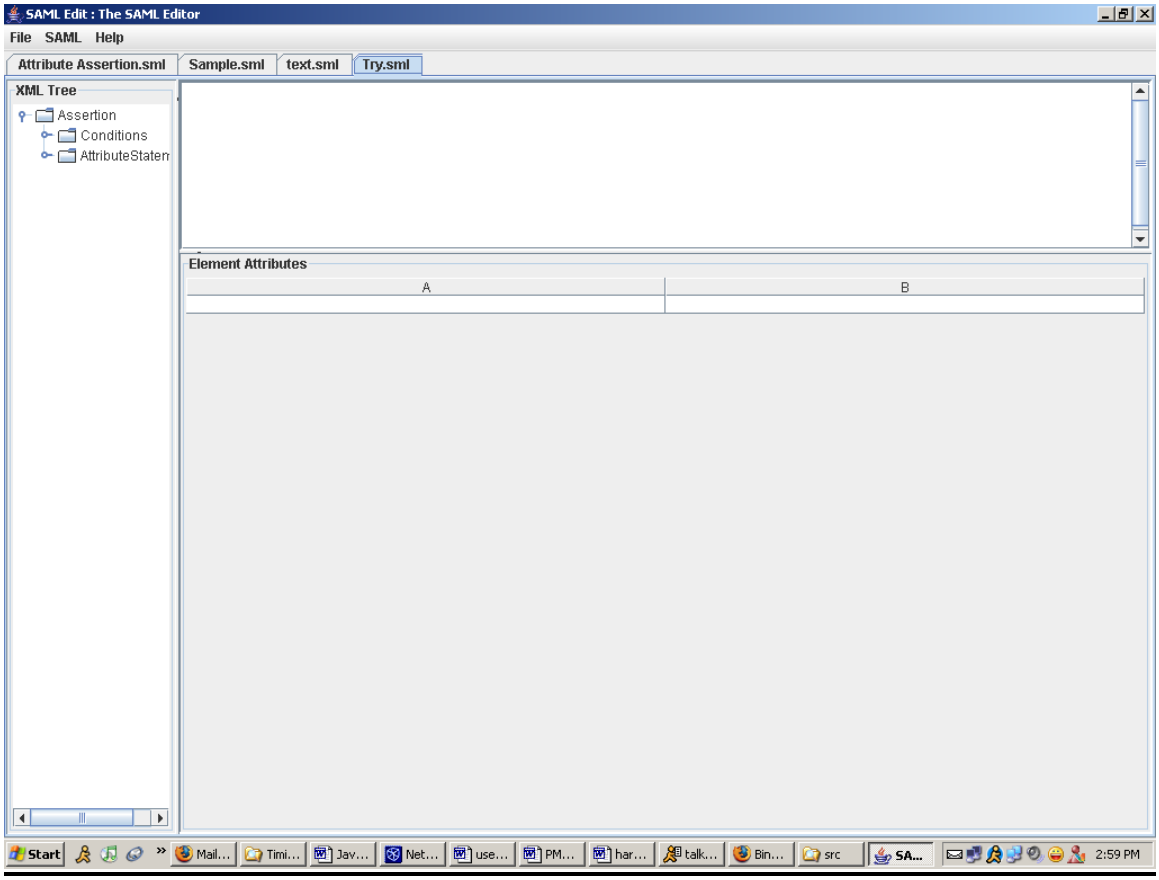

Figure 4: SAML Edit®

### 7.4.1 The User Interface – Multiple Tabs

SAML Edit® allows the user to open and modify multiple SAML Assertion text files simultaneously. This is made possible by using a new, separate tab to display each file. The tab border at the top displays the *file-name* for the file contained in a particular tabbed pane. The selected tabbed pane is highlighted for visual clarity. Using this tabbed-pane approach for multiple files helps keeps the UI un-cluttered and neat.


SAML Edit : The SAML Editor
File SAML Help
Attribute Assertion.sml
Sample.sml
text.sml
Try.sml
XML Tree
Assertion
AttributeStatem
Subject
Attribute
ds:Signature
XML Digital Signature for assertion.....CHANGED by Si
Element Attributes
Attribute
Value
Start
2:59 PM


SAML Edit : The SAML Editor
File SAML Help
Attribute Assertion.sml
Sample.sml
text.sml
Try.sml
XML Tree
article
header
Audience
content
child
Element Attributes
A
B
Start
2:59 PM




Figures 5: Multiple Screen-Shots showing a separate tabbed-pane (file)

## 7.4.2 The User Interface -- Menu Options

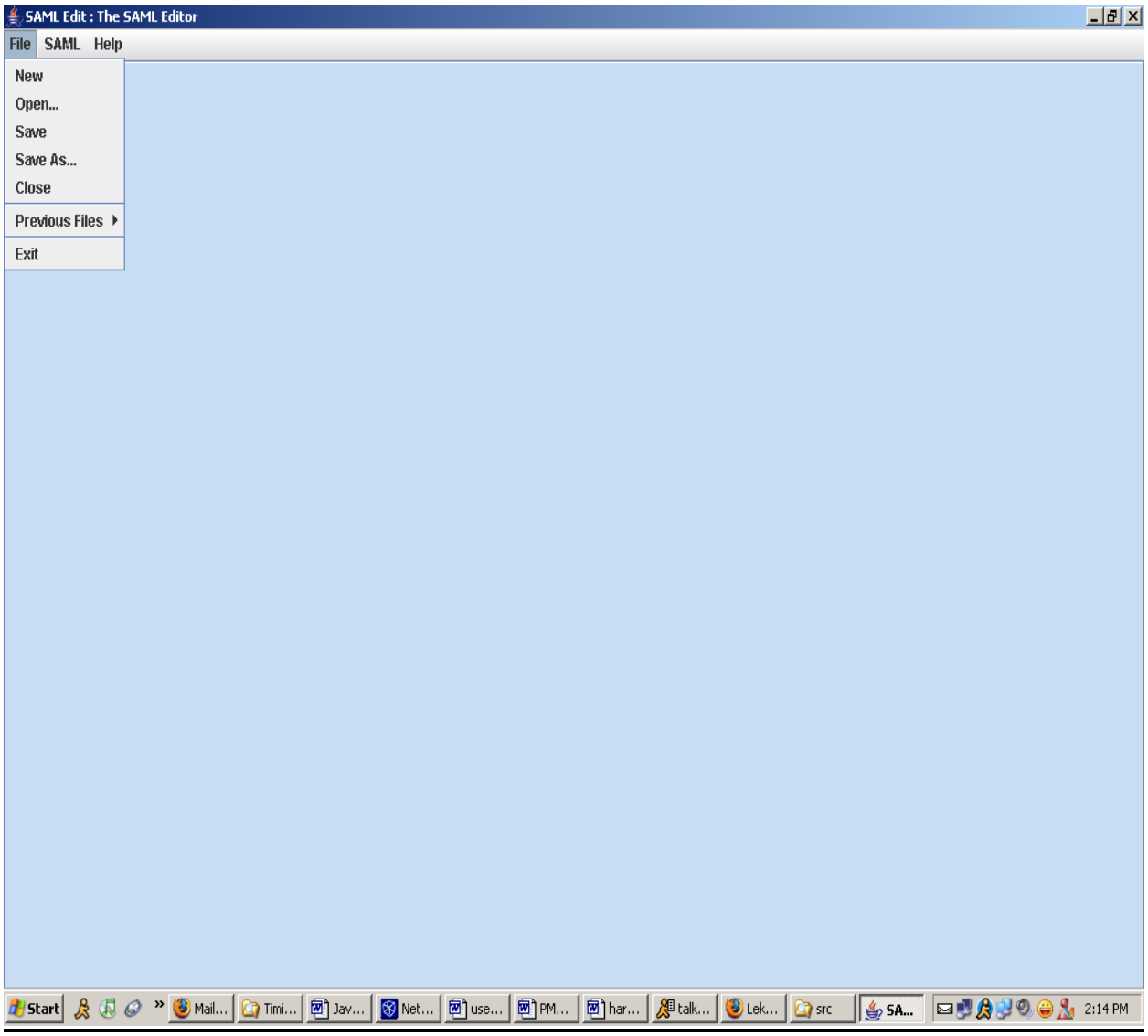



Figure 6: File Menu Options

- **File Menu**
  - **New**

This option allows the User to create new SAML Assertions using the standard templates:

1. **Authentication Assertion Template** → To create a NEW Authentication Assertion as defined by OASIS.

2. **Attribute Assertion Template** → To create a NEW Attribute Assertion as defined by OASIS.

3. **Authorization Decision Assertion Template** → To create a NEW Authorization Decision Assertion as defined by OASIS.

A template file (***.samt*** extension) for each of the here possible categories of Assertions is provided with the editor as a starting point. The User can create *additional* template files based on his needs using the SAML editor itself, as long as he template is well-formed XML.

