XML-Web Services Working Group Final Report

XML-Web Services Working Group Final Report

Background

The consensus of the Information Systems (H) Task Force at its May 10, 2006 meeting was that it was unlikely that progress would occur soon on eXtensible Business Reporting Language (XBRL). In the meantime, requests by regulators for access to data maintained by NAIC using XML-Web Services were occurring more and more frequently. In fact, Web Services had been developed for producer data, SERFF, Online Fraud Reporting and I-SITE financial reports. Other individual state requests for Web Services had been received; however, NAIC standards had not been established for XML taxonomy or delivery of Web Services. In addition, there was no plan in place to prioritize data for Web Service implementation. Therefore, the following was recommended and then adopted at the June 10, 2006 Information Systems (H) Task Force meeting:

- Recommend to the Information Resources Management (H) Committee that the XBRL Working Group be closed down
- Appoint a new XML-Web Services Working Group
- Develop charges for the new working group

The XML-Web Services Working Group (XWSWG) was formed in August 2006 and additional charges were adopted for the Technical Consulting Working Group (TCWG) on the same subject. The XML-Web Services charges were divided into 1) those that are entirely technical in nature, which have been recommended for the TCWG, and 2) those that would recommend a policy direction, which have been proposed for the XWSWG. The charges were defined as follows:

XWSWG:

- Gather input from regulator information technology and business staff members regarding data maintained by the NAIC that would be beneficial to states if offered through Web Services.
- Assess states' ability both from a technical and a resource perspective, to take advantage of Web Services provided by the NAIC. Consider recommendation of a developer track at the 2007 Tech-Ex conference to train state developers interested in leveraging NAIC reference implementations and how to implement NAIC Web Services.
- Recommend priorities for the implementation of Web Services, including relative order of data sets to be implemented. Recommend whether a XML-Web Services project should be proposed to IRMC for sponsorship in the next available budget cycle as a stand-alone project, whether any such project should include software to aid the rapid deployment of Web Services and whether Web Services middleware should be purchased.

TCWG:

- Provide counsel to NAIC staff in determining appropriate taxonomy, schema/naming conventions and reference implementations as standards for data formatted as XML to be made available through Web Services. Review and approve such standards. Standards will be considered as updates to the National Technical Architecture. The working group's goal will be to present any recommended XML-Web Services standards to the Information Systems Task Force at its December 2006 meeting.

- Provide counsel to NAIC staff in developing a plan for security classification of data offered through Web Services and a plan for access control based on such classification. Such plan should be presented to the Information Systems Task Force at its March 2007 meeting.

This report outlines activities of the Working Group and others that have addressed these XML-Web Services charges.

The Working Group began by establishing a basic understanding of Web Services in Service Oriented Architecture (SOA) and the use of XML for transmitting data. Some of the base components include: Extensible Markup Language (XML), XML Schema, Web Service, Web Services Description Language (WSDL), Universal Description Discovery and Integration (UDDI), Simple Object Access Protocol (SOAP), and Reference Implementation (Attachment A). Definitions of these components and how they work together, along with the characteristics of NAIC Web Services and the services available at that time were presented to the Working Group members at their first meeting August 16, 2006 (Attachment B).

The Working Group also reviewed the research and development efforts of the financial XML data and Web Services (Attachment C). These efforts included the creation of an XML schema for the Pick-a-Page data that has been available as a prototype since 2002. However, this schema does not meet the XML standards ultimately recommended to the TCWG. In addition, the I-SITE Report Web Services were released to production. These Web Services provide access to select financial reports in formats available in I-SITE. While the primary focus of this project was to provide the reports in HTML format, a natural by-product was the availability of the Pick-a-Page XML data via the Web Service. This has not been made widely known.

State Technology Survey

In order to assess the states' ability, both from a technical and resource perspective, to take advantage of NAIC Web Services, the Working Group defined a series of questions for the 2006 State Technology Survey (Attachment D). The results of the survey revealed: 33% of the respondents indicated that their department had no plans or experience with consuming Web Services, while 67% were planning, developing or had an application-consuming Web Services in production; 57% of the respondents indicated that their department had no plans or experience with hosting Web Services, while 43% were planning, developing or had an application-hosting Web Services in production; 87% of the respondents indicated that department personnel would be interested in participating in training on implementing NAIC Web Services; 71% of the respondents indicated that NAIC staff should prioritize Web Service implementation as a balanced approach for state retrieval and submission of data; 12 departments, in addition to the XML-Web Services Working Group member states, showed interest in Working Group participation. An analysis of the priority and types of Web Services the respondents' states would use reflected three primary areas: financial, market and producer licensing. The Working Group felt that while the survey results better established appropriate priorities for Web Services development, more information was needed.

Follow-up interviews were conducted with the survey respondents who had indicated that financial Web Services would be a first priority for their state to use. Objectives of these interviews were to: (1) gather specifics regarding the type and format of data requested; (2) determine department's urgency and readiness to utilize the Web Services; (3) further educate other regulators on how Web Services work and the opportunities they provide; (4) assess Web Services training needs; and (5) identify states

interested in participating with the Working Group. Both technical and business representatives from the states were included in the interviews. A summary of the interview results (Attachment E) was reviewed. The Working Group recommended forming a Financial Web Services User Group with the following responsibilities: identify and prioritize financial Web Services; provide technical consulting and review of financial Web Services; if appropriate, draft a Project Request for financial Web Services development; determine appropriate state training; and consider security issues.

The Working Group heard status updates on the Market Regulation Web Services project, which began in September 2006 and State Producer Licensing Reengineering project, which began in January 2007. These projects appeared to be moving in directions that will address the market and producer licensing Web Services needs reflected in the State Technology Survey results. The Working Group offered both projects assistance in identifying and prioritizing Web Services for the states. The project managers for both of these projects indicated that was not necessary.

NAIC Development Guidelines

The NAIC Web Services Implementation Guide (Attachment F) was created to provide development areas with a decision tree on whether to use Web Services or not. It includes a series of questions, such as "Is this an existing application?", "Have users requested Web Services?", and "Is there time to update the Web application to use the Web Services?" Based on the response of each question, the developer is directed to the appropriate course of action. The implementation guide also provides some implementation guidelines, which include phased in approaches.

TCWG and the NTA

The TCWG focused on its charge to review and approve appropriate taxonomy schema/naming conventions and reference implementations as standards for XML data to be made available through Web Services. The TCWG determined appropriate modifications to the National Technical Architecture (NTA) standards for Web Services (Attachment G). The NTA is a set of open standards that the NAIC will use in its architecture, and will also be used by all entities that are required or choose to interface with NAIC systems. At the heart of NTA is the concept that the architecture will not impose any requirements on what tools or methods the external entities use to interface with the NAIC assuming they follow the group of open standards outlined in the NTA. The TCWG determined that it would be appropriate to develop an implementation guideline document referenced by the NTA. This document defines the NAIC's implementation of the NTA open standards (Attachment H). In addition, the TCWG reviewed a recommendation paper (Attachment I) that outlined the pros and cons associated with the current method used to tag elements in I-SITE's Pick-A-Page application.

E-Regulation and TechEx Conference

At the recommendation of the Working Group several sessions related to XML and Web Services were presented at the 2007 E-Regulation and TechEx Conference. These sessions included: Technology in Insurance Regulation, a keynote presentation of the changes that have occurred in insurance regulation with the application of technical solutions to the demands of state-based regulation of insurance; XML & Web Services Revealed, presented in the Company Licensing and TechEx tracks provided an

overview of SOA and its components; and Web Services & Reference Implementations Training provided an overview presentation on Web Services followed by a hands-on exercise on developing a reference implementation.

Web Services Advancements

Several projects that have advanced Web Services have been undertaken over the last year. These projects include the following:

NAIC Web Services Registry

The initial phase of the Market Regulation Web Services project included the creation of the NAIC Web Services Registry. The Registry, which is currently available at https://external-apps.naic.org/NaicRegistry/ , provides a list of available Web Services and access to associated WSDLs, Schemas, Reference Implementations and documentation. In the future, the Registry will be replaced with a UDDI interface which will allow for the automation of the discovery process for our web services, and will also offer additional functionality in the web interface as well.

Web Services PICS Event

In conjunction with the Registry, a new Personalized Information Capture System (PICS) event was created. Subscribers to the new Web Services event are notified as new services are released and existing services are modified.

NAIC Report Web Services

A set of Web Services that provide access to the I-SITE financial reports was released in May 2006. The NAIC Report Search Web Service (NRSWS) provides search operations for finding available reports. The NAIC Report Web Service (NRWS) provides access to I-SITE financial reports in the specified format. The annual and quarterly financial statement data is available in XML format.

SERFF V.5

State insurance departments not only have Web applications to access, but because of the SOA nature of SERFF they can integrate their back-office systems using NTA standards. There are Web Services established to allow states to both pull and push data to and from SERFF. Similarly, industry may have access to Web Services, via data hosters, that will allow them to pull or push data to and from SERFF. Another key aspect of SERFF is providing SPI (SERFF Programming Interface) to business partners so they can write interfaces to provide companies with more robust systems while still using SERFF to submit filings to the states. This kind of flexibility is the cornerstone of a SOA.

Market Regulation Web Services

The objective of this project is to implement the Market Regulation & Consumer Affairs (D) Committee's directive to develop and provide all member states with Web Services, that would allow state insurance departments to retool their legacy computer systems to electronically upload adjudicated regulatory actions, complaints, special activities, and exam tracking data in real-time to NAIC databases. This technology will also provide a method to submit data real time without performing manual data entry within NAIC's I-SITE system. These Web Services will promote the collection of complete, accurate, and timely market information from the state insurance departments.

State Producer Licensing Re-engineering
The objective of State Producer Licensing Re-engineering (SPLR) project is to completely reengineer the producer licensing software code and data model. This system is heavily relied upon by the states to assist them in carrying out their producer licensing and market regulatory functions as well as by the industry in the automation of the licensing process. The Departments of Insurance and industry will not only have Web applications to access, but because of the SOA nature of SPLR, they will be able to integrate their back-office systems using NTA standards. Along with States and Industry, SPLR will also integrate with other NAIC applications such as I-SITE and can also integrate with Approved Business Partners.

SOA/Web Services Tools

Security foundation
The Security Foundation provides a mechanism to secure all information on the NAIC systems through: Authentication, Authorization, Auditing and Administration. This foundation is being implemented with Oracle Core ID and Lightweight Directory Access Protocol (LDAP). LDAP is an industry standard for storing user information as well as the capability to assign "roles" to those users.

National Portal
The framework for the Portal includes services like a search engine, data analysis and reporting and access to outside services as well. The Portal framework provides a secure interface that is user-friendly, personalized, extensible, relevant, and branded. The Portal project, being implemented with Oracle Application Server, will provide a method for communities of all kinds--regulators, industry, and public consumers—to gain access to integrated information from many different business areas. Web Services provide the perfect way to access this information.

Enterprise Service Bus
Enterprise Service Bus (ESB) assists in the SOA aspects of the design. It acts as a dispatcher, guiding the steps through the process. It also provides alert and auditing functionality. The ESB tool selected and being implemented with the SPLR project is iWay.

Extract, Transform and Load (ETL)
ETL is a data warehousing process that involves: Extracting data from outside sources, Transforming it to fit business needs, and ultimately Loading it into the data warehouse. This tool will be used to populate data marts.

Reporting Framework
The reporting framework being used in the SPLR project is WebFOCUS. This tool provides a lot of built-in functionality that developers currently have to build themselves. With this tool developers can more easily create and deploy reporting applications with advanced Web-based features such as integrated proactive hyperlink drill-downs.

UDDI
The Oracle Application Server UDDI is scheduled for delivery in an early iteration of the SPLR project.

Ongoing Tasks

As described above, a lot of work has been done to advance Service Oriented Architecture and Web Services. However, there is still work to be done. The following tasks are ongoing:

It is important that the Market Regulation Web Services and State Producer Licensing Reengineering projects receive the support needed to successfully complete.

Security classification, policies, and procedures must be developed and/or adapted for Web Services, along with an overall formulation of best practices for the states and NAIC.

The Financial Web Services User Group should continue their work with NAIC staff defining and prioritizing appropriate financial Web Services.

Conclusions

After a year of study, it is proposed that this be the final report of the XML/Web Services Working Group and that the working group be discharged. It is believed the above report is comprehensive and that it demonstrates that all charges have either been resolved or that work will continue under the direction of other groups or projects as described below.

XML-Web Services Working Group Charges:

- Gather input from regulator information technology and business staff members regarding data maintained by the NAIC that would be beneficial to states if offered through Web Services.

  This charge was accomplished through questions asked by the 2006 State Technology Survey, and was supplemented through interviews. Financial, market regulation and producer licensing Web Services were found to be most important to state respondents. Two major projects have emerged over the last year concentrating on market regulation and producer licensing. Existing XML schema definitions are in place for their deployment. Web Services will play prominently in their development. Financial Web Services have been the subject of ongoing work by NAIC staff and were made available in a form based on I-SITE reports in May, 2006. Two states, Florida and Kansas, are making use of them. Financial XML schema definitions follow existing financial data table formats. A Financial Web Services User Group will continue to consult with NAIC staff on development. If and when it appears appropriate or necessary to proceed with a Financial Web Services project request, the request will be proposed to the appropriate business committee or the IRMC through members of this group and staff.

- Assess states' ability both from a technical and a resource perspective, to take advantage of Web Services provided by the NAIC. Consider recommendation of a developer track at the 2007 Tech-Ex conference to train state developers interested in leveraging NAIC reference implementations and how to implement NAIC Web Services.

  The 2006 State Technology Survey was used as a tool to gauge state readiness for Web Services. Two-thirds of the state respondents indicated they were consumers of Web Services and more than 40% of states were making plans to host them. States were interested in Web Services training and Web Services topics and training played prominently in 2007 E-Regulation and Tech-Ex Conference

sessions. The development of a Web Services Registry and availability of a Web Services PICS event should help keep states advised of available Web Services. It is recommended that NAIC staff continue to provide states with Web Services training material and opportunities for Web-based training. On-site training opportunities should continue to be considered in coordination with the annual E-Regulation and Tech-Ex conferences.

- Recommend priorities for the implementation of Web Services, including relative order of data sets to be implemented. Recommend whether a XML-Web Services project should be proposed to IRMC for sponsorship in the next available budget cycle as a stand-alone project, whether any such project should include software to aid the rapid deployment of Web Services, and whether Web Services middleware should be purchased.

The priorities became obvious as a result of survey results and evolving member initiatives. Financial, market regulation and producer licensing priorities emerged from the 2006 State Technology Survey and the membership approved projects during the last year for enhancement of two of these areas. The market regulation project was specifically designed to incorporate Web Services. The SPLR project was designed to be based on Web Services in following current NAIC development guidelines. The two projects provided the opportunities and necessities for NAIC to purchase the necessary software tools and middleware for rapid Web Services development. Tools include an Enterprise Service Bus, ETL software, Reporting Framework and UDDI. Concurrent with development of the National Portal, NAIC membership had approved a Security Foundation project, also a necessary component of Web Services Deployment. Financial Web Services development remains ongoing to meet state needs and the needs of the National Portal. A User Group is meeting to support staff in these efforts, and it is premature to determine whether a project request should be recommended; however, that group may make such a recommendation in the future through its membership to an appropriate business committee or the IRMC.

Technical Consulting Working Group Charges:

The Technical Consulting Working Group has completed its first charge, consulting with NAIC staff on existing XML schema, recommending that the schema itself or schema policies not be included in the NTA. The TCWG did recommend additions to the NTA in support of Web Services, which were adopted by the IRMC. The second charge regarding a security classification plan has been deferred for consideration by the IS Task Force and IRMC of consolidation with other security and privacy issues for a more comprehensive study.

Attachments

Service Oriented Architecture Components

Reference Implementation
A reference implementation is an example of code used to help others implement their own code using a web service.

Simple Object Access Protocol (SOAP)
SOAP is a standard for exchanging XML-based messages over a computer network, generally using HTTP. SOAP forms the foundation layer of the web services stack, providing a basic messaging framework on which additional abstract layers can build.

Universal Description Discovery and Integration (UDDI)
An online directory that provides organizations a uniform way to describe their services, discover other companies' services and understand the methods required to utilize the available services provided by a specific company.

Web Service
A Web Service is a software system designed to enable program-to-program and computer-to-computer interaction over the Internet or other networks. Software applications written in various programming languages and running on various platforms can use Web Services to exchange data due to the use of open standards. These open standards normally include formatting data in XML (extensible markup language). NAIC databases become virtual extensions of regulator computer systems when data is offered to the states through Web Services.

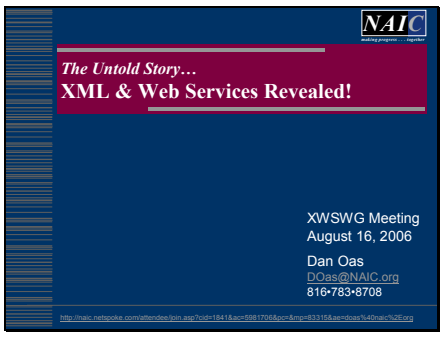
Web Service Description Language (WSDL)
WSDL is an XML-based service description on how to communicate using the web service. The supported operations and messages are described. This means that WSDL describes the public interface to the web service.

eXtensible Markup Language* (*XML)
XML is a general-purpose markup language, which combines text and additional information regarding the text. Its primary purpose is to facilitate the sharing of data across information systems, particularly across the Internet.