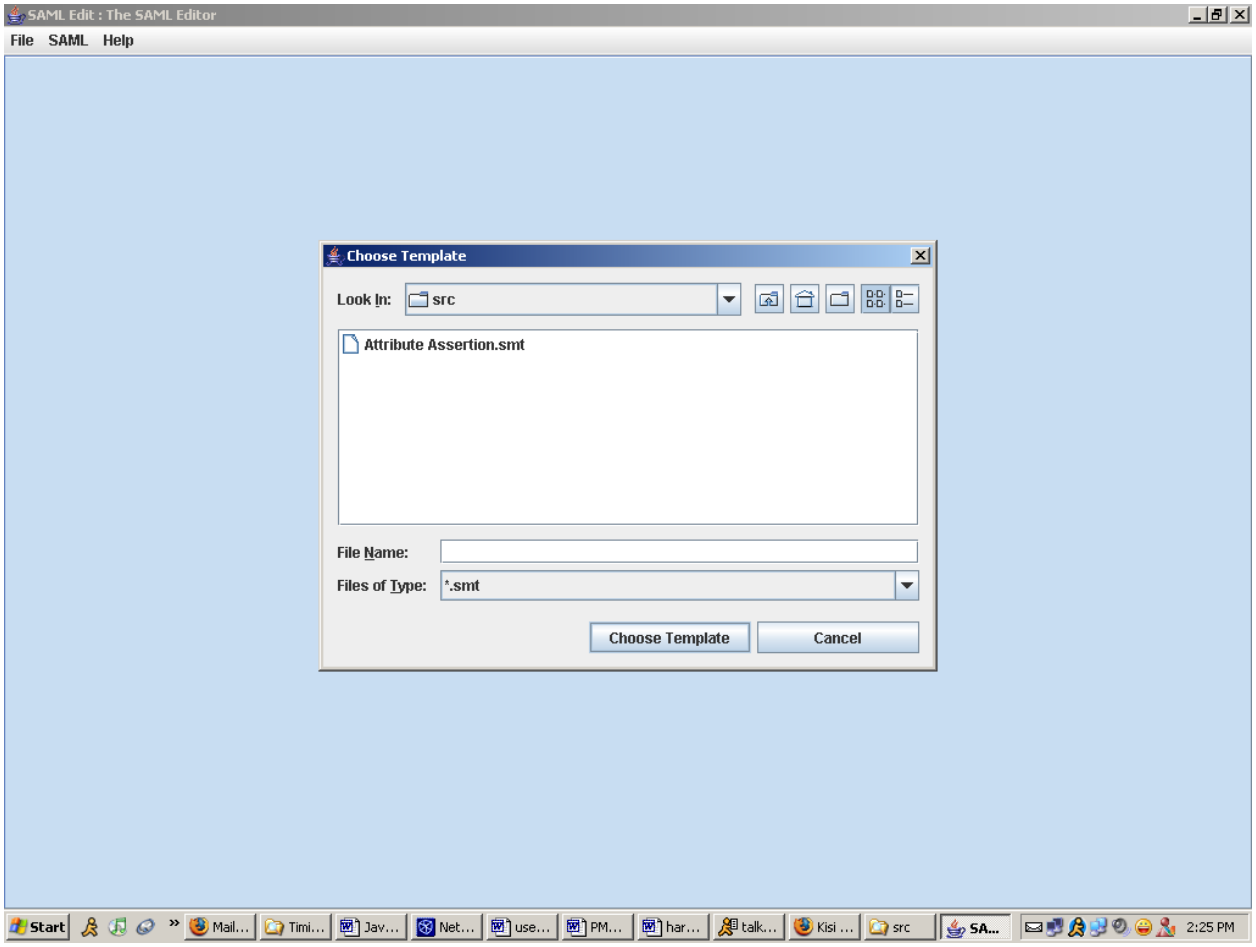



Figure 7: Creating a "New" File from a *Template*.

- **Open**

  The User can open an existing SAML Assertion file (.saml extension) in a new editor tab pane. The editor allows the user to open multiple SAML Assertion files in a single session.
  When a particular file is loaded/opened, the underlying Sun-XML parser converts it into a DOM tree. Any subsequent changes made to the file data is actually reflected in the underlying in-memory DOM tree.

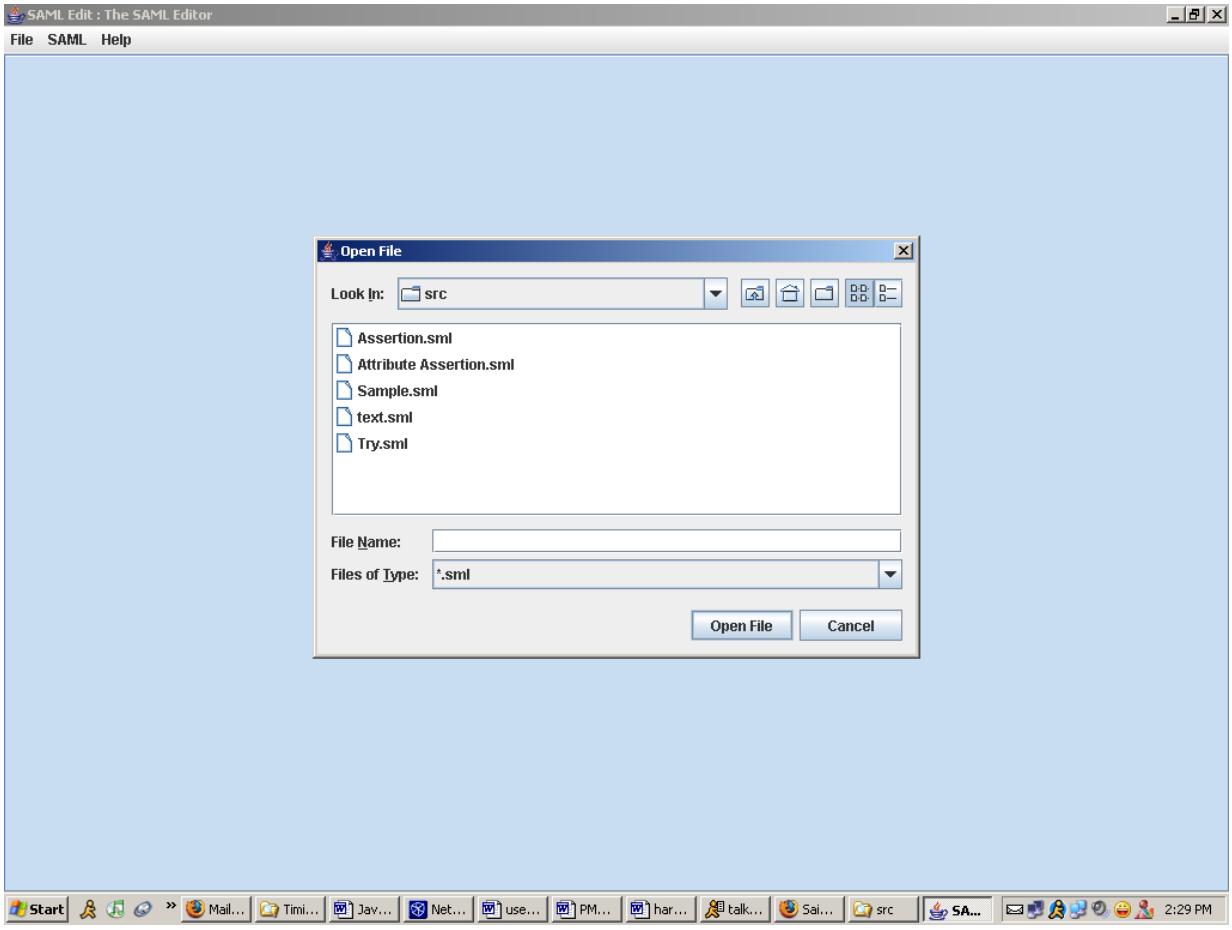



Figure 8: Opening an existing File

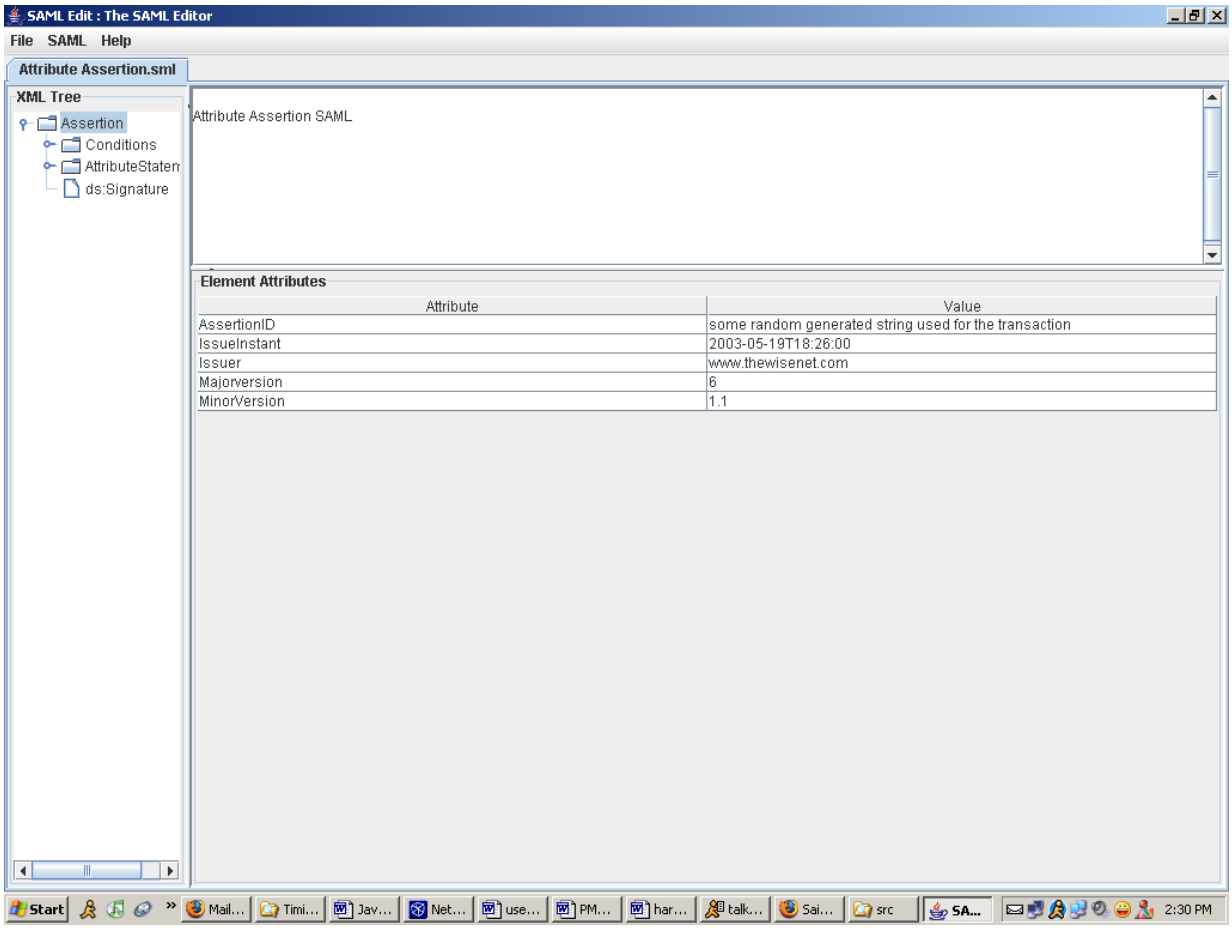



Figure 9: Attribute Assertion text File

- **Save**

  The User can save changes made to a particular SAML Assertion file (as reflected in the in-memory DOM tree representation of the same) to the actual text file by using this option. This writes the data to the persistent storage, like the hard-drive.

- **Save As**

  In case the User makes changes to an existing SAML Assertion file and desires to save the changes a new file, he/she can use the "**Save As...**" option from the **File** menu. Invoking this menu option throws up a "**Save As...**" dialog box that allows the User to provide a new name for the file

- **Close**

  The “File>Close” menu option is tab-specific. Invoking this closes the file opened in the current, selected tab. Since a particular session of the editor allows multiple files to be opened/manipulated simultaneously, the “File > Close” offers the User a neat way to close files he is done editing.

  When multiple files are open at the same time, there is a possibility that the User may make some inadvertent changes to a particular file. This may happen for various reasons:
  → the file names were similar
  → the target and the other file were on adjacent tabs
  → Or, he/she selected the wrong tab, and hence, the incorrect file for modification.

  By closing individual files (and tabs), the User can reduce the clutter and avoid such mistakes.

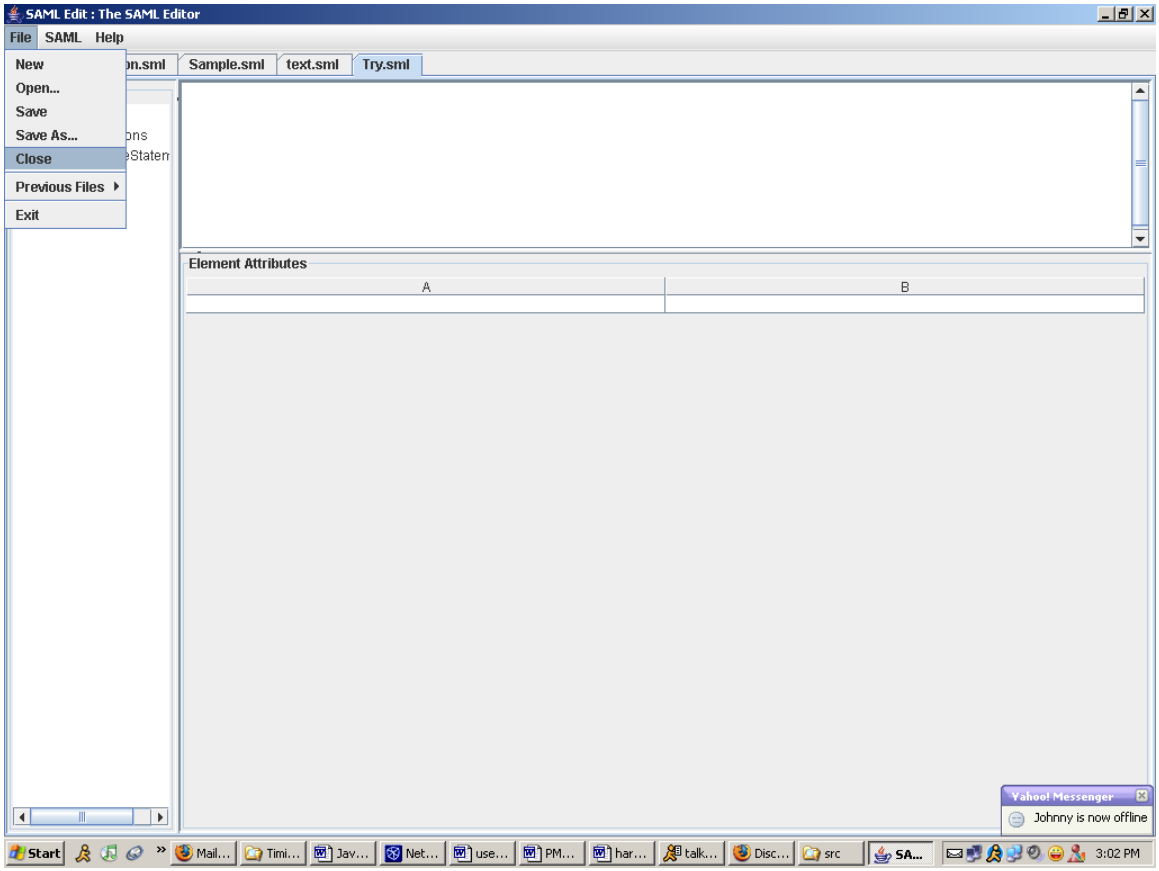



Figure 10: Invoking “Close” on “Try.sml”

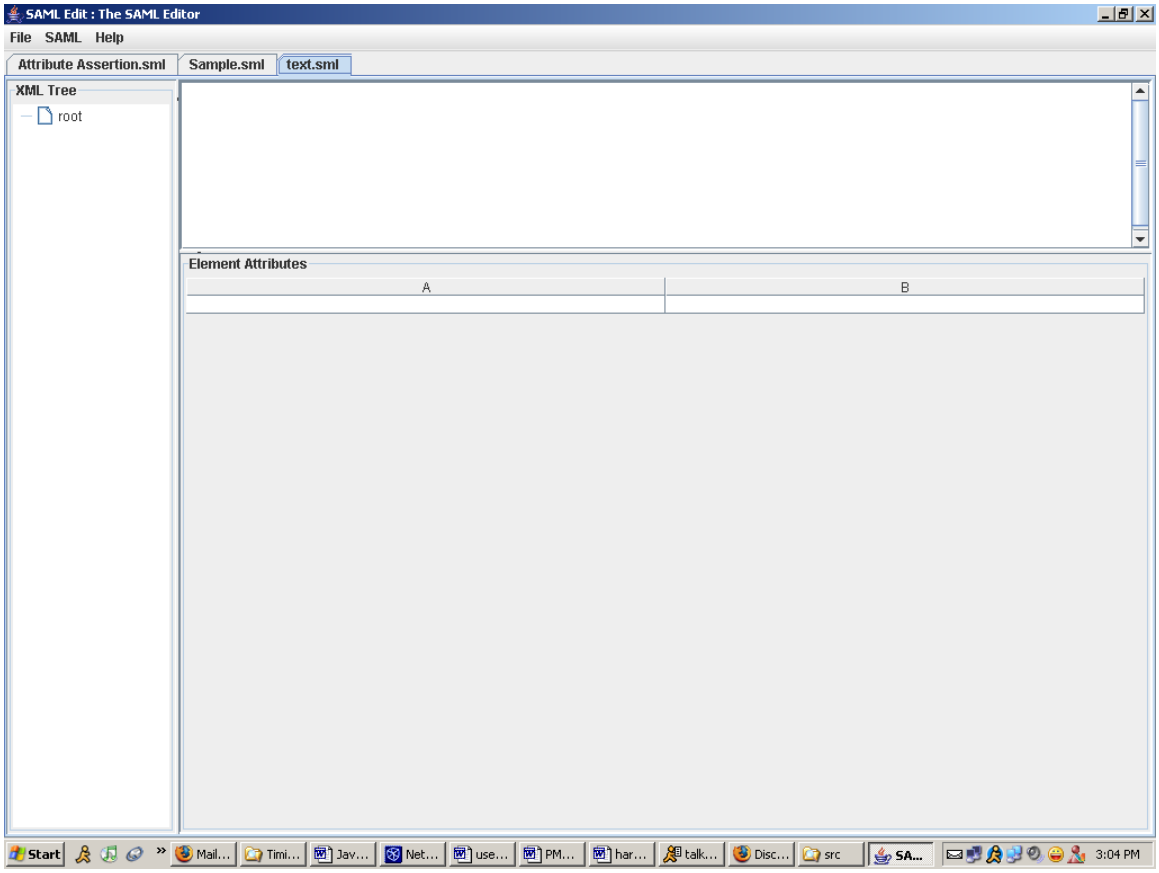



Figure 11: After "Closing" Try.sml

- **Previous Files**

  This is a convenience feature incorporated to save time. This option basically presents a list of the last X-number of files that have been opened. Such a history list allows the User to open frequently modified files with a single click, instead of having to open the file by browsing for it.

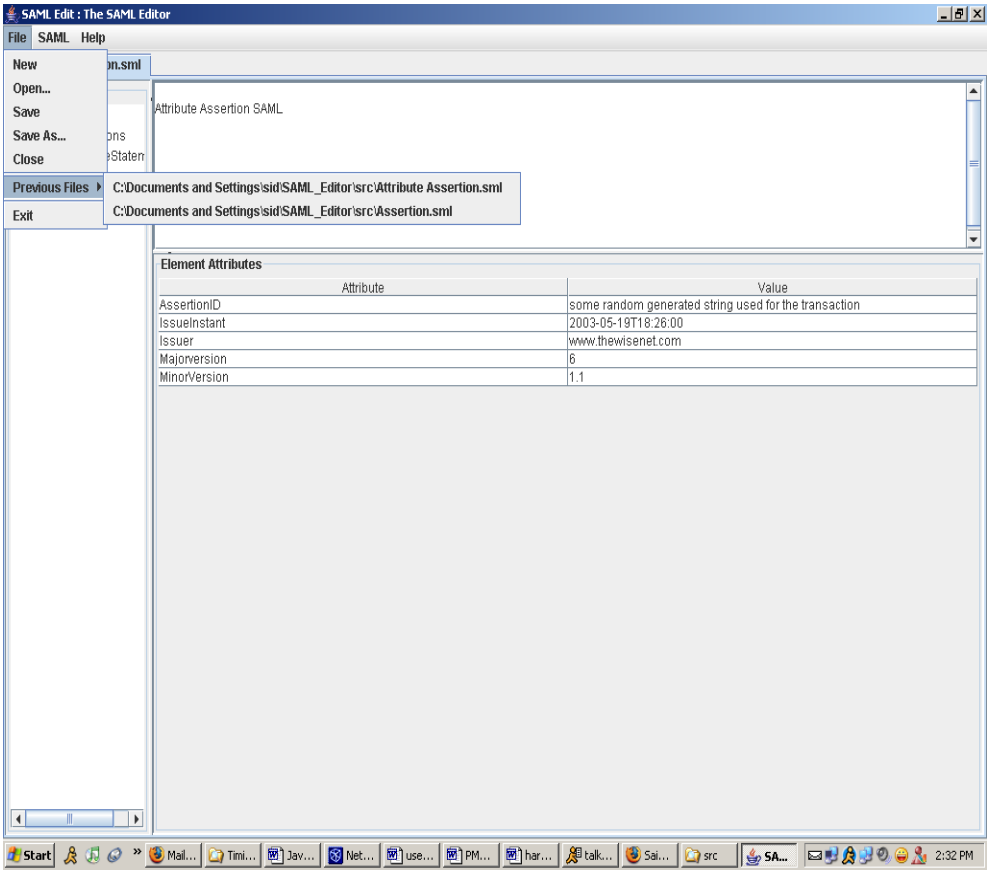



Figure 12: "Previous Files" Menu Option

- **Exit**

This is where the User can exit the entire application by closing ALL the open files (and tabs) in one shot. A *confirmation* dialog box ensures that the action is not inadvertent.