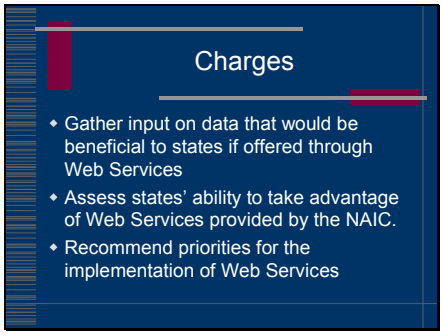
XML Schema
XML Schemas express shared vocabularies and allow machines to carry out rules made by people. They provide a means for defining the structure, content and semantics of XML documents in more detail.

XML Web Services Revealed

Slide 1



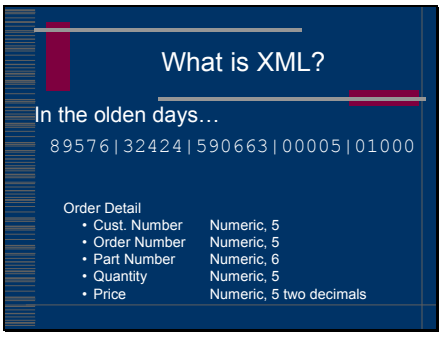
Slide 2



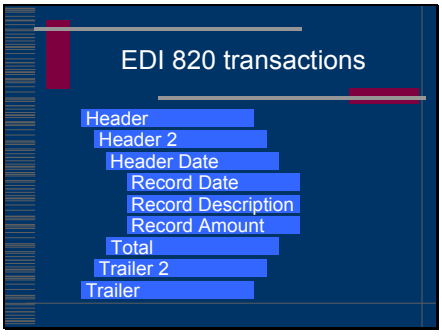
Slide 3

Slide 4

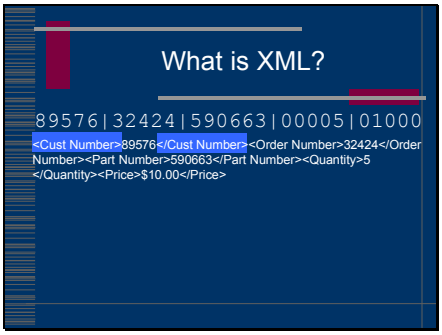
Slide 5



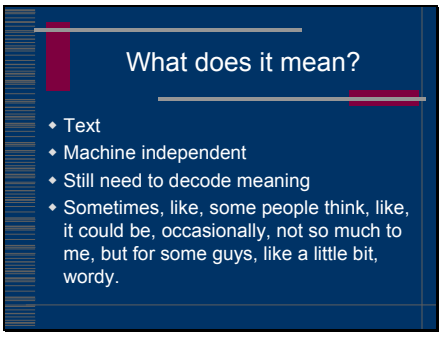
Slide 6



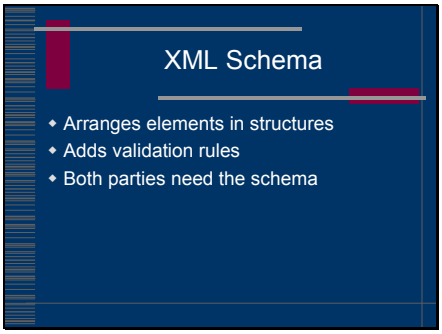
Slide 7



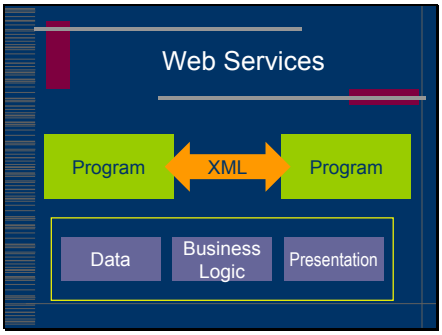
Slide 8



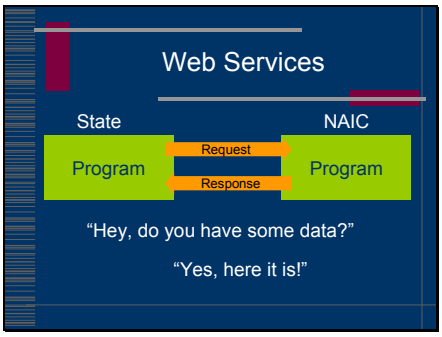
Slide 9



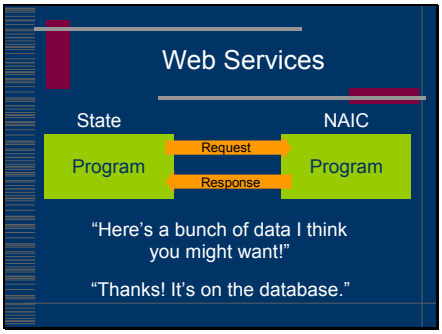
Slide 10



Slide 11

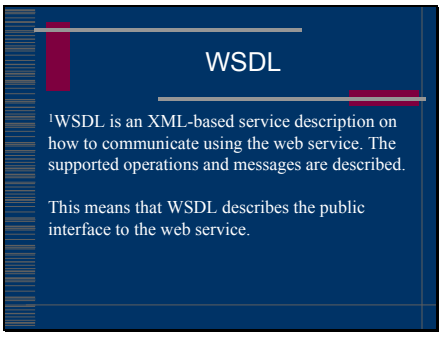


Slide 12

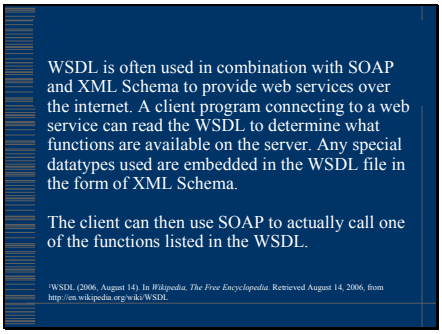


Slide 13

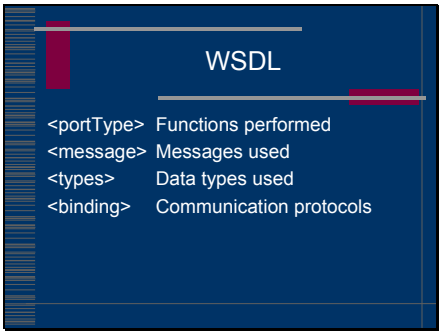
Slide 14



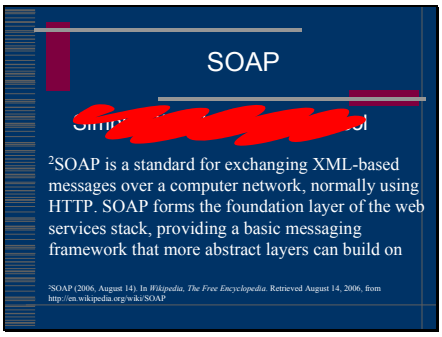
Slide 15



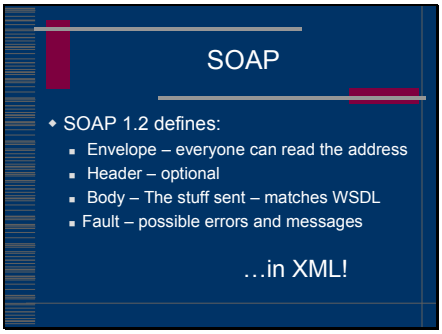
Slide 16



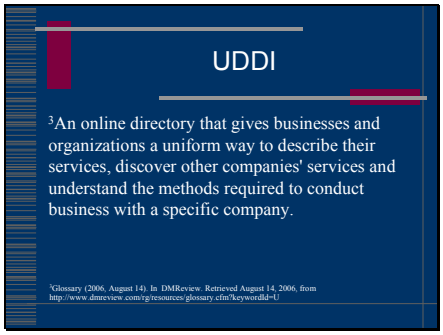
Slide 17



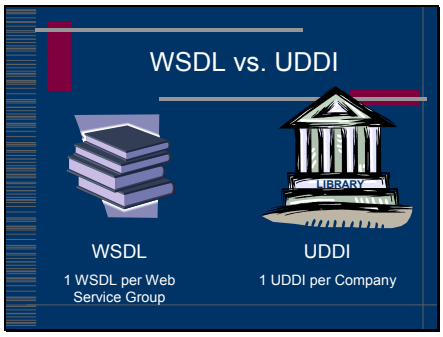
Slide 18



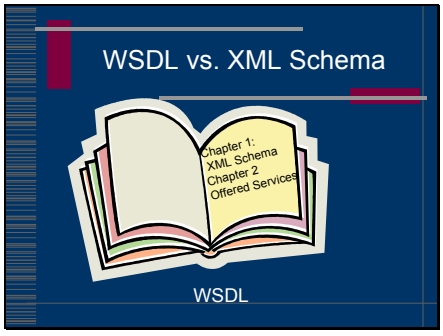
Slide 19



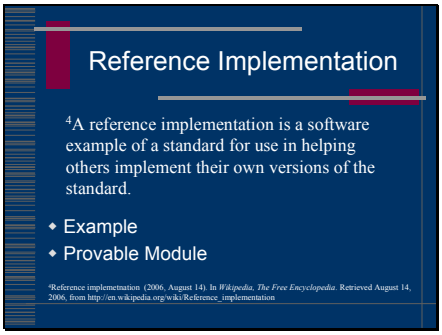
Slide 20



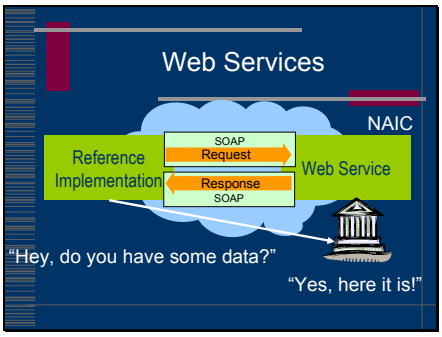
Slide 21



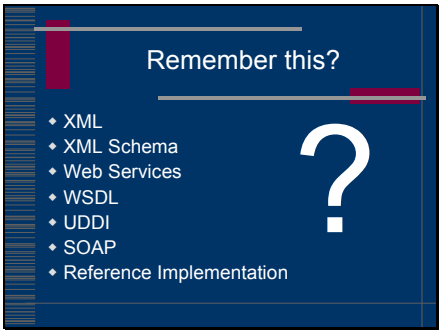
Slide 22



Slide 23



Slide 24



Slide 25

The 6 Characteristics of an NAIC Web Service

1. Exchanges data over the Internet using the http or https protocol
2. Uses SOAP v1.2 to exchange data
3. Uses XML to describe the data in transit
4. Uses WSDL v2.0 to describe the ways to interact with the web service
5. Can be published in a UDDI
6. Has a reference implementation to prove & demonstrate the client.

Slide 26

You *could* use Web Services,

But why?

- Loosely couple systems
- Platform independent
- Single data source

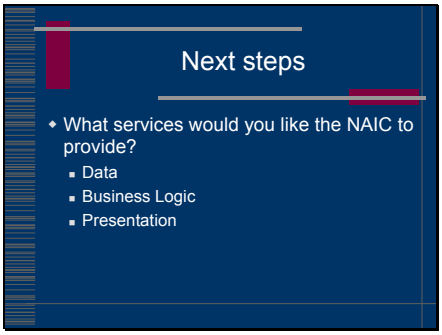
"A national face on our system of state-based insurance regulation"

Slide 27

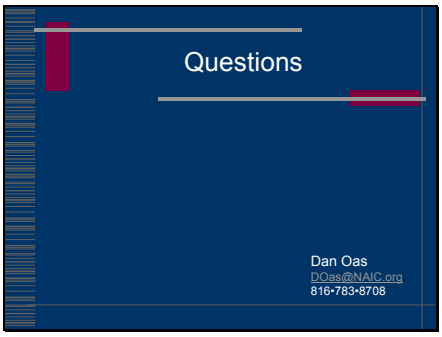
Existing NAIC Web Services

- On-Line Fraud Reporting System (OFRS)
- NIPR State Interface
- SERFF State API/SERFF v5
- I-SITE Financial Reports
- Company Lookup

Slide 28



Slide 29



Financial XML-Web Services

To: XML-Web Services (H) Working Group

From: Ginny Ewing

Date: October 13, 2006

Re: Financial XML-Web Services

This memo attempts to document the research and development efforts surrounding financial xml data and web services – where we've been, where we are, and where we need to be.

Where we've been…

XML/XBRL Research Historical Timeline

Spring 2000 – NAIC Staff began hearing about XBRL initiative and following XBRL International efforts.

Spring 2001 – NAIC Staff received request from Kansas to provide financial XML data. After meeting with them and reviewing their processes, determined bigger issue was accessibility of I-SITE reports. I-SITE was enhanced to more easily provide multiple reports for multiple companies. Next step was to define XML.

Spring/Summer 2002 – NAIC Staff began researching XML; developed sample XML schema, web services, and client tools for Pick-a-Page data.

Summer/Fall 2003 – NAIC Staff exposed and beta tested XML SOAP messaging web service with North Carolina.

September 2003 – H Committee forms XML-Based Open Standards (H) Working Group with charges to: a) Evaluate the options for an XML-based open standard for the NAIC, with specific input of Examination Oversight (E) Task Force members; b) Conduct a public meeting to obtain input on this subject from all interested parties; c) Present to the H Committee at the 2003 Winter National Meeting – a recommendation as to what XML-based open standard the NAIC should adopt for data exchange and; d) If deemed necessary by the Working Group, conduct a pilot project to develop a specific approach to providing an open data-exchange standard based on groupings in XML fields.

Fall 2003 – NAIC Staff put XML/XBRL research and development on hold while XML-Based Open Standards (H) Working Group addressed their charges.

October 2003 – XOSWG holds a public meeting in Chicago on XBRL.

December 2003 – XOSWG makes recommendation and H Committee approves a motion that the NAIC should adopt the formal eXtensible Business Reporting Language (XBRL) as a data reporting standard.

January 2004 – EX1 reviews the H Committee XBRL recommendation and asks for further evaluation of insurance regulator business needs and revenue considerations.

March 13, 2004 – H Committee holds open meeting at NAIC Spring National Meeting to provide more understanding of XBRL and the whole open standards concept. Commissioner Long hosted a group of XBRL experts, financial people and systems people, to answer questions about what XBRL is and what it is not, why we should move forward, the business need for the standard, and revenue and operational implications for the NAIC staff, the state regulators, and the insurance industry.

March 14, 2004 – EX1 approves a $126,000 funding request to conduct a XBRL pilot project.

April 29, 2004 – XBRL Study Group is formed and deliverables drafted and assigned.

October 2004 - December 2005 – XBRL Demonstration Project

January 2005 – NAIC Staff received request from Texas for financial XML data – on hold pending conclusion of XBRL Demonstration project.

Summer 2005 – NAIC Staff received request from SBS to facilitate accessing I-SITE reports via SBS application; received similar request from Florida to access financial I-SITE reports.

December 2005 – The status of the XBRL Demonstration Project was presented. The project had completed four of its five deliverables by hiring XBRL consulting expertise to assist the NAIC staff in developing an XBRL taxonomy and instance documents for a subset of the insurance company financial statement data. While some benefits were derived from developing a taxonomy and instance documents, the next step was for Study Group members to "consume," or use, the data published in XBRL data format using an XBRL analysis-type software tool, in order to evaluate XBRL's capabilities and usefulness for insurance regulators.

March 2006 – The XBRL Study Group reviewed two new XBRL software tools. The XBRL Study Group reviewed two new XBRL software tools. The conclusion was that while both products portrayed promising functionality and features, it was also apparent that only a small number of software vendors had commercially available software tools that are XBRL-enabled. In addition, XBRL had not yet been widely adopted by insurance companies, and, thus, would not yet warrant a recommendation to convert NAIC financial data to the XBRL format (mainly due to the increased costs involved for companies and regulators). The consultant, UBMatrix Inc., noted that the current NAIC FDR data standard already provided many of the benefits users seek from implementing XBRL (i.e., meta-data driven standards; pushing quality to the source; robust validation processes; and expedited "capture-to-publish" turn-around time). Similar to the UK's Financial Services Authority (FDA), the Study Group concluded that while XBRL activity may be on the increase and it is possible that it may well become a major data exchange format in the future, they plan to continue to monitor the evolution and usage of XBRL in 2006 and report back to the Committee on any major developments.

May 2006 – I-SITE Report Web Services released to production. These web services provide access to select financial reports in formats available in I-SITE. While the primary focus of this project was to provide the reports in HTML format, a natural by-product was the availability of the Pick-a-Page XML data via the Web Service. This has not been made widely known, and may need to be removed, as the XML schema has not been validated.

June 2006 – IS Task Force recommended 1) bringing closure to the eXtensible Business Reporting Language (XBRL) Study Group project; 2) creating a working group; and 3) expanding the charges of the Technical Consulting Working Group.

September 2006 – H Committee and EX1 adopted IS Task Force recommendation.

Where we are….

The XML-Web Services (H) Working Group has been formed and is charged to:

1. Gather input from regulator information technology and business staff members regarding data maintained by the NAIC that would be beneficial to states if offered through Web Services.

2. Assess states' ability both from a technical and a resource perspective, to take advantage of Web Services provided by the NAIC. Consider recommendation of a developer track at the 2007 Tech-Ex conference to train state developers interested in leveraging NAIC reference implementations and how to implement NAIC Web Services

3. Recommend priorities for the implementation of Web Services, including relative order of data sets to be implemented. Recommend whether a XML-Web Services project should be proposed to IRMC for sponsorship in the next available budget cycle as a stand-alone project, whether any such project should include software to aid the rapid deployment of Web Services and whether Web Services middleware should be purchased.