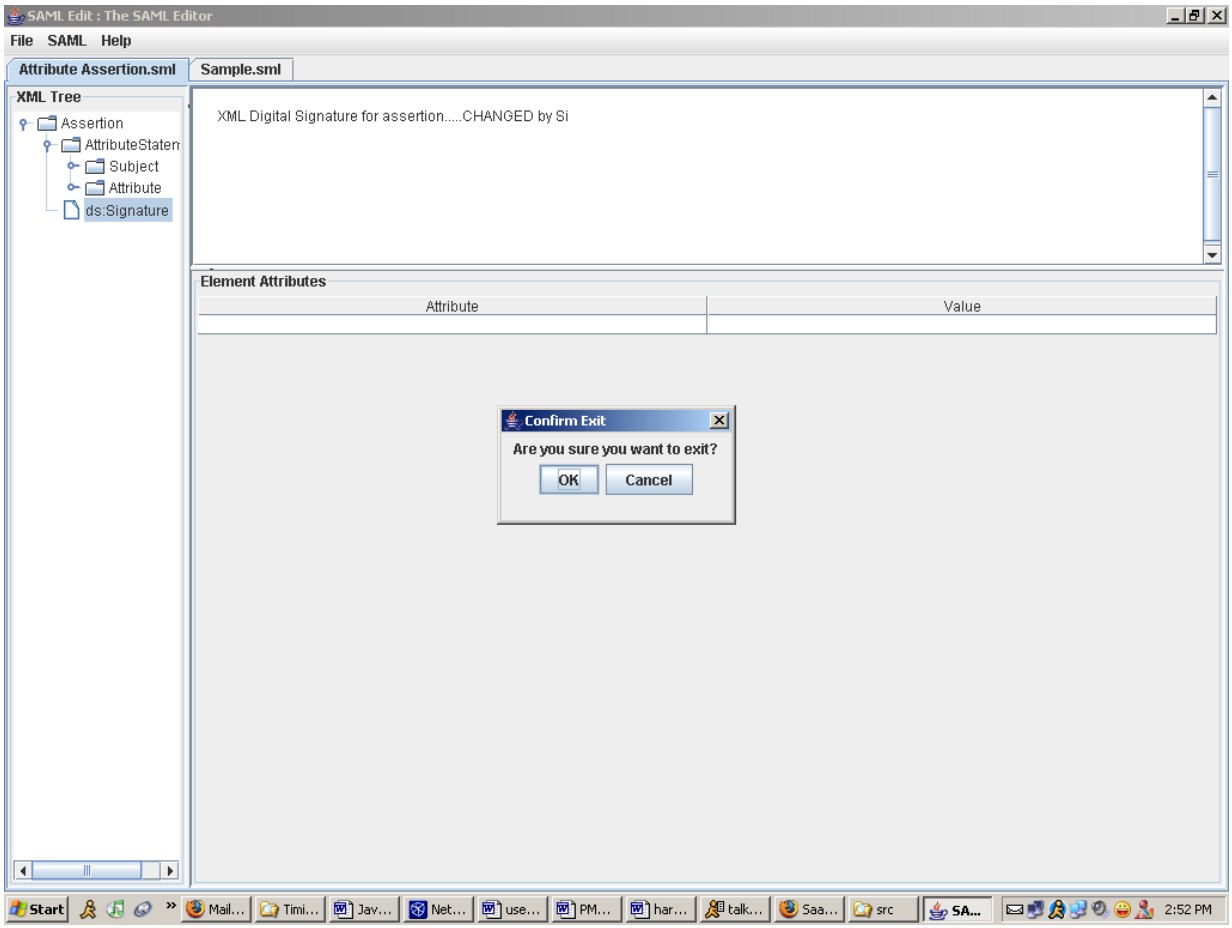



Figure 13: "Exit" Confirmation Dialog Box

- **SAML**

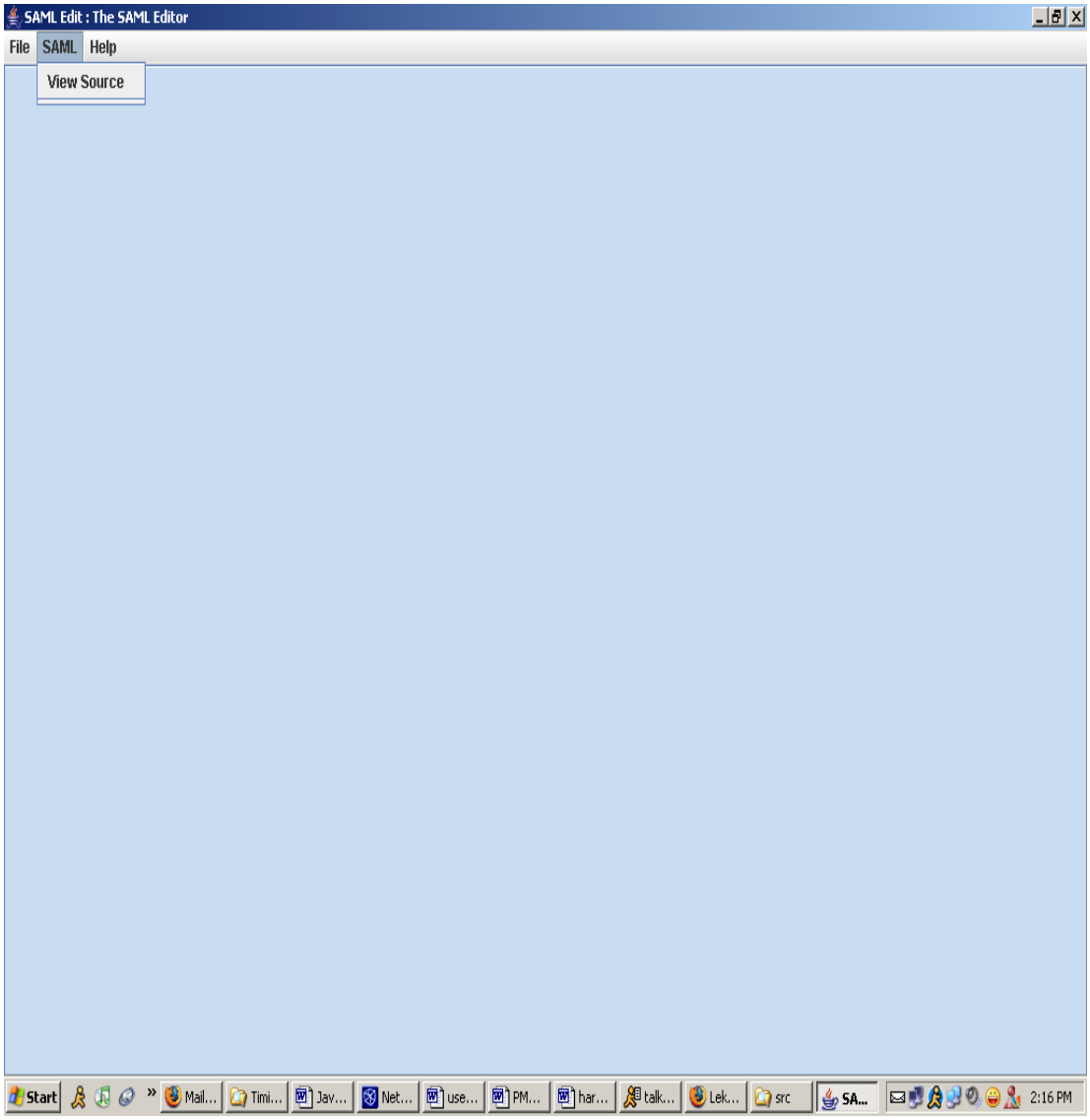



Figure 14: SAML Menu Options

- **View Source**

  This is for the SAML-literate User. Invoking this option opens the actual SAML Assertion text file – complete with all the tags and their attribute/value pairs – in a separate dialog box text area. The file is modifiable as-is. Any changes made here are reflected in the actual SAML Assertion text file on the persistent storage.
  In order to save the changes made here, the User needs to press the "**Save Changes**" button at he bottom-left. Pressing the "**Cancel**" button on the bottom-right dismisses the source dialog box. If the User presses

"**Cancel**" without pressing the "**Save Changes**" button prior to it, any changes made to the SAML Assertion text file is lost.

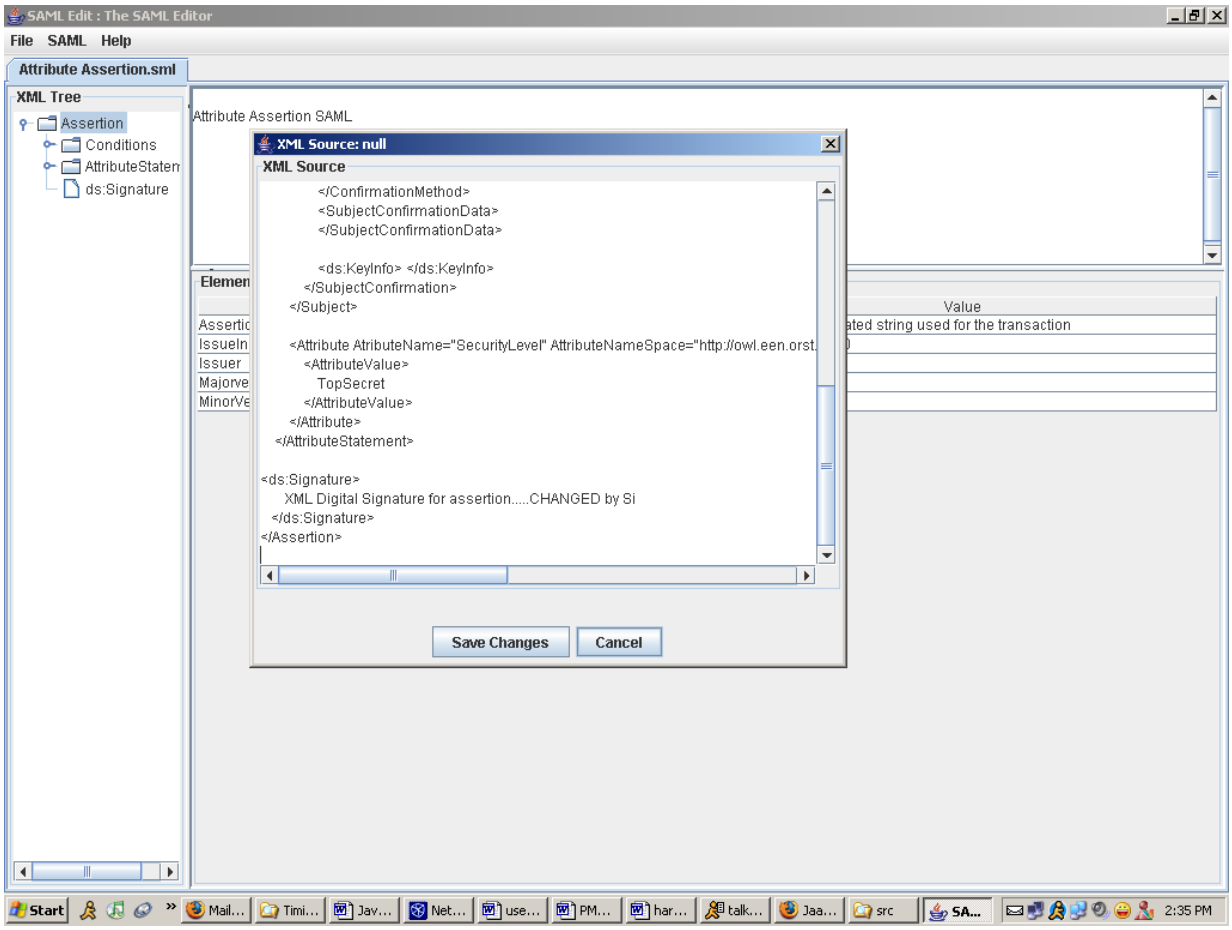



Figure 15: SAML Assertion text-file Source

- **Help**
  - **About**
    Invoking this options displays the "About" dialog box.