A "XML and Web Services" section has been added to the 2006 Insurance Department Technology Survey. These questions were designed to assess the departments' plans to use, and experience using, XML and Web Services. In addition, input into the identification and prioritization of potential NAIC Web Services will be gathered. The survey is currently being piloted and is scheduled for wide release by October 18.

In addition to the formation of the XML-Web Services Working Group, the following additional charges have been assigned to the Technical Consulting (H) Working Group:

1. Provide counsel to NAIC staff in determining appropriate taxonomy, schema/naming conventions and reference implementations as standards for data formatted as XML to be made available through Web Services. Review and approve such standards. Standards will be considered as updates to the National Technical Architecture. The working group's goal will be to present any recommended XML-Web Services standards to the Information Systems Task Force at its December 2006 meeting.

2. Provide counsel to NAIC staff in developing a plan for security classification of data offered through Web Services and a plan for access control based on such classification. Such plan should be presented to the Information Systems Task Force at its March 2007 meeting.

Where we need to be…

Established XML schema/naming conventions and reference implementation standards

System developed for reviewing and prioritizing requests for new Web Services.

Web Services identified, defined, prioritized, and are being implemented.

How to get there…

Review 2006 Insurance Department Technology Survey results. Gather follow up information as necessary, possibly through another short survey or email correspondence.

Establish XML schema/naming conventions and reference implementation and other Web Service guidelines and standards. (TCWG)

Based on survey results define appropriate training for state developers interested in leveraging NAIC reference implementations and how to implement NAIC Web Services.

Recruit interested states to pilot XML and Web Services.

Develop Project Request and obtain approval to develop defined Web Services and offer state training if necessary.

To: XML-Web Services (H) Working Group

From: Ginny Ewing

Date: October 13, 2006

Re: Financial XML-Web Services

This memo attempts to document the research and development efforts surrounding financial xml data and web services – where we've been, where we are, and where we need to be.

Where we've been…

XML/XBRL Research Historical Timeline

Spring 2000 – NAIC Staff began hearing about XBRL initiative and following XBRL International efforts.

Spring 2001 – NAIC Staff received request from Kansas to provide financial XML data. After meeting with them and reviewing their processes, determined bigger issue was accessibility of I-SITE reports. I-SITE was enhanced to more easily provide multiple reports for multiple companies. Next step was to define XML.

Spring/Summer 2002 – NAIC Staff began researching XML; developed sample XML schema, web services, and client tools for Pick-a-Page data.

Summer/Fall 2003 – NAIC Staff exposed and beta tested XML SOAP messaging web service with North Carolina.

September 2003 – H Committee forms XML-Based Open Standards (H) Working Group with charges to: a) Evaluate the options for an XML-based open standard for the NAIC, with specific input of Examination Oversight (E) Task Force members; b) Conduct a public meeting to obtain input on this subject from all interested parties; c) Present to the H Committee at the 2003 Winter National Meeting – a recommendation as to what XML-based open standard the NAIC should adopt for data exchange and; d) If deemed necessary by the Working Group, conduct a pilot project to develop a specific approach to providing an open data-exchange standard based on groupings in XML fields.

Fall 2003 – NAIC Staff put XML/XBRL research and development on hold while XML-Based Open Standards (H) Working Group addressed their charges.

October 2003 – XOSWG holds a public meeting in Chicago on XBRL.

December 2003 – XOSWG makes recommendation and H Committee approves a motion that the NAIC should adopt the formal eXtensible Business Reporting Language (XBRL) as a data reporting standard.

January 2004 – EX1 reviews the H Committee XBRL recommendation and asks for further evaluation of insurance regulator business needs and revenue considerations.

State Survey Questions: XML and Web Services

A new NAIC working group has been formed to propose a plan to deploy Web Services to facilitate interactive computing between states and the NAIC. Another working group is studying the addition of XML and Web Services standards to the NAIC National Technical Architecture. These questions will aid those working groups and ensure that Web Services are deployed to provide maximum benefit to state departments. Your complete and considered responses are critical to planning a more interactive computing environment among the states and NAIC.

1. What is your department's experience with ***consuming*** (using) Web Services?
   - ❑ No plans or experience
   - ❑ Planning an application to consume Web Services
   - ❑ Developing an application to consume Web Services
   - ❑ In production with an application consuming Web Services

2. Applications ***consuming*** (using) Web Services were/are developed by:
   - ❑ No plans or experience
   - ❑ Department Staff
   - ❑ State Staff
   - ❑ Consultant/Outsourced Staff
   - ❑ Third-party software
   - ❑ Other (please specify)

3. Applications ***consuming*** (using) Web Services were/are developed in:
   - ❑ No plans or experience
   - ❑ Java
   - ❑ .Net
   - ❑ Other (please specify)

4. What is your department's experience with ***hosting*** Web Services?
   - ❑ No plans or experience
   - ❑ Planning an application to host Web Services
   - ❑ Developing an application to host Web Services
   - ❑ In production with an application hosting Web Services

5. Applications ***hosting*** Web Services were/are developed by:
   - ❑ No plans or experience
   - ❑ Department Staff
   - ❑ State Staff
   - ❑ Consultant/Outsourced Staff
   - ❑ Third-party software
   - ❑ Other (please specify)

6. Applications ***hosting*** Web Services were/are developed in:
   - ❑ No plans or experience
   - ❑ Java
   - ❑ .Net
   - ❑ Other (please specify)

7. Would department personnel participate in training on implementing NAIC Web Services?
   - ❑ No Interest
   - ❑ A track during the 2007 E-Reg/TechEx conference in Kansas City
   - ❑ A course after the 2007 E-Reg/TechEx conference in Kansas City
   - ❑ A course some other time in Kansas City
   - ❑ A course some other place/time. When/Where?

8. What is your department's use of the NAIC's ***NIPR State API/State License*** Web Service?
   - ❑ No plans or experience
   - ❑ Planning an application to host Web Services
   - ❑ Developing an application to host Web Services
   - ❑ In production with an application hosting Web Services

9. What is your department's use of the NAIC's ***SERFF API 3.0*** Web Service?
   - ❑ No plans or experience
   - ❑ Planning an application to host Web Services
   - ❑ Developing an application to host Web Services
   - ❑ In production with an application hosting Web Services

10. What is your department's use of the NAIC's ***I-SITE Financial*** Web Service?
   - ❑ No plans or experience
   - ❑ Planning an application to host Web Services
   - ❑ Developing an application to host Web Services
   - ❑ In production with an application hosting Web Services

11. What is your department's use of the NAIC's ***Online Fraud Reporting System*** Web Service?
   - ❑ No plans or experience
   - ❑ Planning an application to host Web Services
   - ❑ Developing an application to host Web Services
   - ❑ In production with an application hosting Web Services

12. What other non-NAIC Web Services are planned? (List all)

13. What other non-NAIC Web Services are in development? (List all)

14. What other non-NAIC Web Services are in production? (List all)

15. Do/will you use XML ***other than*** with the Web Services previously listed?
   - ❑ No
   - ❑ Yes (please explain)

16. How should NAIC staff prioritize Web Service development?
   - ❑ Web Services for states to retrieve NAIC data
   - ❑ Web Services for states to submit NAIC data
   - ❑ A balanced approach for state retrieval and submission of data
   - ❑ Should not be a 2007 priority for NAIC staff

These questions provide you with the opportunity to help prioritize which Web Services are deployed at NAIC. Besides the Web Services already available (*NIPR State API/State License, SERFF API 3.0*, etc.) what would you be most likely to use? Be as specific as possible in describing what information you would either ***receive from*** or ***submit to*** the NAIC.

17. What would the ***first*** priority Web Service your state would use?
18. What would the ***second*** priority Web Service your state would use?
19. What would the ***third*** priority Web Service your state would use?
20. What would the ***fourth*** priority Web Service your state would use?
21. What would the ***fifth*** priority Web Service your state would use?

22. Would you be interested in participating in the NAIC XML/Web Services Working Group?

Financial Web Services Interview Summary

To: Financial Web Services Technical Group

From: NAIC Staff

Date: April 20, 2007

Re: Follow Up Interview Summary

At the direction of the XML–Web Services Working Group (XWSWG), NAIC staff conducted follow-up interviews with respondents to the 2006 Insurance Department Technology Survey. The purpose of the interviews was to obtain more details, both from a business and technical perspective, regarding the types of financial web services the states would like the NAIC to provide. The interviews were designed with the following objectives in mind:

- gather specifics regarding the type and format of data requested;
- determine department's urgency and readiness to utilize the Web services;
- further educate other regulators on how Web services work and the opportunities they provide;
- assess Web services training needs; and
- identify states interested in participating with the Working Group.

Selection of the states that were included in the follow-up interviews was based on responses provided in the Technology Survey indicating financial web services as a high priority. Based on this criterion, the following states were identified and contacted:

- Colorado
- Kansas
- Pennsylvania
- South Carolina
- Tennessee
- Virginia

Summary of state responses

Current Uses
The states consistently commented that their primary access of NAIC financial data is through the I-SITE tool. States are predominately utilizing I-SITE to download reports in HTML and PDF formats. Within I-SITE, users are accessing Financial Analysis Handbook, Profiles, IRIS, FAST Scoring, Analyst Team System, and Exam Jumpstart reports.

States included in the survey utilize Microsoft Access to pull financial data into TeamMate. From there they each have some mechanism to pull financial data into their own "home-grown" systems (i.e., Access, Excel) and generate reports for further analysis.

Pennsylvania and South Carolina commented that they based their analysis off of the I-SITE reports and use them as source documents.

Future Uses
Two of the states surveyed stated that current access methods that are available meet their needs. Tennessee commented that they did not foresee any new requirements for additional financial data. Pennsylvania specifically mentioned that they would be interested in a service that would allow them to extract data to support accreditation requirements. South Carolina was looking for a more convenient method of accessing State Page detail. Colorado would like to be able to pull market analysis data at a group level. They also expressed an interest in accessing AM Best and Standard & Poor's data as well as data for long-term care supplements. Kansas was specifically interested in the NAIC providing a better presentation/consolidation of analysis data (e.g., IRIS, Scoring, Profiles, etc.). They would like to have an ability to drill down further into the results provided by these reports.

Expectations
South Carolina, Tennessee, and Virginia commented that they were not currently in a position, either technologically or on a resource level, to be able to support any new web services immediately. All indicated that they would certainly be interested in this technology, but would need to further weigh resource and cost requirements.

Pennsylvania stated that they would be ready when the NAIC had services available for them to consume. Colorado is ready from an IT perspective, but it would not be a high priority for them at this juncture. Kansas stated emphatically that they are ready today. They were particularly looking for a web service to facilitate their tax application requirements.

All states surveyed indicated that they would need reference implementations using either Java, .net, or Excel.

When addressing the question of what type of web services would be most beneficial, most of the states were consistent in stating that they wanted "financial data". Kansas was more specific in their requirements. They noted that they would like to see a service providing data from annual statements for validation of amounts reported for taxes as well as a company/group maintenance service allowing them the ability to update company demographic information.

Training Needs
All states consistently indicated that they would be interested in additional web services training. Suggestions ran from online training, webinars, and hands on technical training (either at TechEx or onsite either at the state or NAIC).

Kansas and Tennessee stated that they would be interested in providing training to other states/departments and sharing their web service experiences.

Future Participation
The majority of the states indicated interested in participating in a Financial Web Services technical subgroup of the XML-Web Services Working Group.

March 13, 2004 – H Committee holds open meeting at NAIC Spring National Meeting to provide more understanding of XBRL and the whole open standards concept. Commissioner Long hosted a group of XBRL experts, financial people and systems people, to answer questions about what XBRL is and what it is not, why we should move forward, the business need for the standard, and revenue and operational implications for the NAIC staff, the state regulators, and the insurance industry.

March 14, 2004 – EX1 approves a $126,000 funding request to conduct a XBRL pilot project.

April 29, 2004 – XBRL Study Group is formed and deliverables drafted and assigned.

October 2004 - December 2005 – XBRL Demonstration Project

January 2005 – NAIC Staff received request from Texas for financial XML data – on hold pending conclusion of XBRL Demonstration project.

Summer 2005 – NAIC Staff received request from SBS to facilitate accessing I-SITE reports via SBS application; received similar request from Florida to access financial I-SITE reports.

December 2005 – The status of the XBRL Demonstration Project was presented. The project had completed four of its five deliverables by hiring XBRL consulting expertise to assist the NAIC staff in developing an XBRL taxonomy and instance documents for a subset of the insurance company financial statement data. While some benefits were derived from developing a taxonomy and instance documents, the next step was for Study Group members to "consume," or use, the data published in XBRL data format using an XBRL analysis-type software tool, in order to evaluate XBRL's capabilities and usefulness for insurance regulators.

March 2006 – The XBRL Study Group reviewed two new XBRL software tools. The XBRL Study Group reviewed two new XBRL software tools. The conclusion was that while both products portrayed promising functionality and features, it was also apparent that only a small number of software vendors had commercially available software tools that are XBRL-enabled. In addition, XBRL had not yet been widely adopted by insurance companies, and, thus, would not yet warrant a recommendation to convert NAIC financial data to the XBRL format (mainly due to the increased costs involved for companies and regulators). The consultant, UBMatrix Inc., noted that the current NAIC FDR data standard already provided many of the benefits users seek from implementing XBRL (i.e., meta-data driven standards; pushing quality to the source; robust validation processes; and expedited "capture-to-publish" turn-around time). Similar to the UK's Financial Services Authority (FDA), the Study Group concluded that while XBRL activity may be on the increase and it is possible that it may well become a major data exchange format in the future, they plan to continue to monitor the evolution and usage of XBRL in 2006 and report back to the Committee on any major developments.

May 2006 – I-SITE Report Web Services released to production. These web services provide access to select financial reports in formats available in I-SITE. While the primary focus of this project was to provide the reports in HTML format, a natural by-product was the availability of the Pick-a-Page XML data via the Web Service. This has not been made widely known, and may need to be removed, as the XML schema has not been validated.

June 2006 – IS Task Force recommended 1) bringing closure to the eXtensible Business Reporting Language (XBRL) Study Group project; 2) creating a working group; and 3) expanding the charges of the Technical Consulting Working Group.

September 2006 – H Committee and EX1 adopted IS Task Force recommendation.

Where we are….

The XML-Web Services (H) Working Group has been formed and is charged to:

1. Gather input from regulator information technology and business staff members regarding data maintained by the NAIC that would be beneficial to states if offered through Web Services.

2. Assess states' ability both from a technical and a resource perspective, to take advantage of Web Services provided by the NAIC. Consider recommendation of a developer track at the 2007 Tech-Ex conference to train state developers interested in leveraging NAIC reference implementations and how to implement NAIC Web Services

3. Recommend priorities for the implementation of Web Services, including relative order of data sets to be implemented. Recommend whether a XML-Web Services project should be proposed to IRMC for sponsorship in the next available budget cycle as a stand-alone project, whether any such project should include software to aid the rapid deployment of Web Services and whether Web Services middleware should be purchased.

A "XML and Web Services" section has been added to the 2006 Insurance Department Technology Survey. These questions were designed to assess the departments' plans to use, and experience using, XML and Web Services. In addition, input into the identification and prioritization of potential NAIC Web Services will be gathered. The survey is currently being piloted and is scheduled for wide release by October 18.

In addition to the formation of the XML-Web Services Working Group, the following additional charges have been assigned to the Technical Consulting (H) Working Group:

1. Provide counsel to NAIC staff in determining appropriate taxonomy, schema/naming conventions and reference implementations as standards for data formatted as XML to be made available through Web Services. Review and approve such standards. Standards will be considered as updates to the National Technical Architecture. The working group's goal will be to present any recommended XML-Web Services standards to the Information Systems Task Force at its December 2006 meeting.

2. Provide counsel to NAIC staff in developing a plan for security classification of data offered through Web Services and a plan for access control based on such classification. Such plan should be presented to the Information Systems Task Force at its March 2007 meeting.

Where we need to be…

Established XML schema/naming conventions and reference implementation standards

System developed for reviewing and prioritizing requests for new Web Services.

Web Services identified, defined, prioritized, and are being implemented.

How to get there…

Review 2006 Insurance Department Technology Survey results. Gather follow up information as necessary, possibly through another short survey or email correspondence.

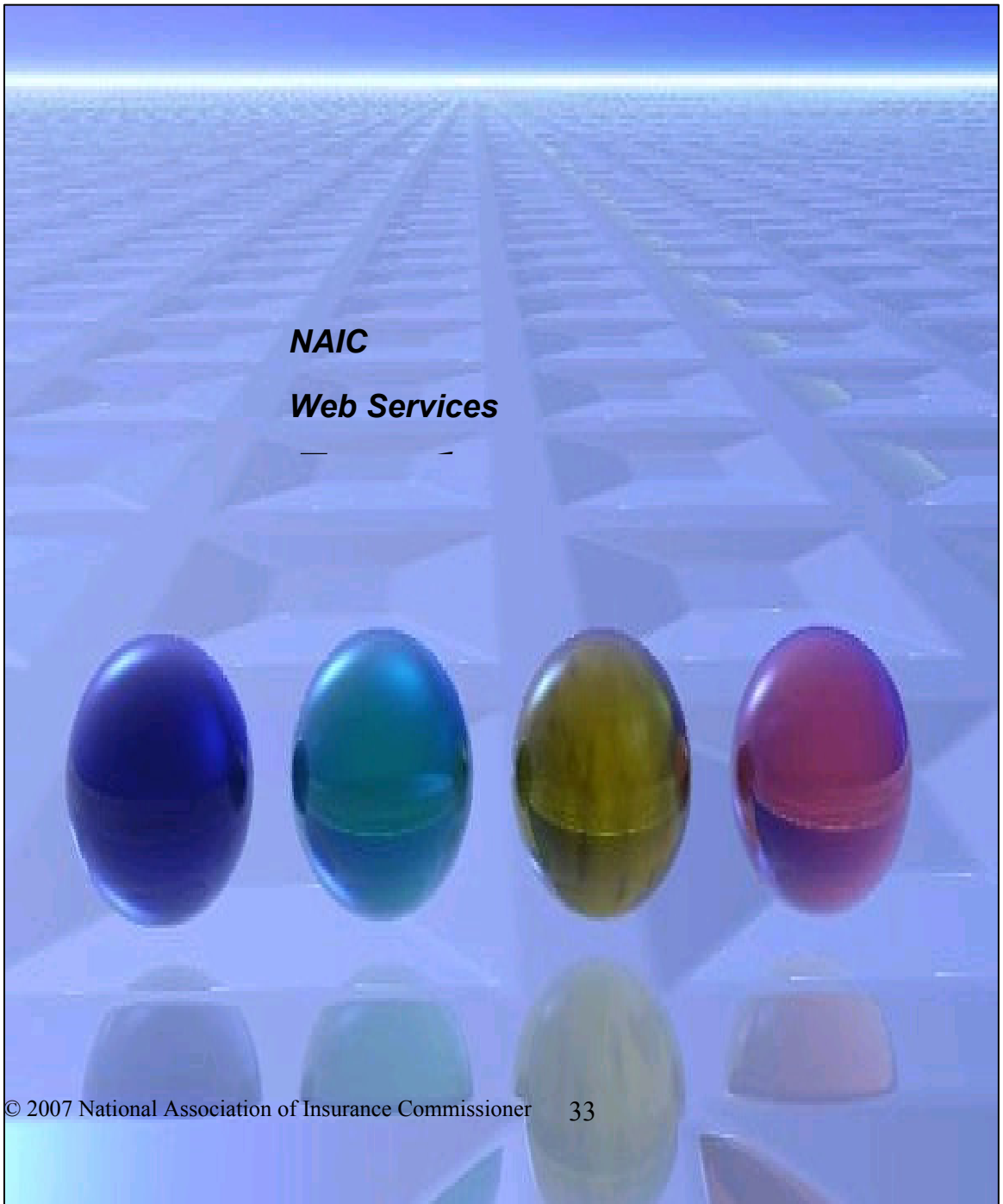
Establish XML schema/naming conventions and reference implementation and other Web Service guidelines and standards. (TCWG)

Based on survey results define appropriate training for state developers interested in leveraging NAIC reference implementations and how to implement NAIC Web Services.

Recruit interested states to pilot XML and Web Services.

Develop Project Request and obtain approval to develop defined Web Services and offer state training if necessary.

NAIC Web Services



Last Updated: 04/14/2006

Author: Architect Team

Table of Contents

Introduction

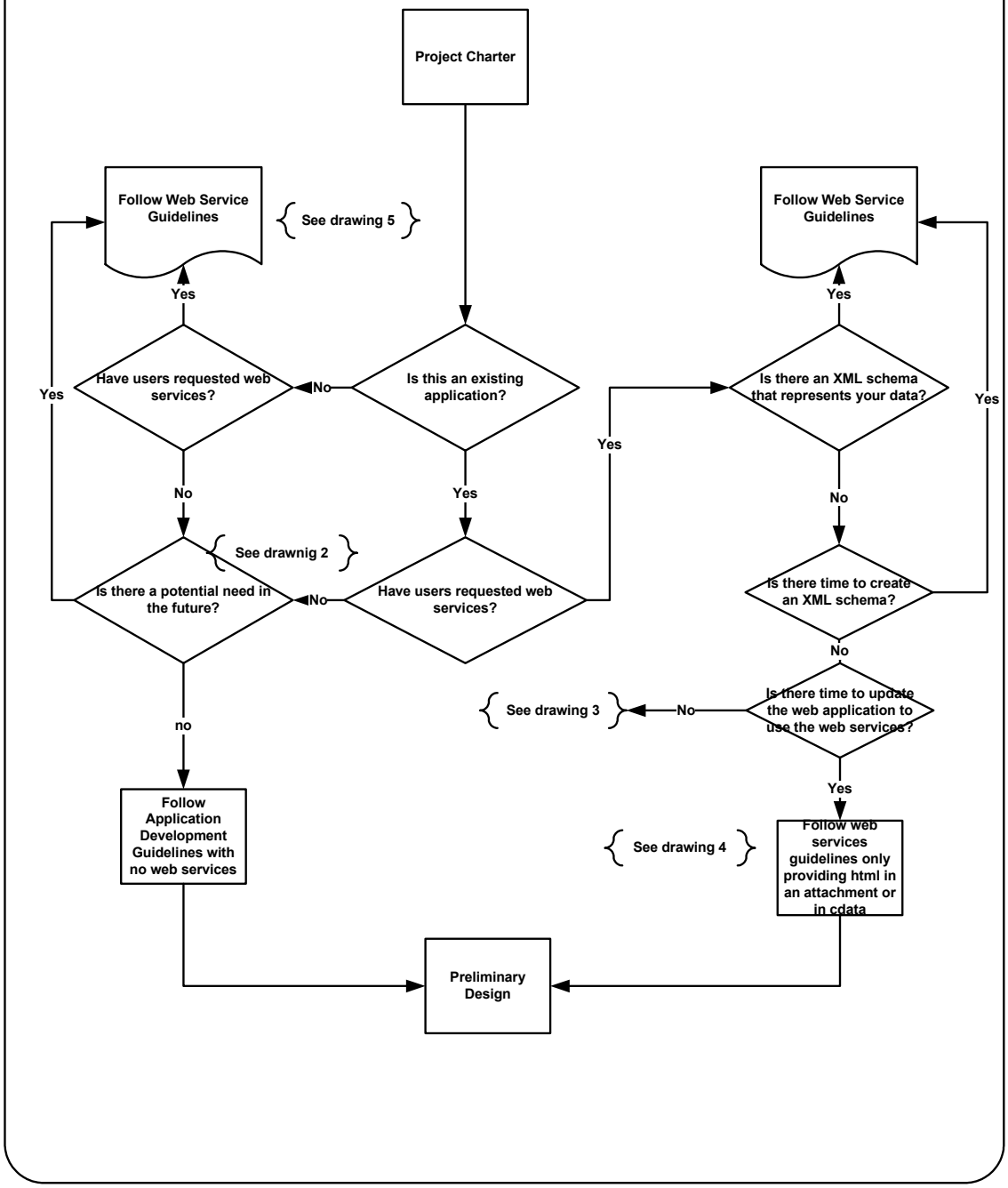
This guide will provide development areas with a decision tree on whether to use web services or not. It will also provide some implementation guidelines, which include phased in approaches.

Obviously, there will be times where this document does not cover all scenarios, and in those cases the Architects and the development areas must work together to make the call on web services. Saying this though, this guide should come pretty close to drawing that line in the sand between web services and no web services.

In Appendix A you can also review the Web Service coding standards (these are referred to as Web Service Guidelines in this doc).

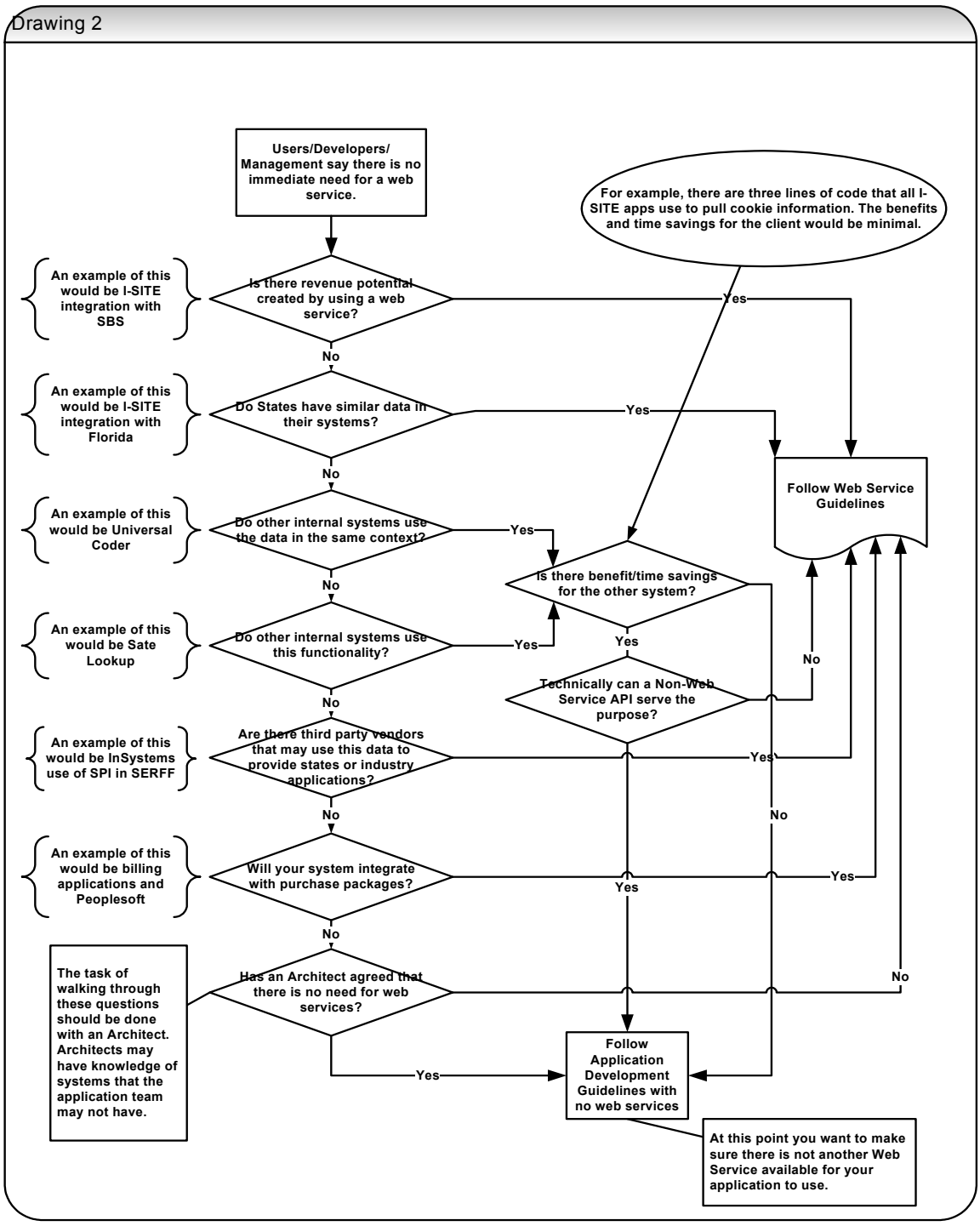
Overview

Drawing 1



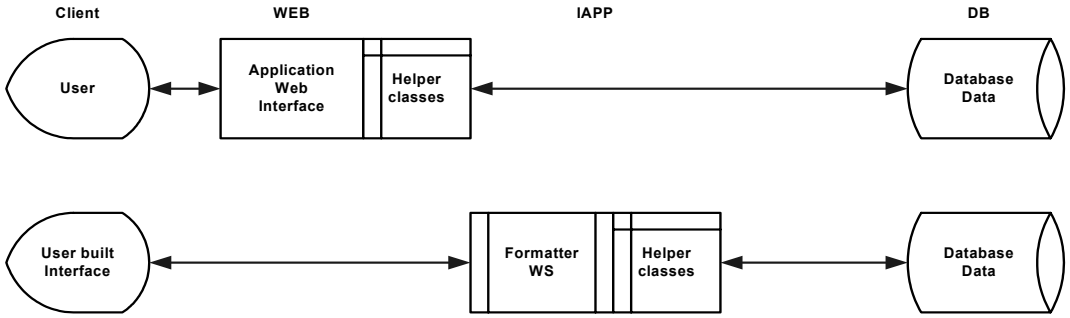
No immediate need, but is there a potential need?

Drawing 2



There is no time to update the web application to use the web services

Drawing 3

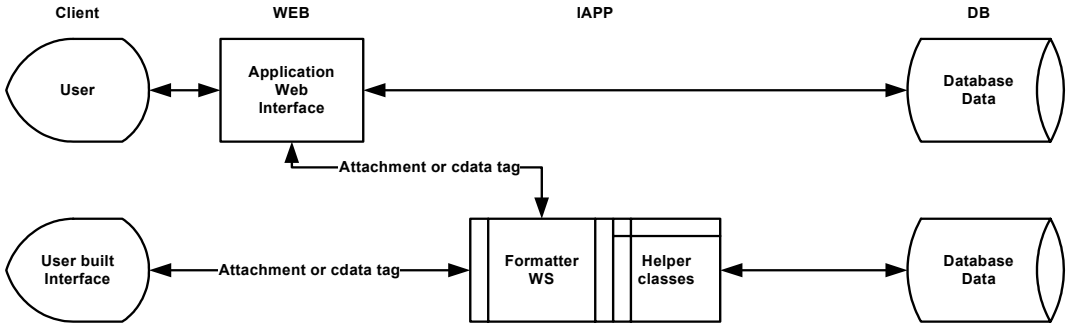


This drawing illustrates how you could deploy web services, and not change your existing application. Basically, it just relies on copying helper classes from the web server to the iApp server.

This is not a great solution by any means, but there may be times where deadlines dictate this direction.

Follow web services guidelines only providing html in an attachment or in cdata

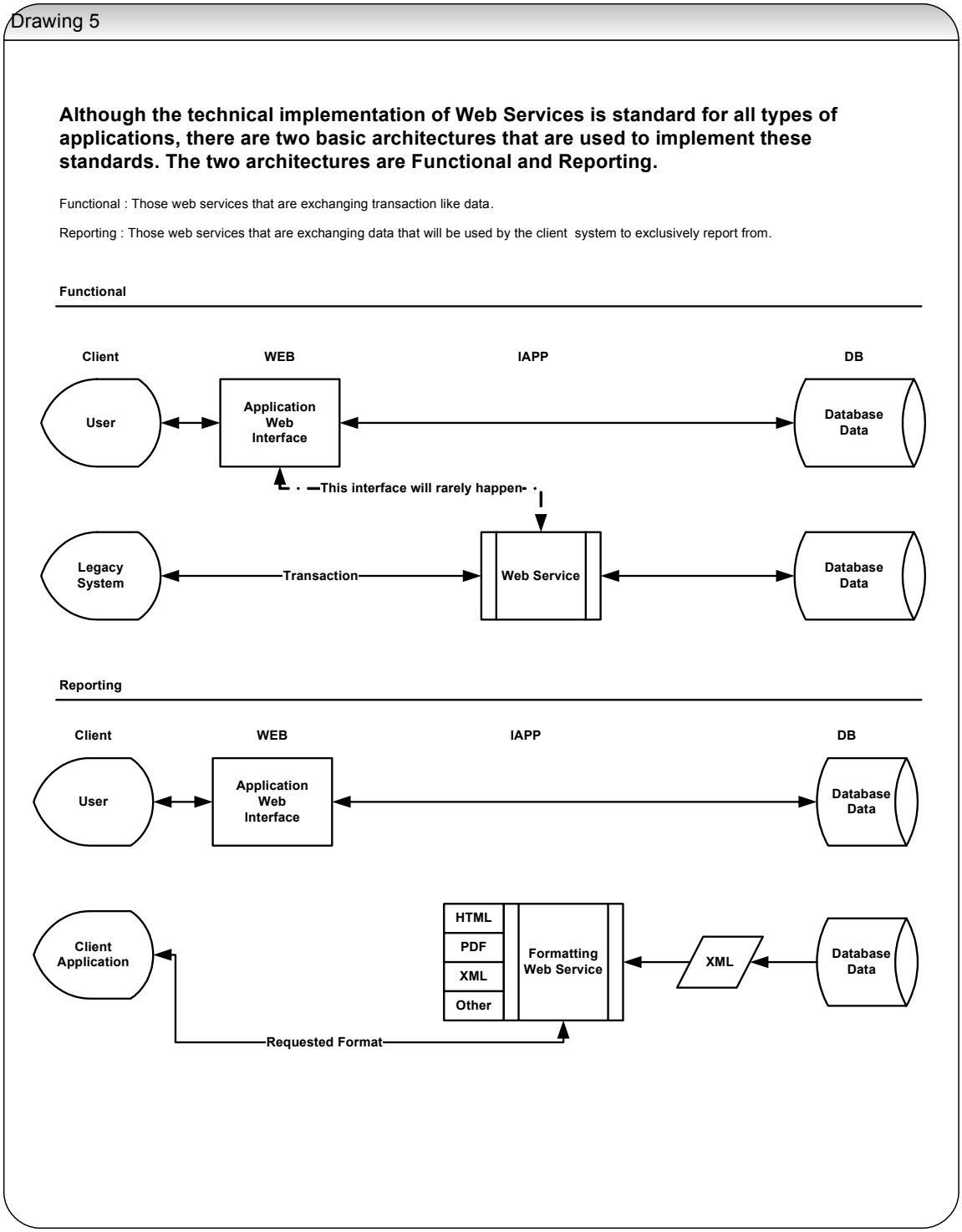
Drawing 4



This drawing illustrates how web services would be implemented if there were not time to do the “Reporting” implementation outlined in drawing 5. Basically, the above architecture excludes the formatting web service, and only provides HTML as the only option.