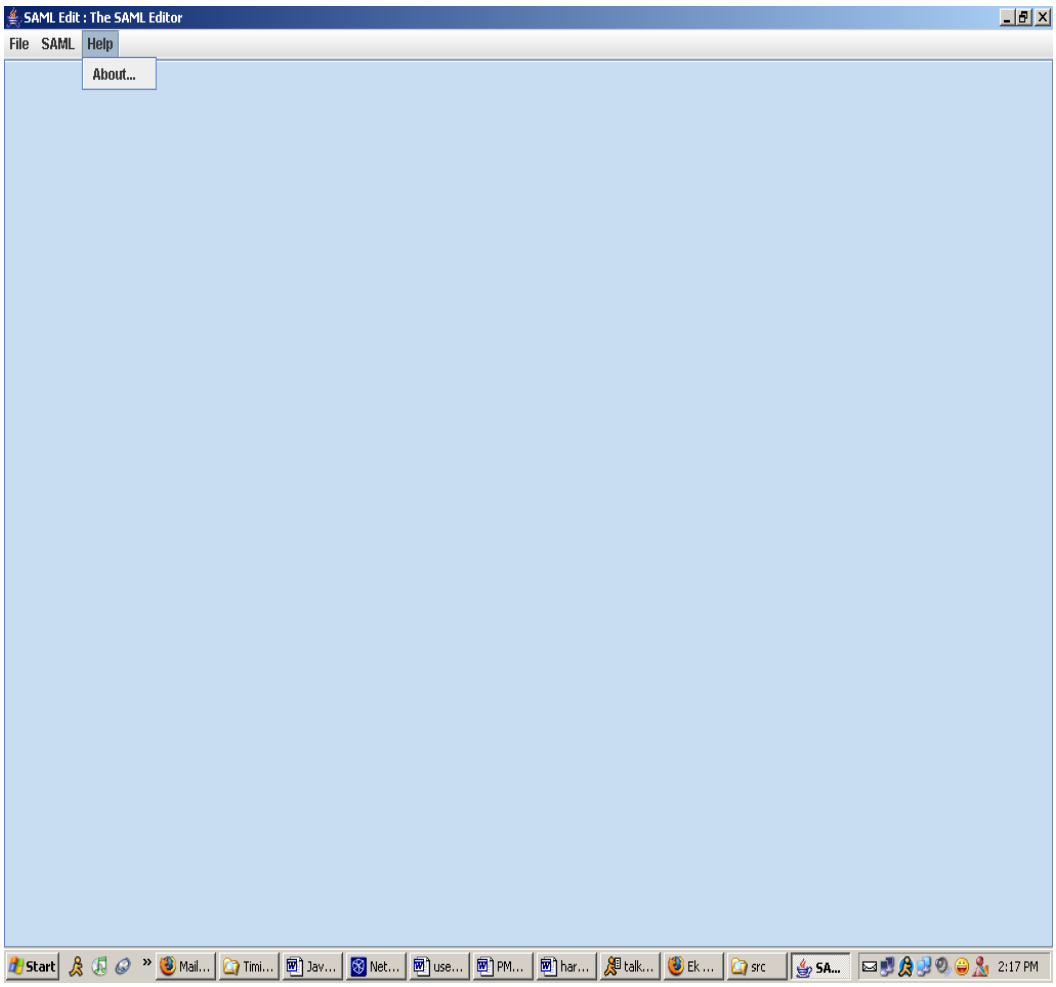



Figure 16: Help Menu Option

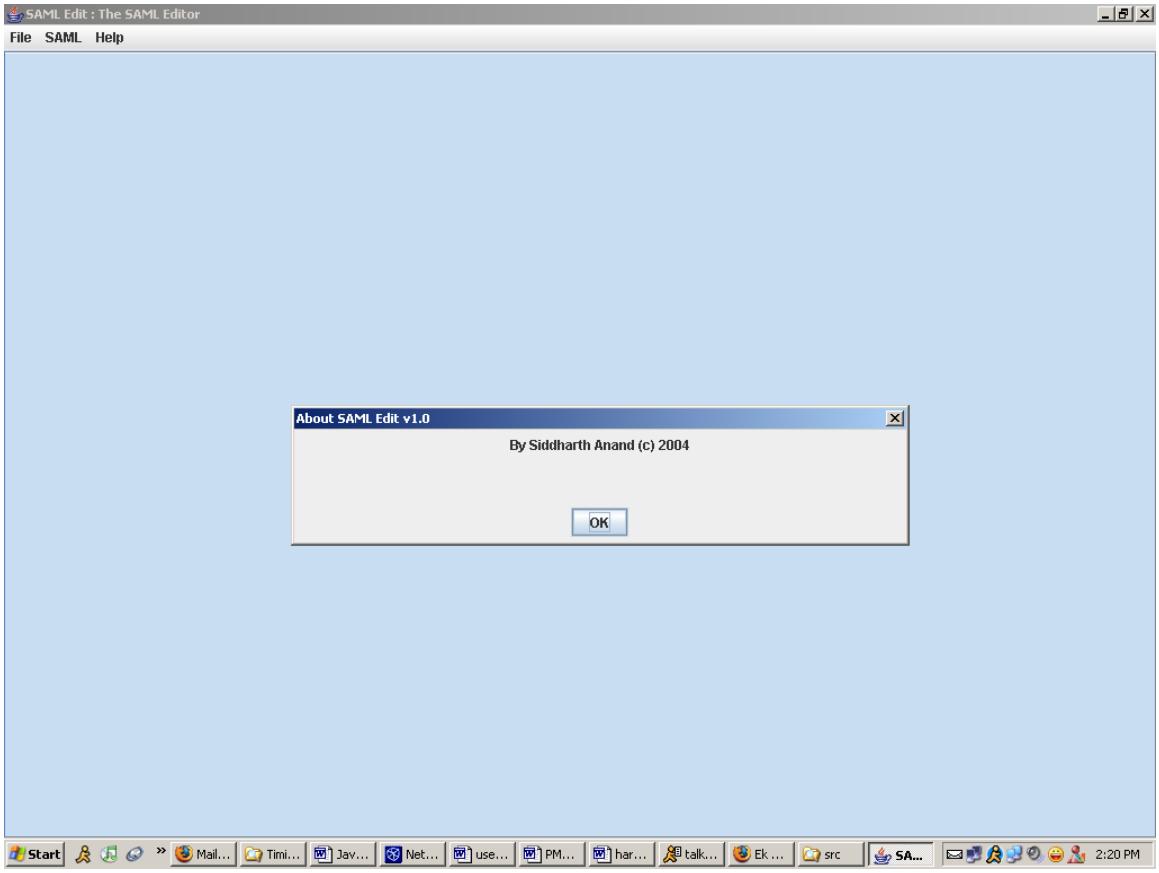



Figure 17: The "About" Dialog Box

### 7.4.3 The User Interface – Display Panes/Views

Every time the User opens a *new* file, a new tab pane is created. A particular tab pane contains a single file, with the file name displayed at the top of each tab. A tabbed-pane approach to opening multiple files offers a seamless way of moving from one file to other without cluttering the User Interface.

Within each tab, a particular SAML Assertion text file is displayed in a very intuitive way:

- A tree-based view of the SAML file is presented in the left pane. This view reflects the structure of the underlying SAML Assertion text file. The visual arrangement of the node (their indentation from the left) represents the nesting level of the underlying tag associated with a particular node.

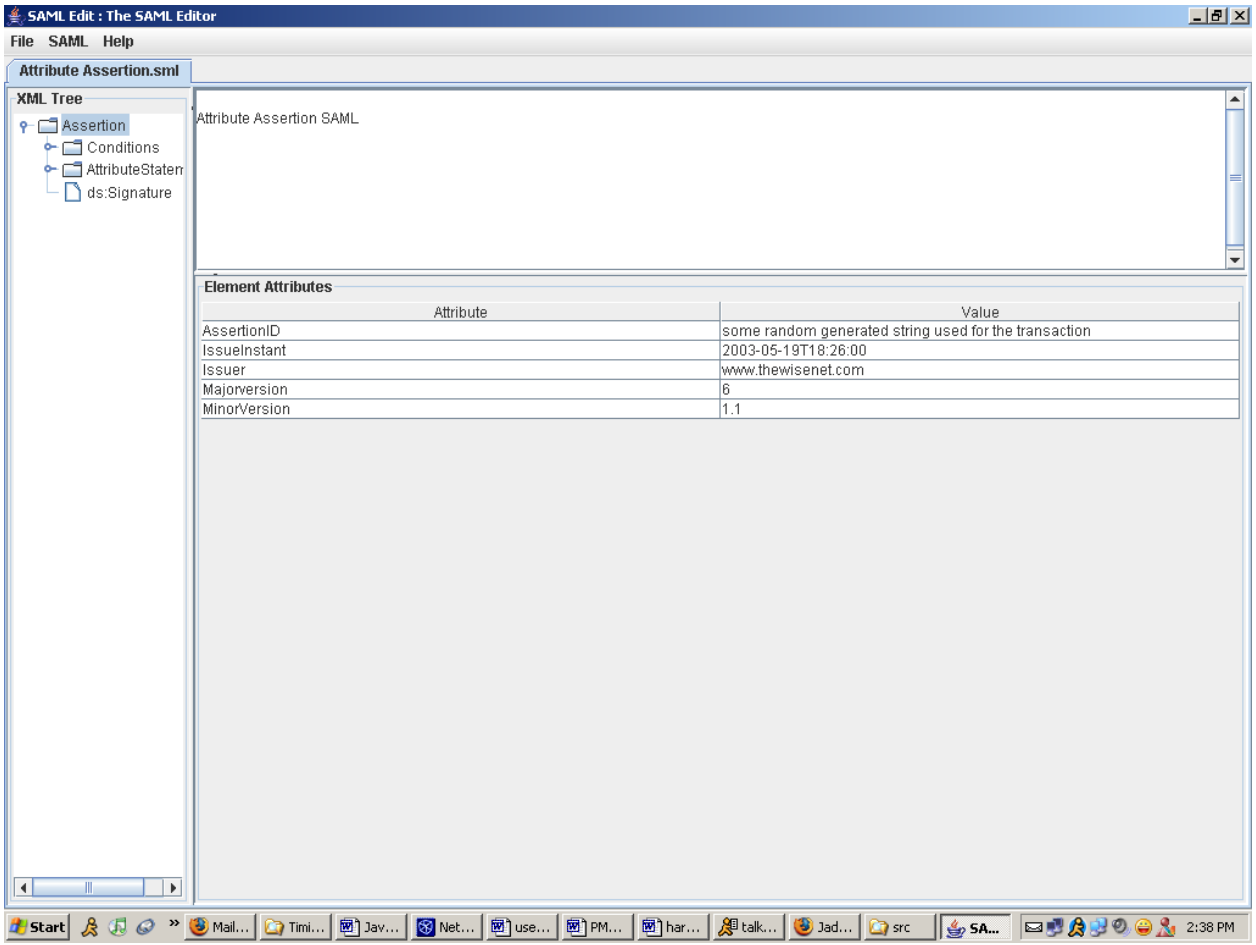



Figure 18: Tree-View of the SAML Assertion in the *Left-Pane*

An additional feature associated with the tree-view is the ability to add/delete tags/node to the tree, and hence to the underlying SAML Assertion text file, using a right-click pop-up menu.