Functional or Reporting Web Service?

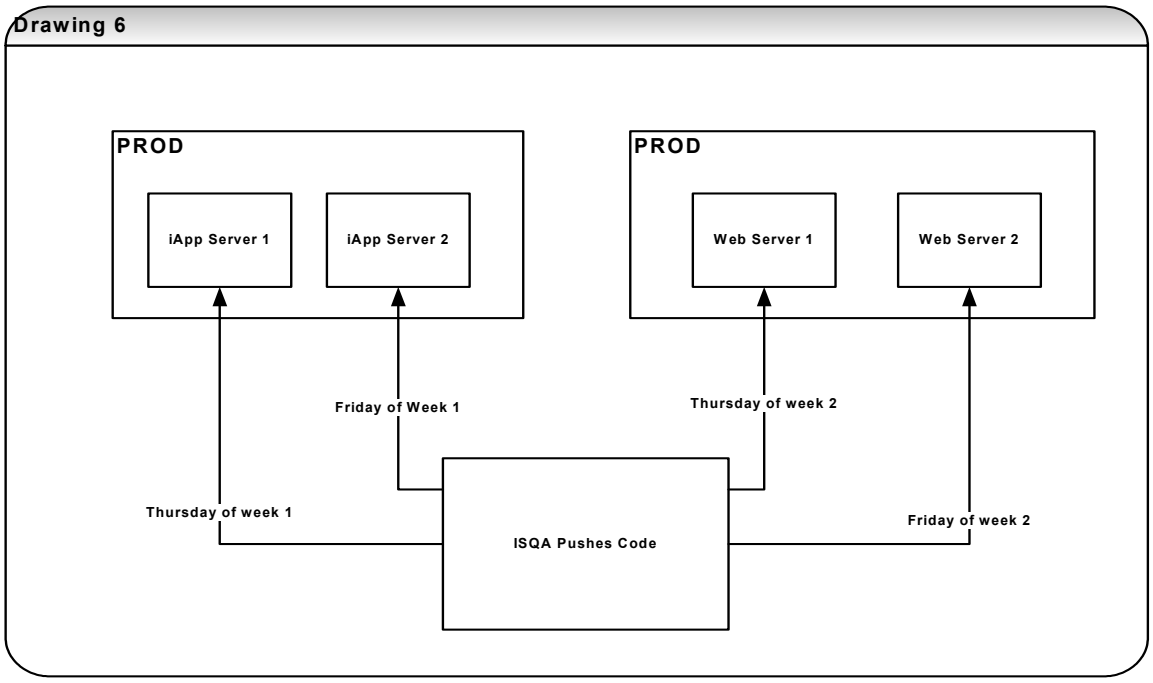
Drawing 5



Production pushes of Web Services that also have Web Server Components

In the web service architecture you will be required to deploy the web services to the iApp server, and all user interface components to the Web Server. For many reasons this will cause the time it takes to push web service applications to PROD to span over two weeks. See drawing 6 below.

Keep in mind though, all web services do not provide a user interface component, and those can still be handled in the current two-day scenario.



Tech Services and the Application groups are currently working on how to get this down to the standard two-day time frame. Currently both areas have some ideas on how to do this, but none of the ideas provide a solid solution yet. So until further notice, all web services with web server components being pushed to PROD will span over two weeks.

A Manager's Checklist

Introduction

Since Web Services is a new way to build applications at the NAIC there are things to make sure you think of that you did not need to worry about in the previous way to architect applications.

Following are areas to keep in mind when planning for a web service project.

XML Schema

There may be times when the need for a web service arises, and the XML has not been defined yet. This happens mostly to reporting web services, and in this case we will release only providing the HTML in a cdata tag or an attachment. As the manager you must be aware that at some point we will need to go back and update the web service once the XML is defined.

Web Interface should use the same services as your external clients use.

As you know the web services will reside on the iApp server, and your web interface will reside on the Web Server. We do not want to have the same business rules in two different places. Saying this though, there may be occasions where deadlines have been set, and to expedite the release of the web service this architecture will be allowed. As the manager you must be aware that at some point we will need to go back and update the we application to use the web services on the iApp box.

Plan for rollout to PROD to span over two weeks. {Maybe?}

{Working on this problem now}

Reference implementation

A Reference implementation will be required for all web services that will be used by external customers. There are two main reasons for making this a requirement. First of all, the reference implementation will hopefully be used by the clients to expedite their use of the service. And secondly, there are legal issues that need to be addressed so the clients know what the NAIC is and is not responsible for when it comes to supporting the web services.

Supporting your clients

Even though in the reference implementation we lay out what we will do and we will not do, there will most likely be more questions than there were in the previous architecture. Remember the users of the web service are not the end-users, but the developers that are writing the clients. For this reason you should plan on your developers working to some extent with the developers that are developing the clients. It is the goal, and even a requirement for making this successful that we put ourselves in a position that will minimize this type of support.

Supporting versions

Unlike in the previous architecture, we will need to support multiple versions of the web service. We are looking at supporting two versions at once, and only supporting the prior version six months after the newest version is released. This has yet to be approved by the IS Task Force, but hopefully it will be soon. So as the Manager you will need to plan for this new responsibility.

Appendix A

Introduction

A web service is just a remote method invocation using a standard XML encoding for data sent to and from the method over the http or https protocol. The best use of a web service is to provide an open interface to clients so that they can use the method externally to the application. An application should use web services to provide its business rules to other internal or external application.

The most fundamental level of the web services infrastructure used by the NAIC is based on the same concepts of a web form. A browser sends data in a standard format captured on a web form to a web server. The web server passes this data to a CGI or Java servlet container (e.g. JRun) with the POST data coming from the standard input stream for a CGI and some other communication method for a servlet container. The application generates a text response which the web server returns to the web browser. A web service does the exact same behavior but the format of the data for input and output follows a different standard.

SOAP Messaging

The SOAP standard version 1.2 contains the basic format specification of the data sent and received by web services called SOAP Messaging. Amending the SOAP standard is WS-I Basic Profile. WS-I promotes of Web Services across operating systems, platforms, and languages. It is a large industry organization that includes Sun, IBM, Oracle, and Microsoft. The basic profile places more requirements than the SOAP standard on messages. The data sent between the client and the server are called messages in the standard and follow an XML format. The following example shows the required tags for a message:

```
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
  <soap-env:Body>
  ....
  </soap-env:Body>
</soap-env:Envelope>
```

The Envelope tag is the root tag of all SOAP messages and it is defined in the http://schemas.xmlsoap.org/soap/envelope/ namespace which is given an alias of soap-env. The next tag, Body, is the container for any user data in the message.

As with any web-based technology, communication and server errors can occur. For these types of errors, SOAP defines the same behavior as web servers currently perform. This behavior is to return an error code in the header with supporting information. These error codes are standard and SOAP implementations know how to handle them.

In addition to server errors, the SOAP standard addresses defines the communication of program errors to a client. This example shows the required tags for a program error:

```
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope">
  <S:Body>
    <S:Fault>
      <faultcode>S:Server</faultcode>
      <faultstring>Server Error</faultstring>
      <detail> .... </detail>
    </S:Fault>
  </S:Body>
</S:Envelope>
```

For this example, the same namespace, http://www.w3.org/2001/12/soap-envelope, as above is aliased by S. Notice that Envelope and Body are the same as the above example. The faultcode is the application supplied error code and is required. The faultstring is the explanation of the error and is

required. The optional faultactor is an intermediate recipient to whom a message was routed. Lastly, the detail tag gives the application specific error information related to the Body element processing and is required if processing the message body.
In addition to the standard Envelope and Body tags, SOAP allows a Header tag. The Web Services Security (WS-Security) specification for passing username and password utilizes the Header tag to send this information. The following example shows the security header.

```
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope">
  <S:Header>
    <se:Security xmlns:se="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-secext-1.0.xsd">
      <se:UsernameToken>
        <se:Username>swa</se:Username>
        <se:Password>swapw</se:Password>
      </se:UsernameToken>
    </se:Security>
  </S:Header>
  <S:Body>
   ....
  </S:Body>
</S:Envelope>
```

The Security tag is the container for all the authentication tags. It can contain other security authentication tags than given in the example above. UsernameToken contains the Username and Password tags. An application uses the data in these two tags for authentication. This message is sent over SSL between the server and the client.
For SOAP-Messaging, the developer should publish a XML schema describing the SOAP body XML content. Optionally, an application can validate the incoming request against the schema. For a XML schema, the file should follow this layout:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation>
      [package].[method] - [request or response]
      [description of request or response]

      Copyright 2004 National Association of Insurance Commissioners
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name=[name]>
    <xsd:annotation>
      <xsd:documentation>
        [description]
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  ...
</xsd:schema>
```

Adding the annotation tags to the element tags provides a data dictionary for the XML and allows for a program to generate other documentation about the web service.

XML Parsing

For SOAP Messaging, the server and client code parse the XML request and response. SAX and DOM parsers provide a standard way to do this task. A SAX parser provides an event-based model where the parser calls a method when a tag, string, or error is encountered. The most common methods are startElement, characters, and endElement. The parser calls the startElement method when it encounters a new tag and the endElement method at the closing of a tag. It calls the characters method when it encounters the text between tags. Using these methods, the developer writes code to track the state of the parsing and captures the data value into structures or objects. A DOM parser takes the XML data and creates a tree that represents the document. The main node types are Document, Element, and Text Node. The developer writes code that transverses the tree to get data from it. For creating XML data, the developer can create a DOM tree and have it converted into XML data. A DOM parser is usually implemented using a SAX parser and is slower than SAX parsing, but for small messages, there is little speed difference. For Java developers, the web services developer's kit provides JAXB (Java API for XML Binding). JAXB creates classes and interfaces from an XML schema that can parse XML data into objects and create XML data from objects. The creation of XML data from an object is called marshalling an object and unmarshalling an object is the creation of an object or objects from XML data. JAXB is implemented using a SAX parser and is very efficient. All of these parsing methods provide optional schema validation.

SOAP RPC

Built on top of SOAP Messaging is SOAP RPC (remote procedure call) which is also part of the SOAP standard. SOAP RPC defines the contents of the SOAP-BODY, the contents of the SOAP-FAULT, and the datatype mappings. A datatype mapping details the transformation of SOAP datatypes (e.g. SOAP-INT) into a programming language data type (e.g int in Java and C++). The purpose of SOAP RPC is to provide a way for a developer to code using objects that use SOAP RPC to define the communication layer. This communication layer is mostly hidden from the developer.

The developer puts the description of the methods that a SOAP RPC service provides into a WSDL (Web Service Definition Language) file. While tools exist to generate WSDL from class or interface declarations in a programming language, the developer must not use these tools automatically but should control the interface defined in the WSDL over time. This document will discuss this problem in the change management section. Given a WSDL file, a developer generates server or client code using a tool provided in a development kit. These generated classes combined with the development tool kit hide the SOAP RPC communication and the developer uses them to implement the server and client code.

SOAP WSDL Binding Styles

WSDL supports both messaging and RPC using a binding style. The standard allows four styles, RPC/encoded, RPC/literal, Document/Encoded, and Document/Literal. There is also a convention for Document/Literal called wrapped Document/Literal. RPC/encoded is the SOAP RPC method discussed above. It follows the convention of the method name as the root tag and parameters as children. Each parameter has the xsi:type attribute set to the SOAP datatype. RPC/literal follows the same convention of RPC/encoded except the xsi:type attribute is not in the message body of the data. Document/Encoded is a binding style that is not used and might be removed from the standard. Document/Literal allows the WSDL schema to use any valid XML schema to describe the request, response, and error messages. Wrapped Document/Literal is just the same as Document/Literal except the root tag for the message is the name of the message. A good article describing WSDL binding is at http://www-106.ibm.com/developerworks/webservices/library/ws-whichwsdl/.

The rest of this document describes messaging verse RPC, development guidelines, Sun's Web Services Developer Kit, SOAP messaging and JAXB, deployment of the web service, change management, and example code.

Messaging, RPC/Encoded, and Document/Literal
Most of the web services that we have created today follow low-level SOAP messaging. No WSDL file exists. All of these web services should be Document/Literal or wrapped Document/Literal. There are three acceptable ways to do a web service, RPC/Encoded, Document/Literal, and wrapped Document/Literal. All of these require doing a WSDL file describing the web service.
Since there are three ways to provide a web service, the question arises of when to use a certain method. While there is no simple answer to the question, the following guidelines might provide a solution:

- If the developer wants to provide a very detailed contract between the service and the client, the service should be Document/Literal or wrapped Document/Literal. By using XML schemas for the request, response, and error messages, the developer puts more validation into the contract between the service and its clients.
- If the request or response message is large in nature like a report, the service could be Document/Literal or wrapped Document/Literal. Another option is to use RPC/Encoded and the xsi:any datatype. This option should not be used since it eliminates tools from being able to generate parsing code for the xsi:any data.
- The web service is small, quick and the developer doesn't need additional validation, use RPC/Encoded.
- If the service and client want to be insulated from the involving standards and easier to upgrade, the service should be RPC based.
- Basically, if you are not sure, it should be wrapped Document/Literal.

Web Service verse Database View
A common question is when to use a web service or a database view. For external clients, the only approach is a web service except for something very unusual. Please ask an application architect about how to handle this design decision. For internal clients, the most significant factor is the size of the input and output data. If either datasets become very large (i.e. around 1 MB.) then a database view should be considered. Another reason to use a database view is the speed of the web service cannot handle the transaction load of the client application. In order to use a view, a prototype web service must be created and the developer must prove that the web service does not handle the load. Using a view is a last alternative since it exposes part of the implementation logic and reduces the flexibility of the application. The design might be restructured to change the flow of the application to make web services feasible.

Development Guidelines
Developers must follow the SOAP standard and the WS-I Basic Profile standard. For implementations that require secure methods, use the Web Services Security (WS-Security) specification for passing username and password. Both of these technologies are discussed in the Introduction of this document.
For implementations using Java, the standard development toolkit is the Sun's Web Services Developer Kit version 1.5. This kit provides a XML DOM parser, a XML SAX parser, JAXM, JAXB, and JAX-RPC. A standard C++ toolkit is gSOAP and xerces.
A developer is responsible for creating test clients or a real application that uses the web service and testing plans for QA to use. These test clients should call the web service and can be web-based or client-based. The web-based testing clients should go into internal-apps-dvlp and internal-apps-qa.
A developer must create the following documentation about the web service: a short description of the web-service, a detailed document giving the web service message format or RPC parameters, and a service level agreement detailing the availability of the web service and how often it can be called. The

service-level agreement should detail the notification time frame for changes in the web service interface to the clients.

Sun's Web Services Developer Kit version 1.5

The Sun Web Services Developer Kit version 1.5 is the standard Java framework for developing web services. The package contains JAX-RPC, JAXB, and XML parsing.

Starting with a WDSL file, a developer generates the server code using wscompile. Please make sure you generate the model file and the features include wsi. Once the war file is created, wsdeploy should be run on it to create a new war file. This command changes the web.xml file to contain a web service dispatcher.

SOAP Messaging and JAXB

1. The developer should publish a XML schema describing the request format. Optionally, an application can validate the incoming request against the schema. For a XML schema, the file should follow this layout:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:annotation>
      <xsd:documentation>
         [package].[method] - [request or response]
         [description of request or response]

         Copyright 2003 National Association of Insurance Commissioners
      </xsd:documentation
   </xsd:annotation>

   <xsd:element name=[name]>
      <xsd:annotation>
         <xsd:documentation>
            [description]
         </xsd:documentation>
      </xsd:annotation>
   </xsd:element>
   ...
</xsd:schema>
```

   Adding the annotation tags to the element tags provides a data dictionary for the XML and allows for a program to generate other documentation about the web service.

21. The developer should publish a XML schema for the XML response if the tags are not dynamic. The schema should have the annotation tags like in the previous item.
22. The XML schemas should be versioned.
23. The Java classes implementing the web service must be in a package.
24. Prefer SAX parsing to DOM parsing. SAX parsing is faster than DOM parsing.
25. The SOAP Message body contents should not contain extra carriage return or line feed characters.

26. The application must not use attachments on SOAP Messages. The Sun implementation of attachments is incompatible with the Microsoft version. The Sun developer kit implements attachments using MIME/multipart and Microsoft uses DIME.
27. Be careful of using JAXB with dates. Jaxb generates dates in the format CCYY-MM-DD+HH:MI where +HH:MM is the timezone offset from GMT. The offset is optional in the XML standard datatype specifications. I have not found a way to remove this offset value when marshalling the Java object.
28. JAXB creates the line, <?xml version="1.0" encoding="UTF-8" standalone="yes"?>, at the beginning of the marshalled string. It must be removed before placing this string into the SOAP Body contents.

Deployment of the Web Service

If it is an internal application, put those services into internal-apps which has a DVLP. For external use, NIPR should put it into the license registry website. In the near future, the internet app servers will host the web services. The web-based testing clients should go into internal-apps-dvlp and internal-apps-qa. Once in QA, the web service will be deployed onto the web application server. Please contact an architect for the instance.

Schemas and Change Management

This section deals with schema and WSDL files and trying to minimize client issues when a web service interface must change. The goal of the developer is to provide zero to small changes in client code when a developer changes to a web service. A web service must define its interface in a schema or WSDL file. The directory structure and versioning of the files is presented in a later document. For message-based web services, the service developer should add optional tags after pre-existing tags. If the client developer wrote a SAX parser that ignores extra data in tags, the code does not change. Another option is to version the new interface by changing the top-level tag with a version number. The code still supports the old and the new interface.

Example Code

An example of a Company data retrieval service and client is on hpa5web01 with the source here. The client in this example is an Excel spreadsheet, testsoap.xls, that returns the company data for a cocode using the CompanyService.GetServlet service. The GetServlet class provides an example of implementing the service as a Java Servlet using a SAX parser. The GetJaxb is an example of using JAXB to create the SOAP response. GetRpcImpl is the implementation class for the GetRpc method that is created using a WSDL file. To build the examples, use the command, ant build. The NAIC.base libraries is required along with the ini file.

NTA

NATIONAL TECHNICAL ARCHITECTURE

National Technical Architecture

Proposed Updates

Prepared 02/27/2007

Introduction

The National Technical Architecture (NTA) is a set of open standards that the NAIC will use in its architecture, and will also be used by all entities that are required or choose to interface with NAIC systems. At the heart of NTA is the concept that the architecture will not impose any requirements on what tools or methods the external entities use to interface with the NAIC assuming they follow the group of open standards outlined in the NTA.

Over time standards are enhanced, or new standards are introduced that may need to be included in the NTA. This document proposes that several open standards be added to the NTA.

Proposed Updates to NTA

NFS (Network File System)

A standard for accessing files on a remote computer appearing as a local volume.

FCIP (Fibre Channel over IP)

A network storage technology that combines the features of Fibre Channel and the Internet Protocol (IP) to connect distributed SANs (Storage Area Network) over large distances.

NDMP (Network Data Management Protocol)

Provides network-based backup of NAS (Network Attached Devices) devices.

JaxB (Java Architecture for XML Binding.)

JAXB is a Java technology that enables you to generate Java classes from XML schemas. It provides a fast and convenient way to bind an XML schema to a representation in Java code, making it easy for Java developers to incorporate XML data and processing functions in Java applications.

UDDI (Universal Description, Discovery and Integration)

UDDI is a XML-based protocol that provides a distributed directory that enables businesses to list themselves on the Internet and discover other services. Similar to a telephone number, businesses can list themselves by name, product, location, or the Web services they offer.

XML Schema

XML Schema is a language for describing the structure and constraining the contents of XML documents.

BPEL (Business Process Execution Language)

An XML-based language for the formal specification of business processes and business interaction protocols. BPEL extends the Web Services interaction model and enables it to support business transactions. It is the result of a cross-company initiative between IBM, BEA and Microsoft to develop a universally supported process-related language.

NTA

NATIONAL TECHNICAL ARCHITECTURE

National Technical Architecture

Standards Implementation Case Study

Prepared 02/09/2007

Introduction

The National Technical Architecture (NTA) is a set of open standards that the NAIC will use in its architecture, and will also be used by all entities that are required or choose to interface with NAIC systems. At the heart of NTA is the concept that the architecture will not impose any requirements on what tools or methods the external entities use to interface with the NAIC assuming they follow the group of open standards outlined in the NTA.

Saying this though, the real effort is either putting together a system that uses these open standards or ensuring your existing system does. The purpose of this document is to show how the NAIC in Kansas City is applying these Open Standards. Hopefully this will assist other entities if they need a guide for implementing these standards.

In this document you will see specific vendors mentioned that the NAIC has used, but remember you are in no way limited to these vendors, and the NAIC has no intention of trying to persuade external entities to use a particular vendor. This is solely to give an example that external entities can use for information in getting their own systems ready to interface with the NAIC. Also note that in some areas where vendors are mentioned the final selection of the vendor has not been made, but for the purpose of this document one of the finalists will be shown.

A Case Study

To show the NAIC's use of NTA, we will use a project currently underway and how that project uses NTA. The project is State Producer Licensing Reengineering (SPLR). This is a very large project that will cover many of the open standards outlined in the NTA. Although some of the components in this implementation guide are complete, the majority of them will be the design phase and not actually in production.

To understand what is talked about in this document, you will need some understanding of what SPLR actually is, so here is a brief description:

> The producer licensing system encompasses the licensing and market data as well as the software applications and reports based on this data. States submit licensing and regulatory action information to the NAIC that is aggregated in the State Producer Licensing Database for use by state insurance regulators. A copy that is compliant with the Fair Credit Reporting Act is provided to the NIPR for use by the industry. The system generates several types of reports for state and industry users. The system also facilitates transactions initiated by industry customers for transmission and processing through to the state insurance departments such as appointment/terminations, non-resident licensing and renewals, and resident licensing and renewals.

Architecture Overview

The following drawings show the main components of State Producer Licensing Reengineering project. On each drawing you will see the open standards notated with an
 Click on this link, and it will take you to NAIC specific detail on implementing the standard. The template will cover:

1. *Description*
2. *Editor*
3. *Container*
4. *NAIC Developer Guidelines*

Descriptions of each application component

Gateway (Component #1)
Gateway provides the ability to allow Industry and States to interact electronically for the processing of the following transaction types: appointments, terminations, non-resident licensing, resident licensing, and appointment renewals. In this context, the Gateway performs the following business activities:

- Processing of Industry files which contain the above transaction types
- Transaction processing and validation based on State specific business rules
- Electronic Fund Transfers (EFT) for those Industry members and States that have this capability
- Billing system updates for invoicing and EFT tracking
- Notification to States of Industry requests and processing Approvals and Denials from the States
- Provides notification to Industry regarding the status of submitted transactions

Reporting Framework (Component #2)
This framework provides a set of standards and guidelines for creating all reports in the system. It also provides an IDE with reusable common classes.

Business Rules Framework (Component #3)
This is a framework for representing state business rules in such a way that the applications can alter their functionality by reacting to changes in the rules metadata instead of requiring changes to the actual program code.

This allows the separation of the state business rules from the application logic.

Common Arch (Component #4)
Common Architecture (CommArch) is the middleware that works as an interface between state and PDB. CommArch receives a file that is a snapshot of the state licensing system tables and loads it to a staging database that mirrors the state's licensing system. In a nutshell, CommArch does the following:

- Initial set up of the state in PDB. All data elements in the state database are mapped to PDB data elements.
- All transactions are created to establish the entities and their underlying information.
- CommArch keeps a copy of the transactions submitted in the database. It also keeps a copy of the snapshot of the record from the initial load.
- On a daily basis, CommArch goes through a change management by using the previous day's snapshot versus the current snapshot received and detects the changes.
- CommArch submits the transactions to the PDB load process.

Load (Component #5)
The PDB load process accepts and processes transaction files from the states (through Common Arch). The transactions are in a fixed-length format defined by the PDB Layout document. The following transactions are available for inserting, updating and deleting data:

1. Biographic – name, birth date
2. Demographic – phone, address, email address
3. License
4. Appointment
5. Regulatory Actions (RIRS)
6. Special Activities (SAD)
7. Closed Complaints (CDS)
8. Entity Relationships
9. State Contacts

The system also includes support applications.

1. NAU – NPN Assignment Utility. This application allows a data analyst to assign a new or existing NPN to a transaction that arrived without an SSN.
2. X-Ref – This application detects data integrity errors of entity-level identifiers by comparing the data submitted from all states. Data analysts can determine the correct value for a particular identifier and instruct the PDB Loader to use the correct value when the incorrect value has been submitted by the state on a transaction.
3. Transmission Viewer – This allows state employers and data analysts to review the errors associated with incoming state transactions.
4. Transaction Utility – This allows state employers and data analysts to create and submit transactions for RIRS, SAD and CDS data.
5. NPN extract – newly created NPNs are submitted back to the state using FTP or the state process system.
6. XML translation and validation – Allows for transactions to be submitted in XML format.

Billing (Component #6)

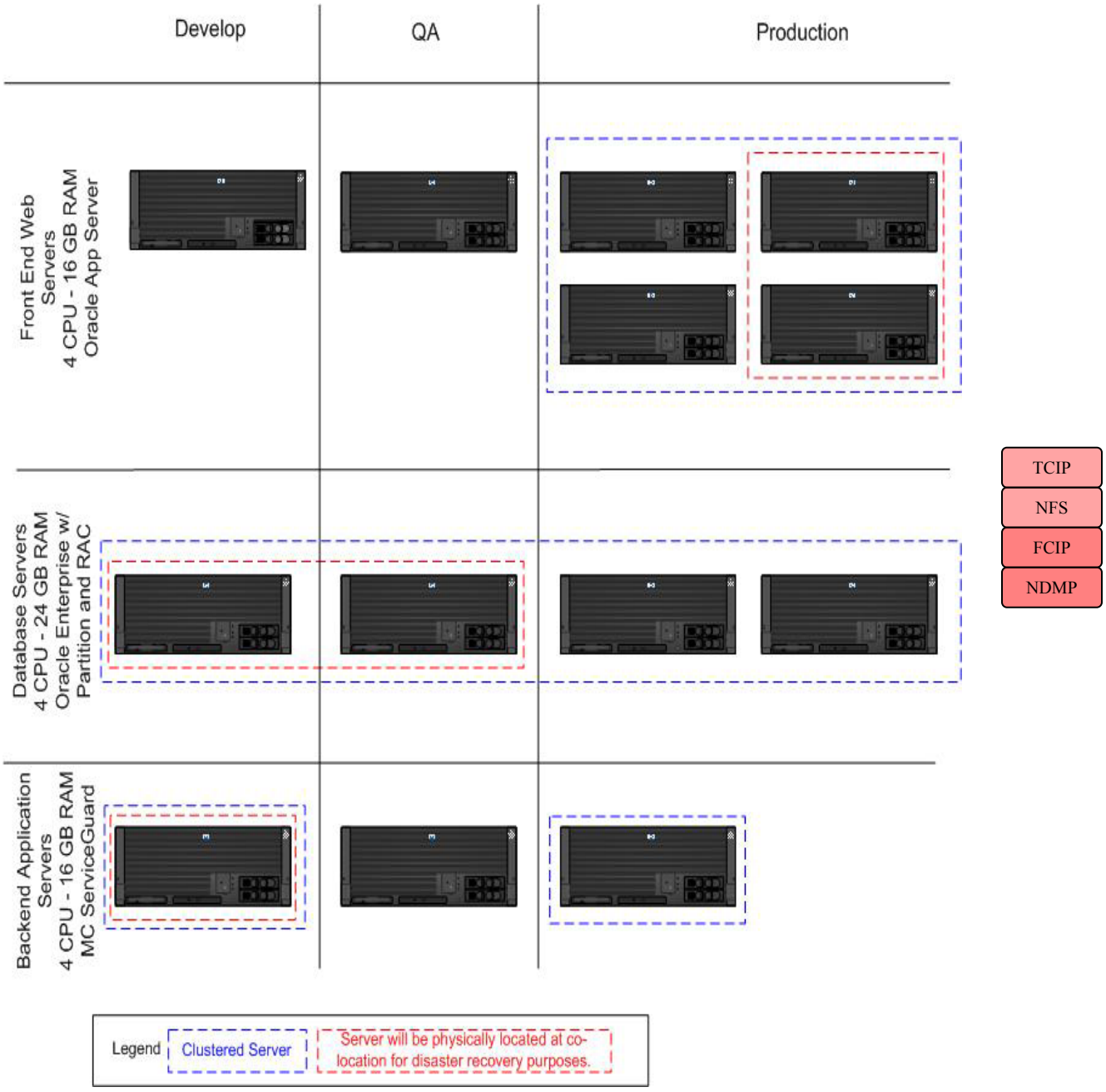
The NIPR Billing system collects, processes, calculates fees and summarizes billable items. It also is used to track revenue and volumes/usage of NIPR products. The NIPR Billing system is used to setup and store Industry customer and account information. This system sends the above information over to PeopleSoft after all fee calculations and summarizations have been completed on a weekly and monthly schedule. The application utilizes a web–based interface for maintaining customer and account information and their billable transactions. Today, a report is provided to SBS for processing royalty payments. This functionality needs to be retained.

Monitoring & Auditing (Component #7)

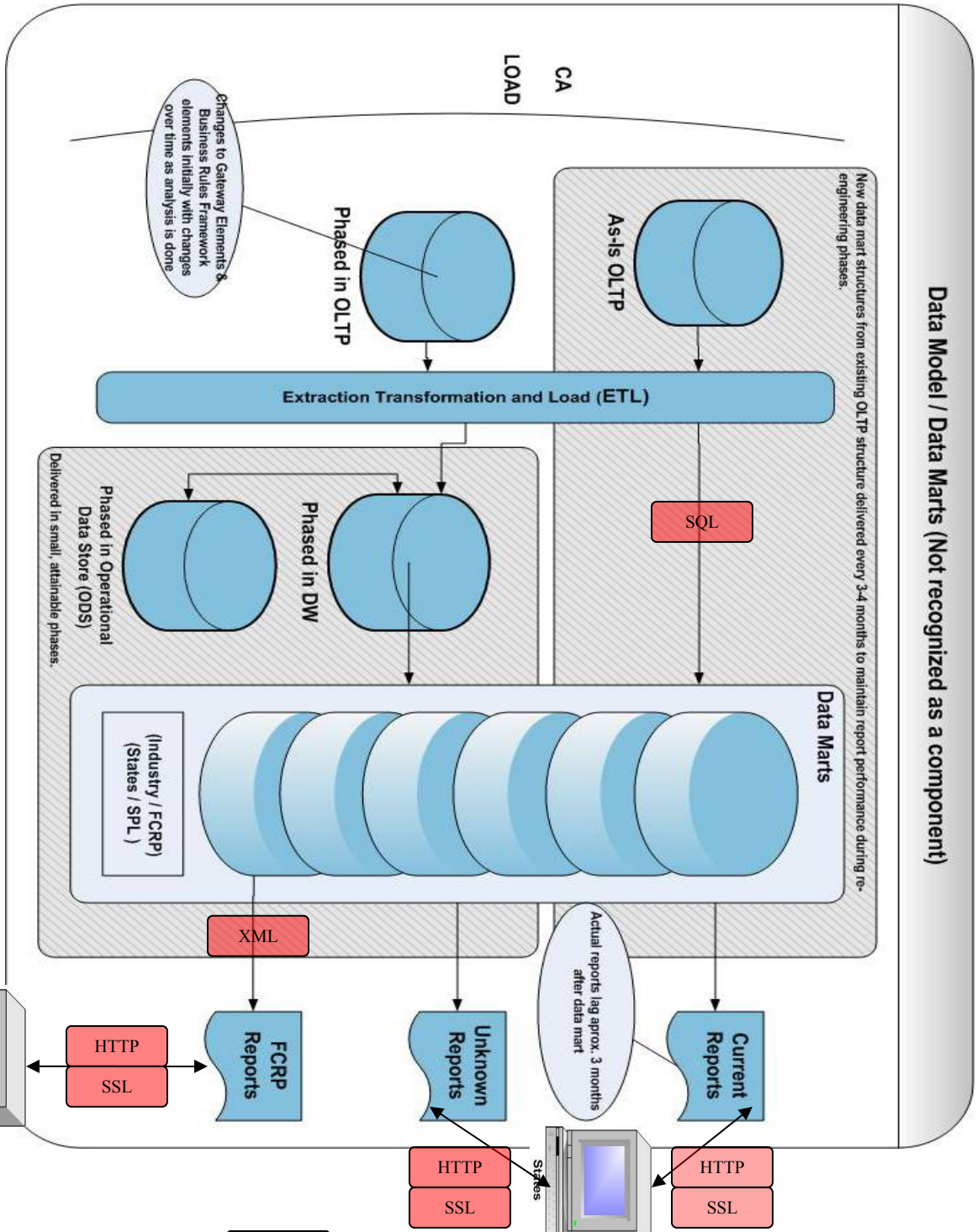
Performance Monitoring - Provides the capability to monitor the performance of the various NIPR applications (PDB Loader, Gateway, Reports, Billing, etc.). Specific measurement criteria and parameters are determined on an application-by-application basis. The measurements are performance, load and business function related. The suite of tools that are currently being designed and developed are targeted to provide this capability across the suite of NIPR developed an/or maintained applications. The deployment is currently intended for Java and C++.

Usage Tracking – Usage tracking consists of data regarding who has had access to an application at any given time. Usage tracking consists of USERID, IP Address, and date and time stamp. This is an enterprise utility that provides the capability for NAIC/NIPR systems to implement a common system usage framework. The data collected is stored within the application's native database. Periodically, system level utilities copy the usage data to a central repository for reporting and auditing.