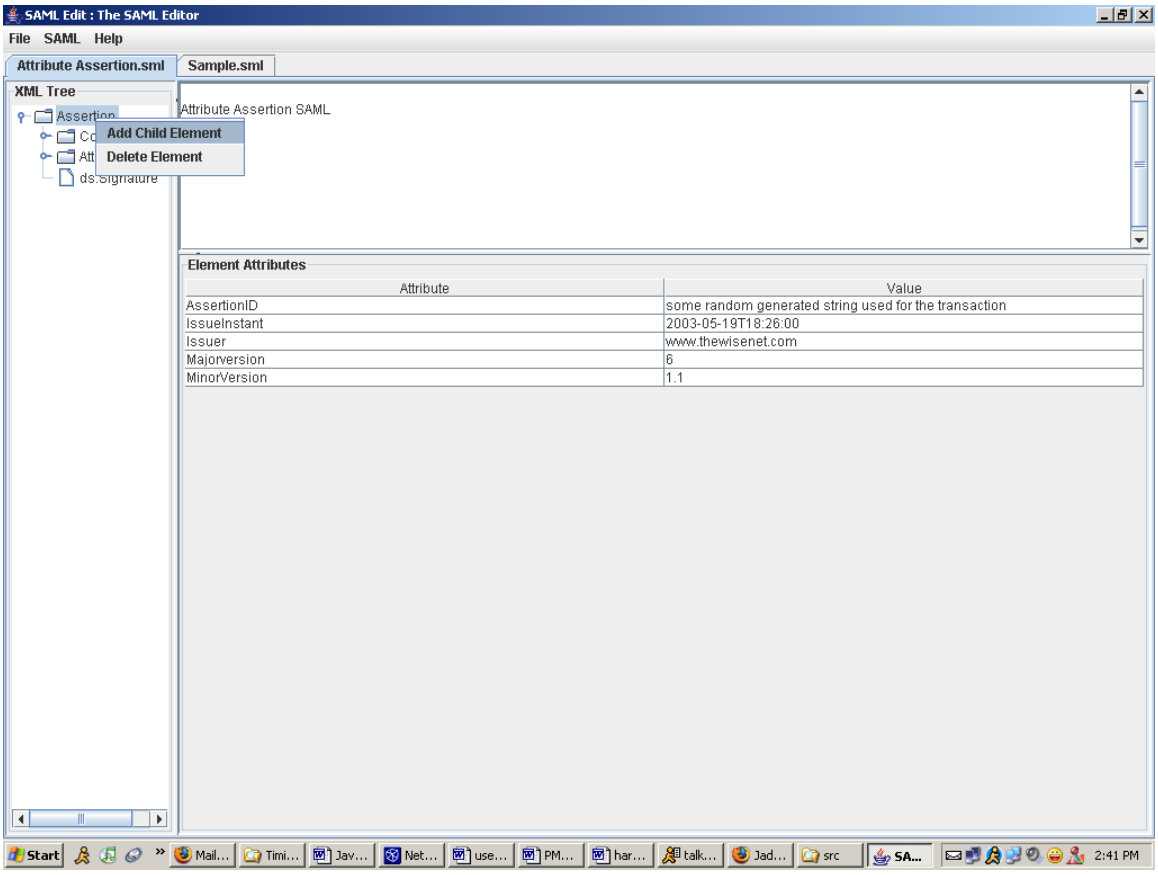



Figure 19: Pop-up Menu

- First the User needs to select a particular node in the tree

- Right-clicking the mouse on a node presents a pop-up menu that allows you to **Add/Delete** a node.

- The pop-up menu allows the User to "**Add a child element"**: The node/tag/element added is one nesting level below the *selected* node. A separate dialog box, with a text field, is presented where the User supplies the name of the *new* child tag/node/element. The editor automatically adds a *closing tag* to ensure that the SAML file is well-formed.

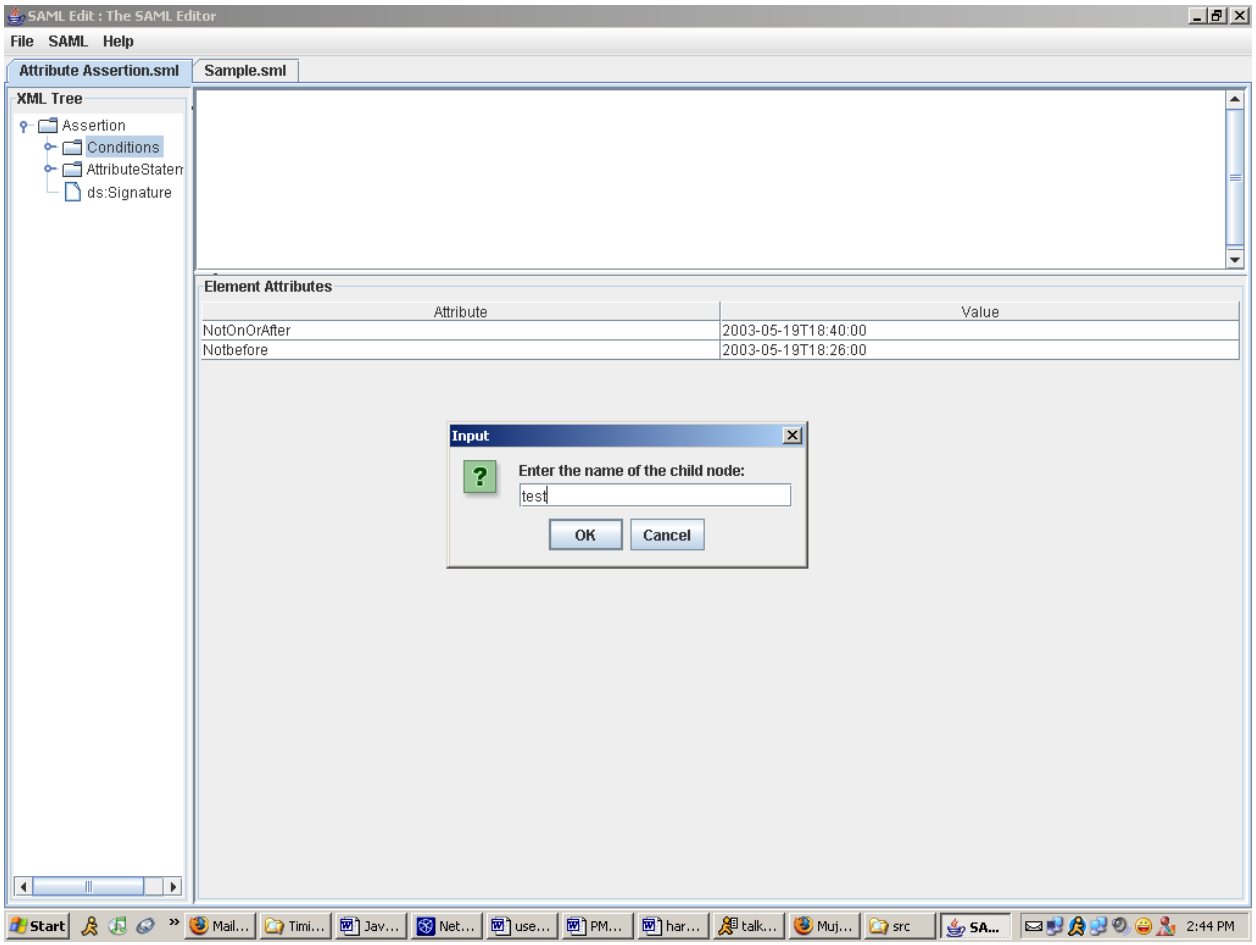



Figure 20: “Adding a Child Element” Dialog Box

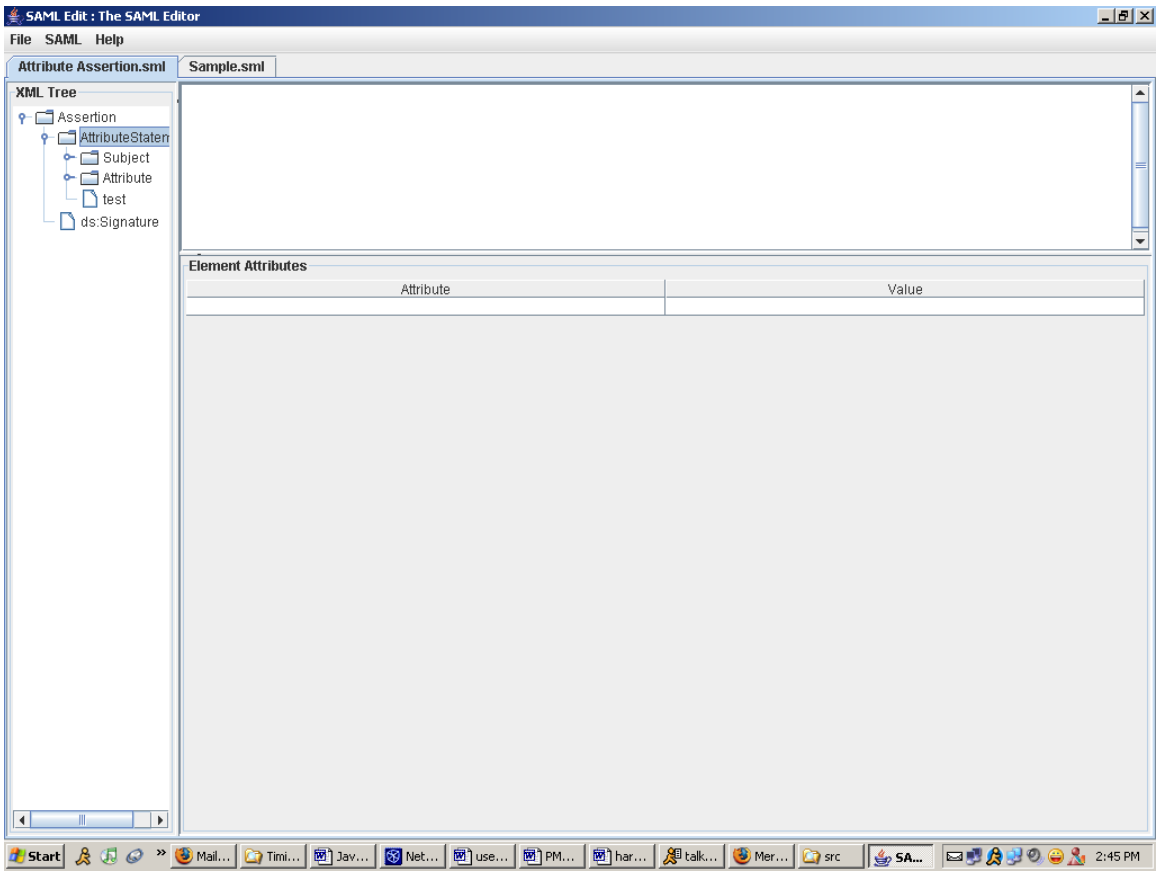



Figure 21: New Child Element "test" is added as a child to "AttributeStatement" element

- The pop-up menu also allows the User to "**Delete Element**": This options simply *deletes* the *selected* node/tag/element. A confirmation dialog box is displayed before the *selected* node/tag/element is removed.

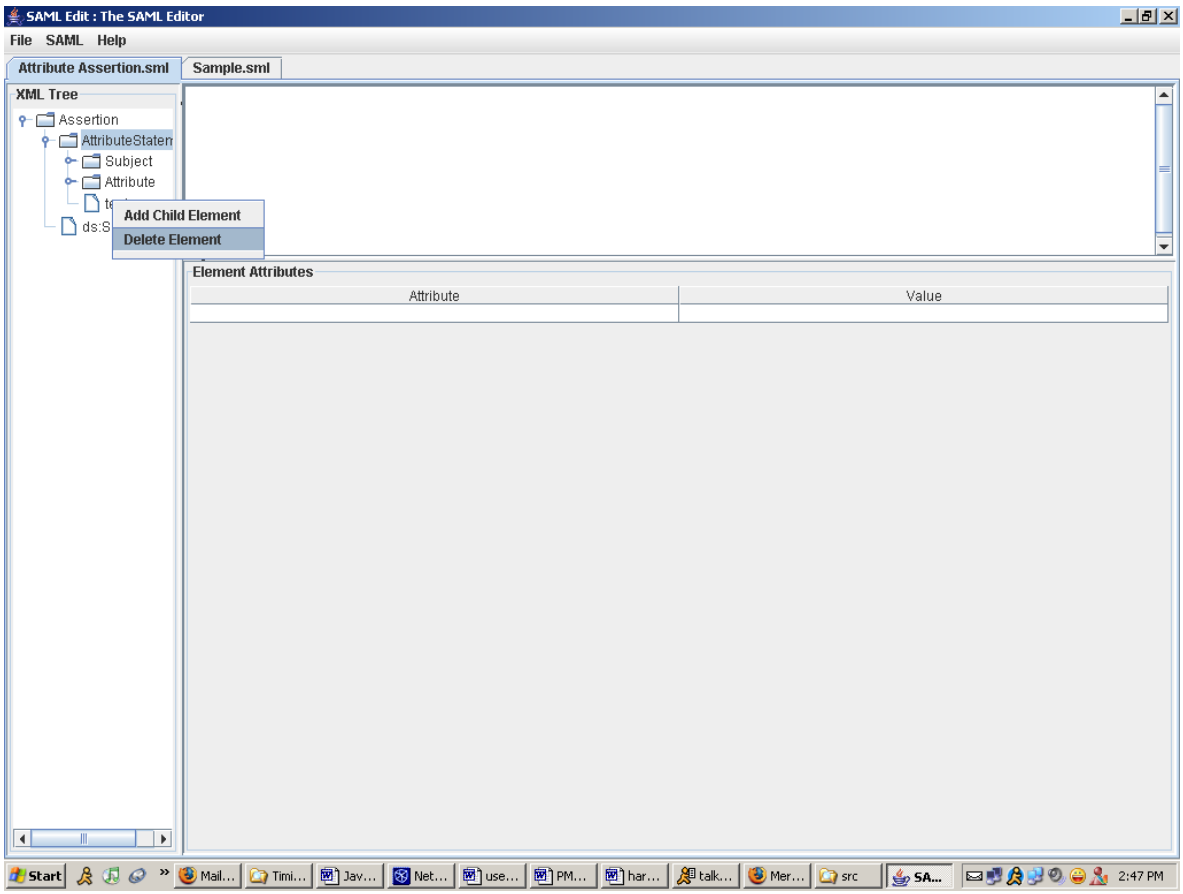



Figure 22: Deleting the “test” tag

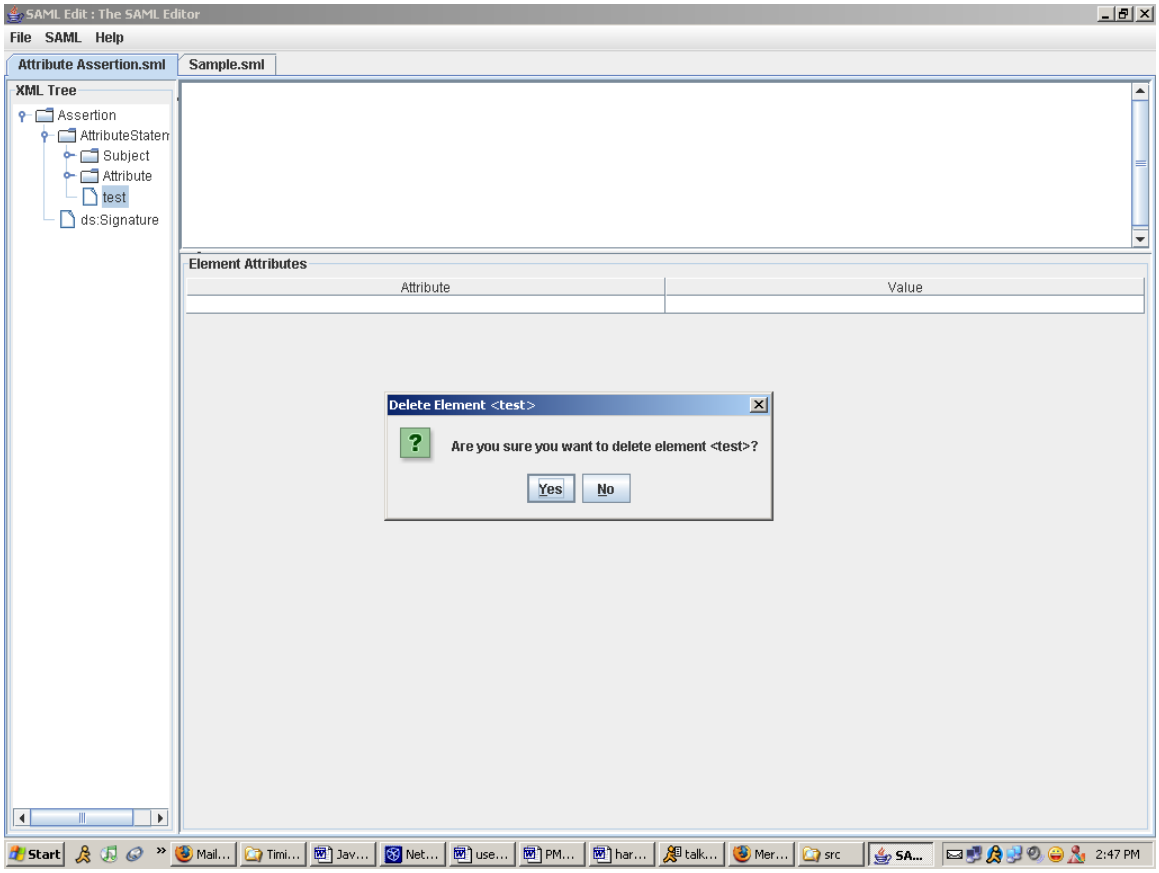



Figure 23: Confirmation Dialog Box to delete the *selected* element/tag/node

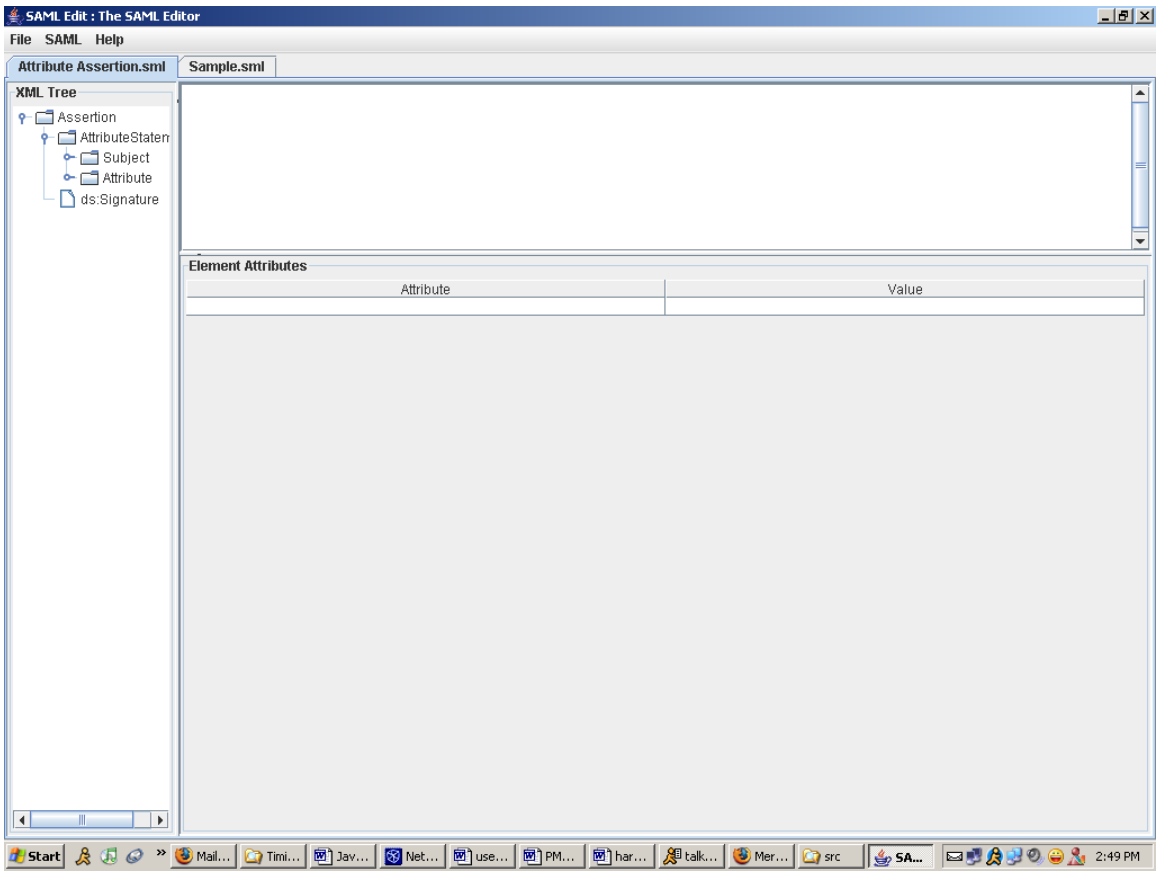



Figure 24: After the "test" tag has been deleted

- When the User selects a particular node in the left pane, its value is displayed in the top-right text pane. He/She can modify this value. Any changes made here are reflected in the underlying in-memory DOM tree that represents the SAML document. These changes are NOT saved to the actual SAML Assertion text file until the User specifically saves it using either the "**File > Save** "or the "**File > Save As…**"

- The User can move through the tree, selecting **one** node at a time and make changes along the way. He/She can make changes to multiple node values before he/she decides to save it to persistent storage. Recording user changes in the in-memory DOM tree makes this possible. When the user finally decides to save the file to persistent storage using either the "**File > Save** "or the "**File > Save As…**", the editor just walks the in-memory DOM tree representing the SAML document and writes the changes to the text file.