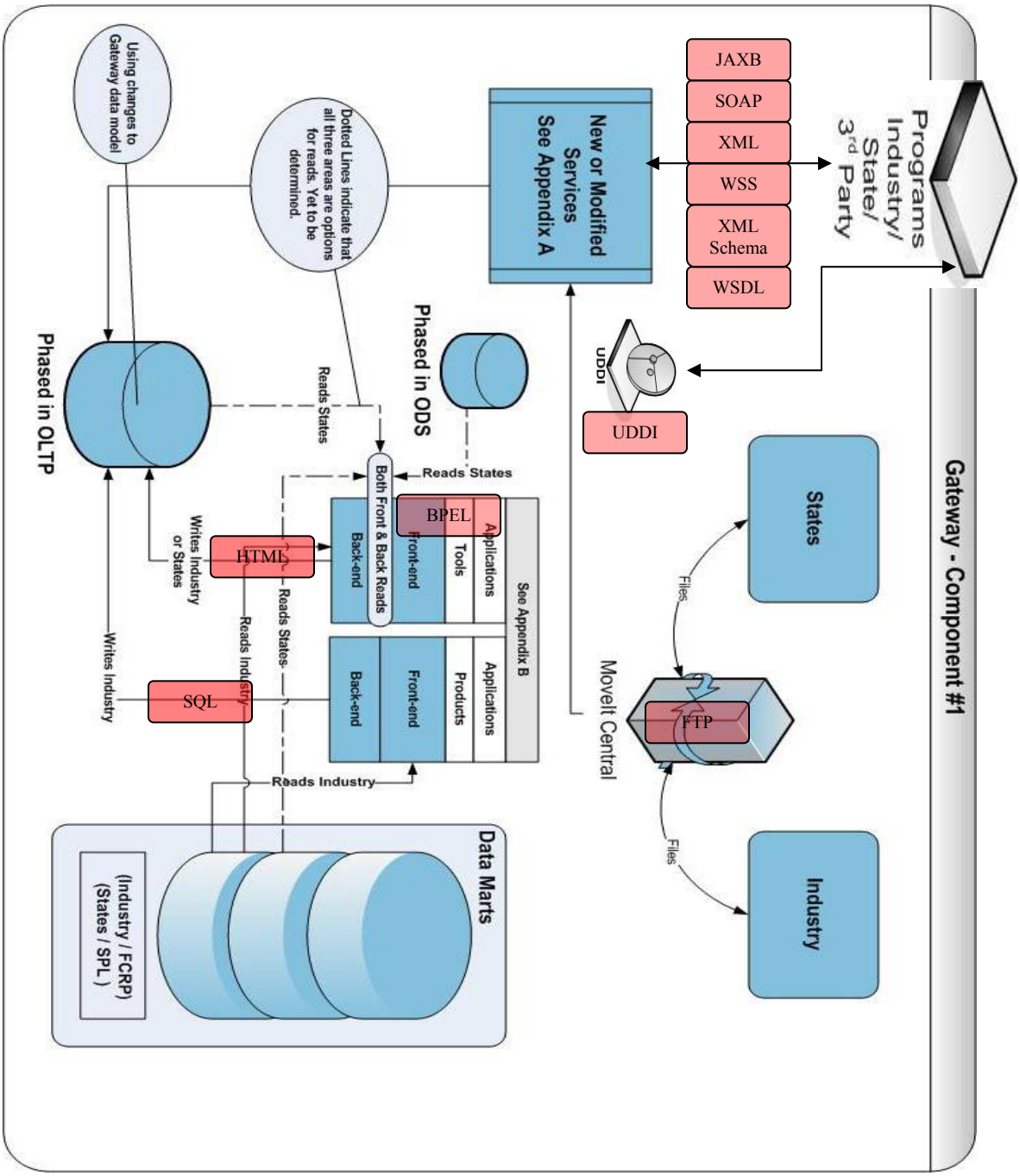
Infrastructure (Click on [ ] to see standard implementation information)



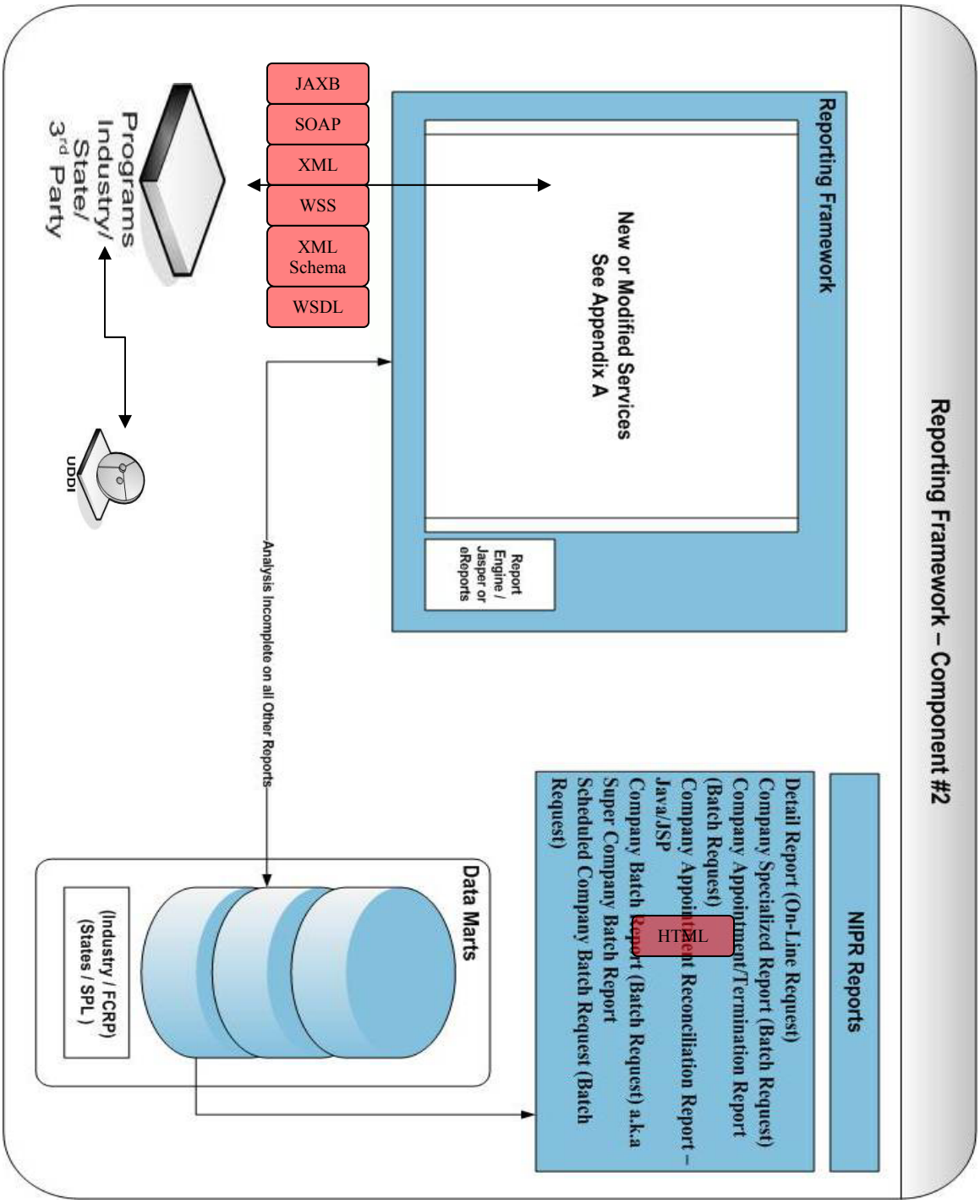
Database (Click on [ ] to see standard implementation information)



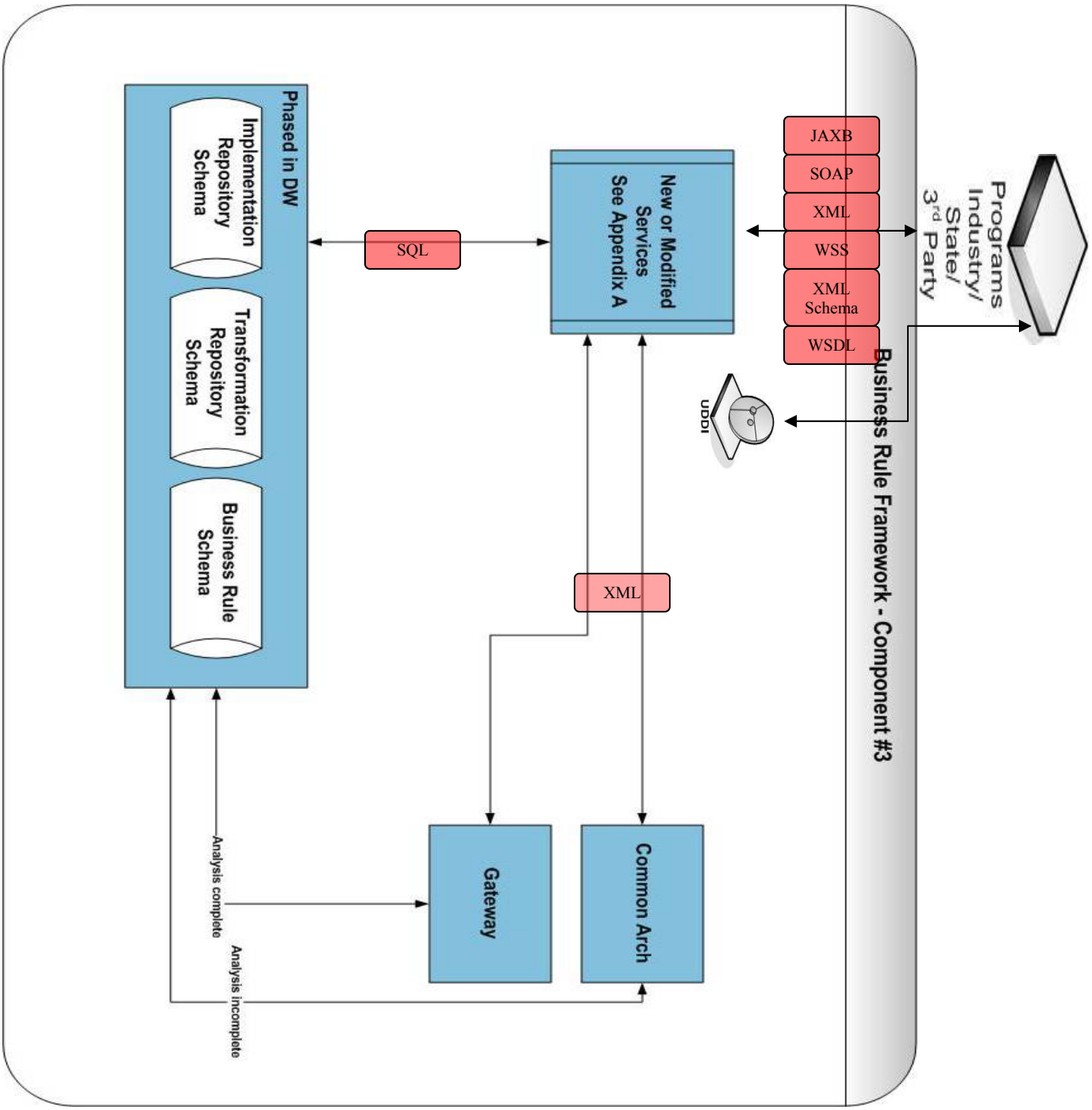
Gateway (Click on to see standard i...ation information)



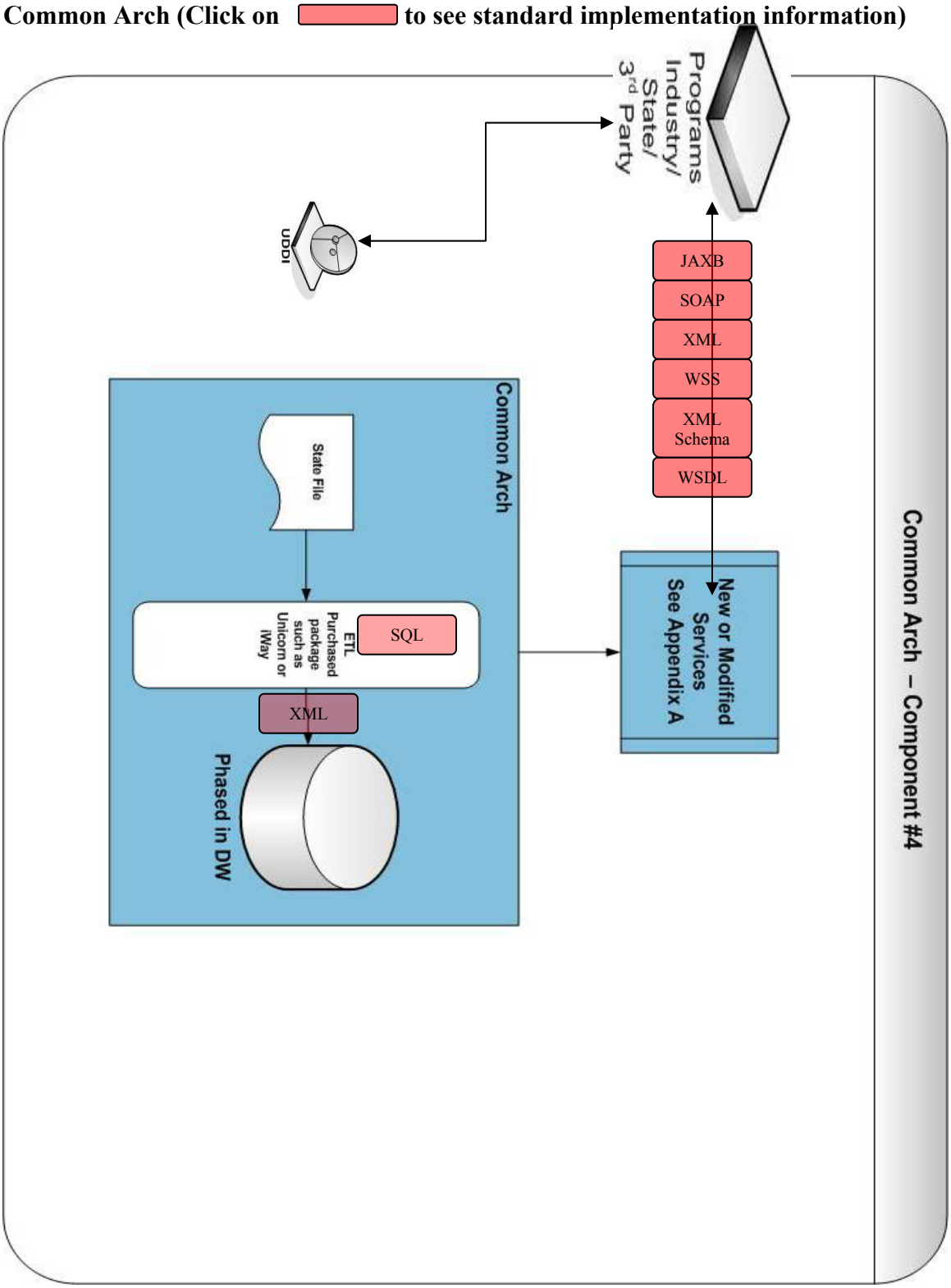
Reporting Framework (Click on [ ] to see standard implementation information)



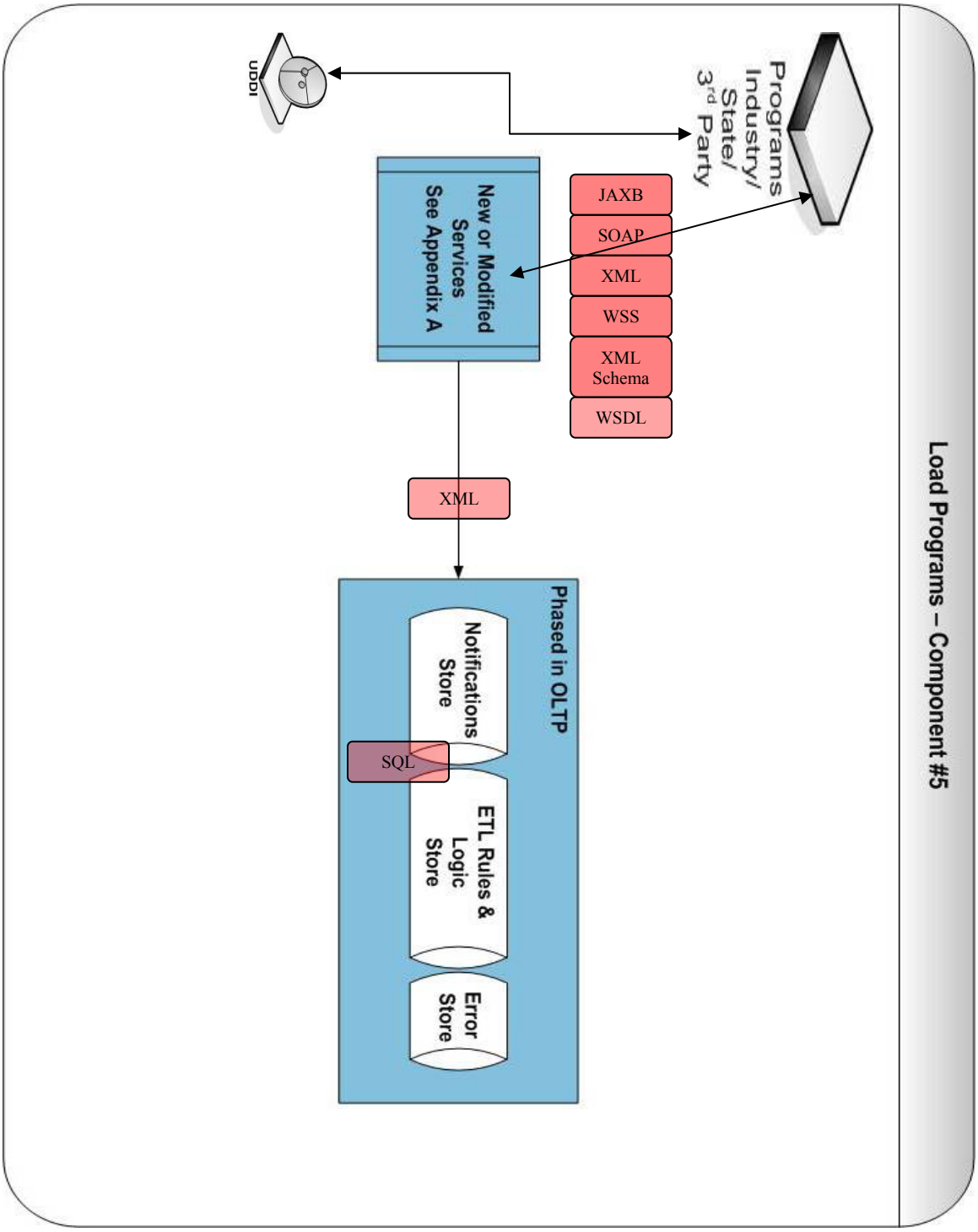
Business rules Framework (Click on to see standard implementation information)