- When the User selects a node in the left pane, the tag value is displayed in the top-right text pane. As we know, like XML, SAML allows ***attribute/value pairs*** in a tag. All the ***attribute/value pairs*** associated with a particular tag (and hence node) is displayed in a two-column table in the bottom-right pane.

  The left column displays the **attribute name** while the right column the **attribute-value**. The **attribute names** are NOT modifiable; the **attribute-values** are.
  The User is NOT allowed to *add/delete* attribute/value pairs as the standard pre-defines these attributes. So, in case, the User does not want to supply a value for a particular attribute, he can simply leave the "**Value**" column blank.

  As with the tag/node values, the User can modify attribute/value pairs for multiple tags before he decides to save the changes to the persistent storage. As before, user changes are recorded in the in-memory DOM tree.

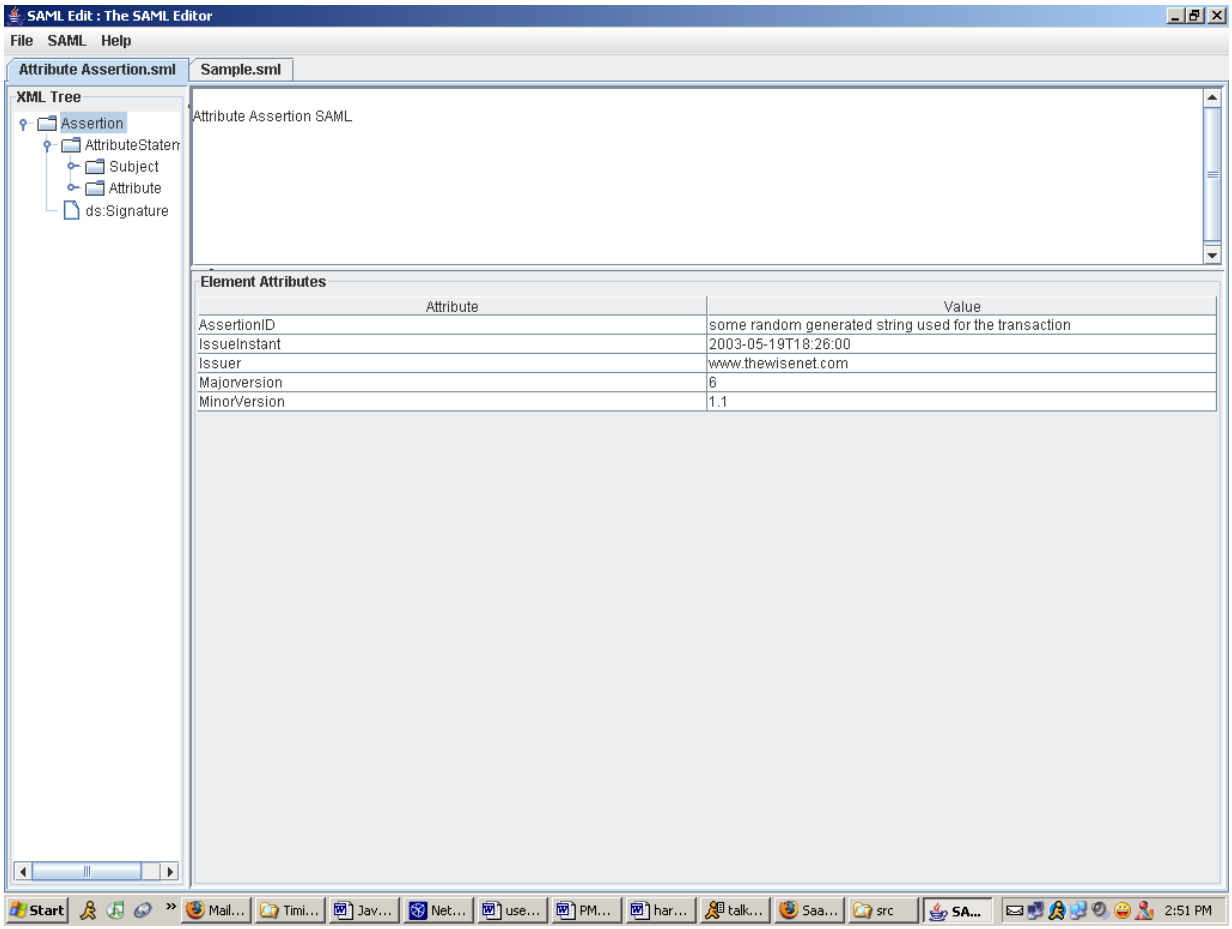



Figure 25: Assertion tag selected

Recording both the tag value and attribute/value pair changes in the in-memory tree not only allows the user to make multiple changes before he finally saves the file, it also helps display the most current information for both the tags and the attribute/value pairs as the user navigates the tree back and forth while making additional changes. He/She DEOS NOT have to save *every* change to persistent storage before he/she can change another one.

# 8. Conclusion and Future Work

SAML Edit® is a PC-based tool for creation and manipulation of SAML Assertion text files (as defined by *OASIS*). The tool allows its user to modify the data/information contained in the underlying file. As security information between entities is exchanged using such SAML Assertion text files, the ability to manipulate information contained in such a file is essential.

An obvious area of future work could be to create a ***micro*** version of SAML Edit® using J2ME. This would allow users with cell-phones, PDAs and such limited-resource devices to manipulate SAML Assertion on the fly.

Another suggested improvement could be to integrate a database back-end to store the SAML Assertions, instead of in a File Folder. The user of SAML Edit® can then be required to login before he can use the editor. The login process can be used to determine the SAML Assertions that the user is allowed access.

# 9. Glossary:

**Attribute Authority:** (Conf) A system entity that produces Attribute assertions, based upon TBD inputs.

**Authentication** – (from glossary with principal added) (Conf) Authentication is the process of confirming an entity's asserted principal identity with a specified, or understood, level of confidence.
The process of verifying a principal identity claimed by or for a system entity.

**Authentication Assertion:** Data vouching for the occurrence of an authentication of a principal at a particular time using a particular method of authentication. Synonym(s): name assertion.

**Authentication Authority:** (Conf) A system entity that verifies credentials and produces authentication assertions

**Authorization Attributes**: Attributes about a principal which may be useful in an authorization decision (group, role, title, contract code,...).

**Authorization Decision Assertions** : In concept an authorization assertion is a statement of policy about a resource, such as:
the user "noodles" is granted "execute" privileges on the resource "/usr/bin/guitar."

**Credential**: Data that is transferred or presented to establish a claimed principal identity.

**DOM:** Document Object Model, a W3C standard.

**OASIS:** Organization for the Advancement of Structured Information Standards

**Persistent Storage:** A long-term storage media like the hard-drive.

**Policy Decision Point** : (from glossary, access control decision) The place where a decision is arrived at as a result of evaluating the requester's identity, the requested operation, and the requested resource in light of applicable security policy.

**Policy Enforcement Point** : (access enforcement function) The place that is part of the access path between an initiator and a target on each access control request and enforces the decision made by the Access Decision Function.

**Principal, or Principle Identity:** An instantiation of a system entity within the security domain.

**Resource**: Data contained in an information system (e.g. in the form of files, info in memory, etc); or a service provided by a system; or a system capability, such as

processing power or communication bandwidth; or an item of system equipment (i.e., a system component--hardware, firmware, software, or documentation); or a facility that houses system operations and equipment.

**SAML:** Security Assertion Markup Language

**SAML Edit®:** The SAML Editor.

**SAML Assertion Text File:** The text file that contains security data/information on a persistent storage in *OASIS* defined SAML format, a subset of XML.

**Security Domain:** Domain to which a given agent/user belongs.

**Security Policies**: A set of rules and practices specifying the "who, what, when, why, where, and how" of access to system resources by entities (often, but not always, people).

**Sign-on:** The process of presenting credentials to an authentication authority for requesting access to a resource

**System Entity**: An active element of a system--e.g., an automated process, a subsystem, a person or group of persons--that incorporates a specific set of capabilities.

**User:** A human individual that makes use of resources for application purposes. This may also be non- human such as parties and processes

# References:

[1.] www.oasis.org
[2.] http://www.oasis-open.org/committees/security/docs
[3.] www.opensaml.org
[4.] Various Online Java Resources
[5.] www.w3c.org to understand the Document Object Model

**Document identifiers:**

1. draft-sstc-core-21
2. cs-sstc-core-01 (PDF, Word)
3. draft-sstc-core-discussion-01.doc
4. draft-sstc-dsig-02.doc
5. draft-sstc-protocol-discussion-01.doc
6. draft-sstc-saml-issues-12.doc
7. draft-sstc-saml-reqs-01
8. draft-sstc-protocol-discussion-01.doc
9. draft-sstc-bindings-model-07.doc

[6.] K.C. Toth, M. Subramanium, The Persona Concept:: A Consumer-Centered Identity Model, MobEA (Emerging Applications for Wireless and Mobile Access), Budapest, Hungary, May 2003.

[7.] K.C. Toth, M. Subramanium, Persona Concept for Privacy and Authentication, International Business & Economics Research Journal, June 2003.

[8.] K.C. Toth, M. Subramanium, I. Chen, Persona Concept for Privacy and Authentication, International Applied Business Research Conference, Acapulco, Mexico, March 2003