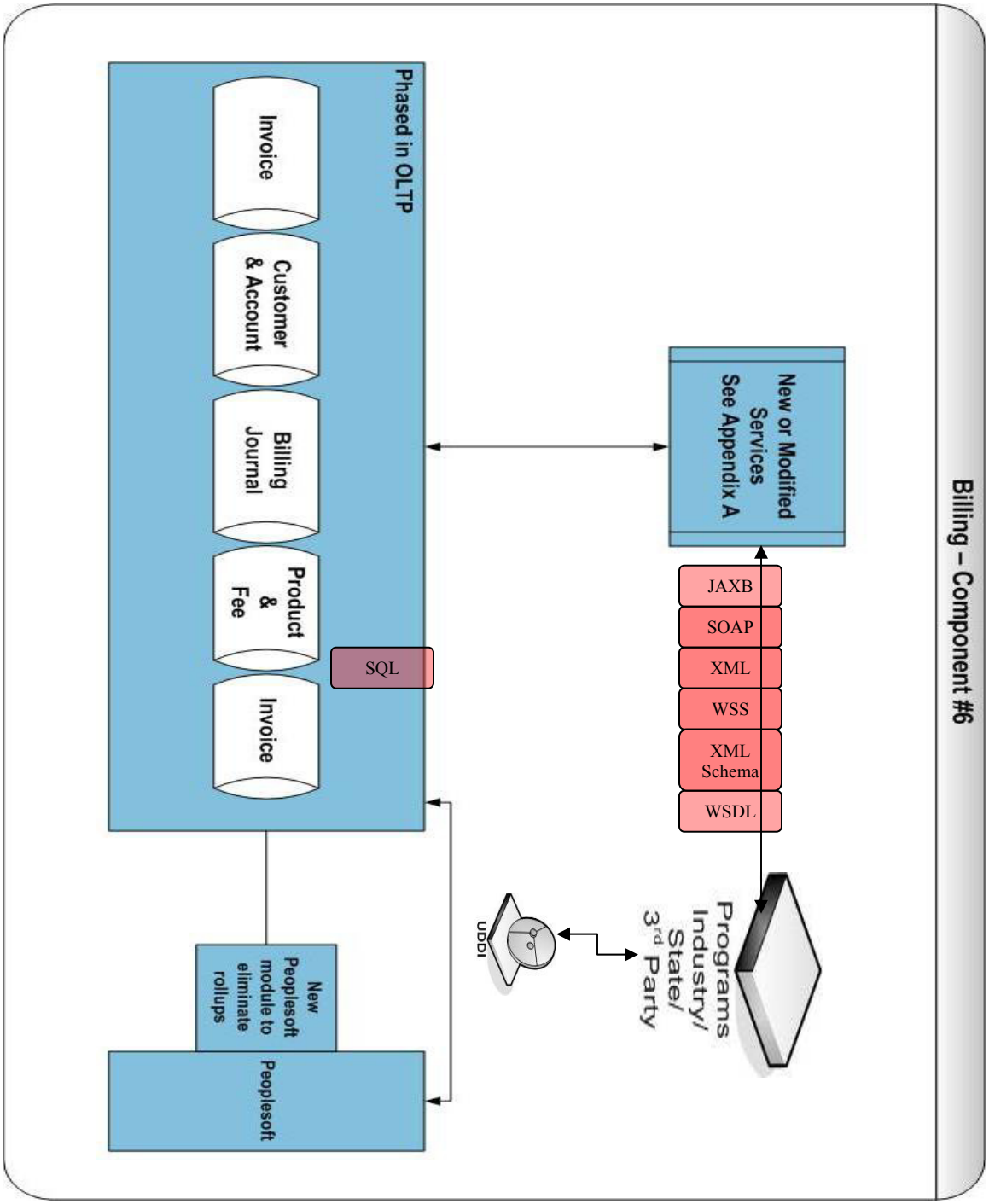
Common Arch (Click on to see standard implementation information)



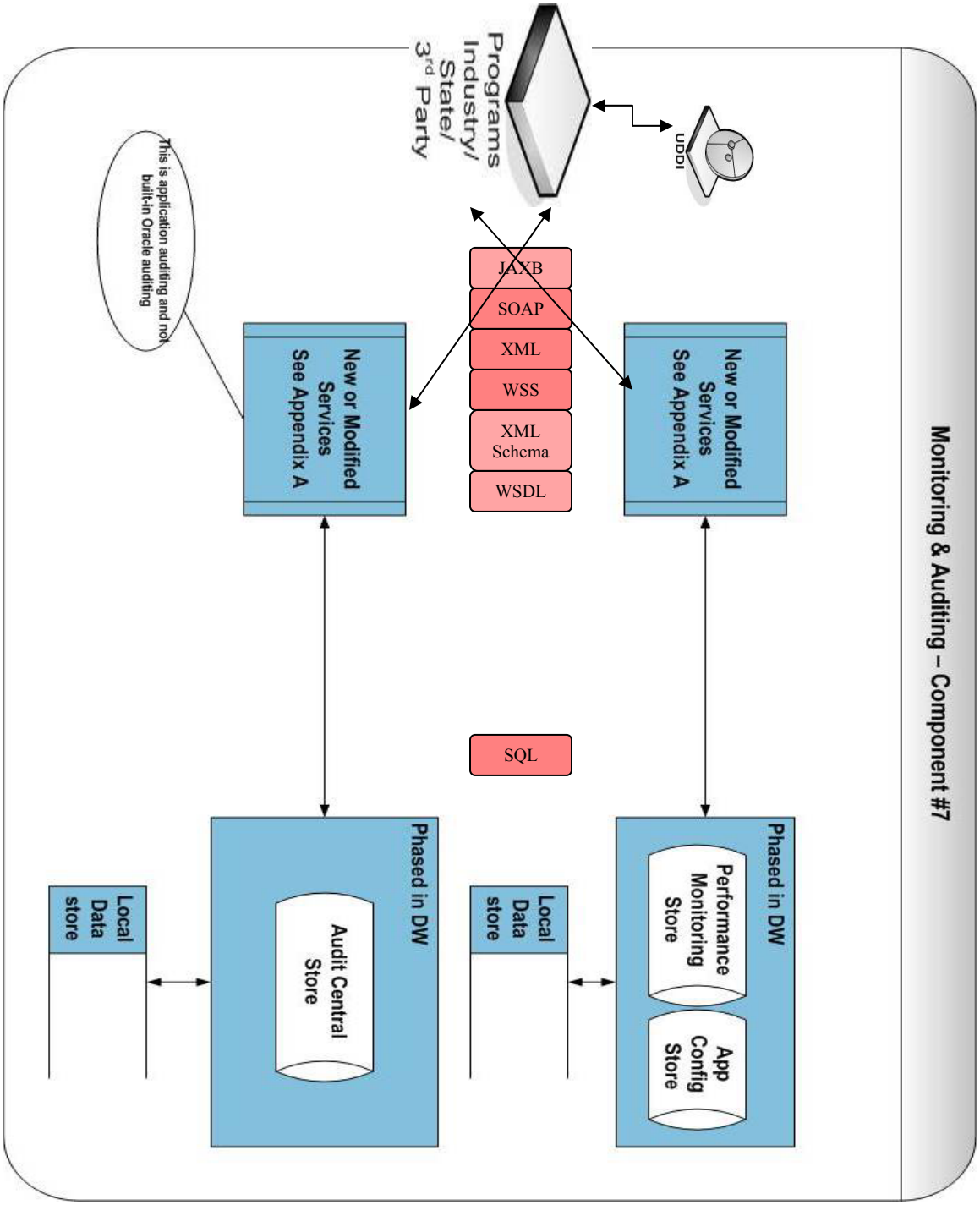
Load (Click on ▭ to see standard implementation information)



Billing (Click on [ ] to see standard implementation information)



Monitoring & Auditing (Click on to see standard implementation information)



TCPIP

Description

The Internet protocol suite is the set of communications protocols that implement the protocol stack on which the Internet runs. It is sometimes called the TCP/IP protocol suite, after the two most important protocols in it: the Transmission Control Protocol (TCP) and the Internet Protocol (IP), which were also the first two defined.

Editor

N/A

Container

Cisco

Clippings from NAIC Developer Guidelines

N/A

NFS

Description

Network File System. Standard for accessing files on a remote computer appearing as a local volume.

Editor

N/A

Container

Cisco

Clippings from NAIC Developer Guidelines

N/A

FCIP

Description

Short for *Fibre Channel over IP*, a network storage technology that combines the features of Fibre Channel and the Internet Protocol (IP) to connect distributed SANs over large distances.

Editor

N/A

Container

Cisco

Clippings from NAIC Developer Guidelines

N/A

NDMP

Description

Network Data Management Protocol. Provides network-based backup of NAS devices.

Editor

N/A

Container

Cisco

Clippings from NAIC Developer Guidelines

N/A

SQL

Description

Structured Query Language (SQL), pronounced "sequel", is a language that provides an interface to relational database systems. It was developed by IBM in the 1970s for use in System R. SQL is a de facto standard, as well as an ISO and ANSI standard.

Editor

SQL Navigator

Container

Oracle 10g

Clippings from NAIC Developer Guidelines

The NAIC uses Oracle's guidelines for this:

http://www.oracle.com/technology/sample_code/tech/pl_sql/index.html

HTTP

Description

Hyper Text Transfer Protocol (HTTP), the actual communications protocol that enables Web browsing.

Editor

N/A

Container

Cisco

Clippings from NAIC Developer Guidelines

N/A

SSL

Description

The Secure Sockets Layer (SSL) is a commonly-used protocol for managing the security of a message transmission on the Internet.

Editor

N/A

Container

Cisco

Clippings from NAIC Developer Guidelines

N/A

JaxB

Description

Java Architecture for XML Binding. JAXB is a Java technology that enables you to generate Java classes from XML schemas. It provides a fast and convenient way to bind an XML schema to a representation in Java code, making it easy for Java developers to incorporate XML data and processing functions in Java applications.

Editor

Oracle JDeveloper

Container

Sun's Web Service Developer Kit / Oracle Application Server

Clippings from NAIC Developer Guidelines

• • •

SOAP Messaging and JAXB

1. The developer should publish a XML schema describing the request format. Optionally, an application can validate the incoming request against the schema. For a XML schema, the file should follow this layout:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <xsd:annotation>
                <xsd:documentation>
                        [package].[method] - [request or response]
                        [description of request or response]

                        Copyright 2003 National Association of Insurance Commissioners
                </xsd:documentation
        </xsd:annotation>

        <xsd:element name=[name]>
                <xsd:annotation>
                        <xsd:documentation>
                                [description]
                        </xsd:documentation>
                </xsd:annotation>
        </xsd:element>
        ...
</xsd:schema>
```

   Adding the annotation tags to the element tags provides a data dictionary for the XML and allows for a program to generate other documentation about the web service.

2. The developer should publish a XML schema for the XML response if the tags are not dynamic. The schema should have the annotation tags like in the previous item.
3. The XML schemas should be versioned.
4. The Java classes implementing the web service must be in a package.
5. Prefer SAX parsing to DOM parsing. SAX parsing is faster than DOM parsing.
6. The SOAP Message body contents should not contain extra carriage return or line feed characters.

7. The application must not use attachments on SOAP Messages. The Sun implementation of attachments is incompatible with the Microsoft version. The Sun developer kit implements attachments using MIME/multipart and Microsoft uses DIME.
8. Be careful of using JAXB with dates. Jaxb generates dates in the format CCYY-MM-DD+HH:MI where +HH:MM is the timezone offset from GMT. The offset is optional in the XML standard datatype specifications. I have not found a way to remove this offset value when marshalling the Java object.
9. JAXB creates the line, <?xml version="1.0" encoding="UTF-8" standalone="yes"?>, at the beginning of the marshalled string. It must be removed before placing this string into the SOAP Body contents.

• • •

SOAP

Description

A lightweight XML based protocol used for invoking web services and exchanging structured data and type information on the Web.

Editor

Oracle JDeveloper

Container

Sun's Web Service Developer Kit / Oracle Application Server

Clippings from NAIC Developer Guidelines

The SOAP standard version 1.2 contains the basic format specification of the data sent and received by web services called SOAP Messaging. Amending the SOAP standard is WS-I Basic Profile. WS-I promotes of Web Services across operating systems, platforms, and languages. It is a large industry organization that includes Sun, IBM, Oracle, and Microsoft. The basic profile places more requirements than the SOAP standard on messages. The data sent between the client and the server are called messages in the standard and follow an XML format. The following example shows the required tags for a message:

```
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
  <soap-env:Body>
  ....
  </soap-env:Body>
</soap-env:Envelope>
```

The Envelope tag is the root tag of all SOAP messages and it is defined in the http://schemas.xmlsoap.org/soap/envelope/ namespace which is given an alias of soap-env. The next tag, Body, is the container for any user data in the message.

As with any web-based technology, communication and server errors can occur. For these types of errors, SOAP defines the same behavior as web servers currently perform. This behavior is to return an error code in the header with supporting information. These error codes are standard and SOAP implementations know how to handle them.

In addition to server errors, the SOAP standard addresses defines the communication of program errors to a client. This example shows the required tags for a program error:

```
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope">
  <S:Body>
    <S:Fault>
      <faultcode>S:Server</faultcode>
      <faultstring>Server Error</faultstring>
      <detail> .... </detail>
    </S:Fault>
  </S:Body>
</S:Envelope>
```

For this example, the same namespace, http://www.w3.org/2001/12/soap-envelope, as above is aliased by S. Notice that Envelope and Body are the same as the above example. The faultcode is the application supplied error code and is required. The faultstring is the explanation of the error and is required. The optional faultactor is an intermediate recipient to whom a message was routed. Lastly, the detail tag gives the application specific error information related to the Body element processing and is required if processing the message body.

In addition to the standard Envelope and Body tags, SOAP allows a Header tag. The Web Services Security (WS-Security) specification for passing username and password utilizes the Header tag to send this information. The following example shows the security header.

```
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope">
  <S:Header>
    <se:Security xmlns:se="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-secext-1.0.xsd">
      <se:UsernameToken>
        <se:Username>swa</se:Username>
        <se:Password>swapw</se:Password>
      </se:UsernameToken>
    </se:Security>
  </S:Header>
  <S:Body>
   ....
  </S:Body>
</S:Envelope>
```

The Security tag is the container for all the authentication tags. It can contain other security authentication tags than given in the example above. UsernameToken contains the Username and Password tags. An application uses the data in these two tags for authentication. This message is sent over SSL between the server and the client.

For SOAP-Messaging, the developer should publish a XML schema describing the SOAP body XML content. Optionally, an application can validate the incoming request against the schema. For a XML schema, the file should follow this layout:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```
  <xsd:annotation>
    <xsd:documentation>
      [package].[method] - [request or response]
      [description of request or response]

      Copyright 2004 National Association of Insurance Commissioners
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name=[name]>
    <xsd:annotation>
      <xsd:documentation>
        [description]
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  ...
</xsd:schema>
```

Adding the annotation tags to the element tags provides a data dictionary for the XML and allows for a program to generate other documentation about the web service.

• • •

HTML

Description

HTML, short for *Hypertext Markup Language*, is the predominant markup language for the creation of web pages.

Editor

Oracle JDeveloper

Container

SunOne Web Server

Clippings from NAIC Developer Guidelines

• • •

1) HyperText Markup Language(HTML) Authoring Standards
    i) Required Tags
    ii) Title
    iii) Images
    iv) Frames

HyperText Markup Language(HTML) Authoring Standards

Required Tags

1. You should make the first line of your HTML document a DOCTYPE declaration, like this:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.1 Transitional//EN">
<HTML>
<HEAD>
<TITLE>
```

2. You should use the following tag in the <HEAD> section for the character set

<META HTTP-EQUIV="Content-Type" content="text/html;charset=ISO-8859-1">

3. You should use the following tag in the <HEAD> section for the defining the default script language.

<META HTTP-EQUIV="Content-Script-Type" content="text/javascript">

4. You should use the HTML tags <html> and </html>
5. You should use the HEAD tags <head> and </head>
6. You should use the TITLE tags <title> and </title>
7. You should use the BODY tags <body> and </body>

• • •

UDDI

Description

UDDI is a XML-based protocol that provides a distributed directory that enables businesses to list themselves on the Internet and discover other services. Similar to a telephone number, businesses can list themselves by name, product, location, or the Web services they offer.

Editor

Oracle Service Registry Component

Container

Oracle Application Server

Clippings from NAIC Developer Guidelines

Yet to be developed for “true” UDDI, but for the current homegrown NAIC Web Service Registry it is as follows:

XML

Description

XML (Extensible Markup Language) is a W3C initiative that allows information and services to be encoded with meaningful structure and semantics that computers and humans can understand. XML is great for information exchange, and can easily be extended to include user-specified and industry-specified tags.

Editor

Oracle JDeveloper

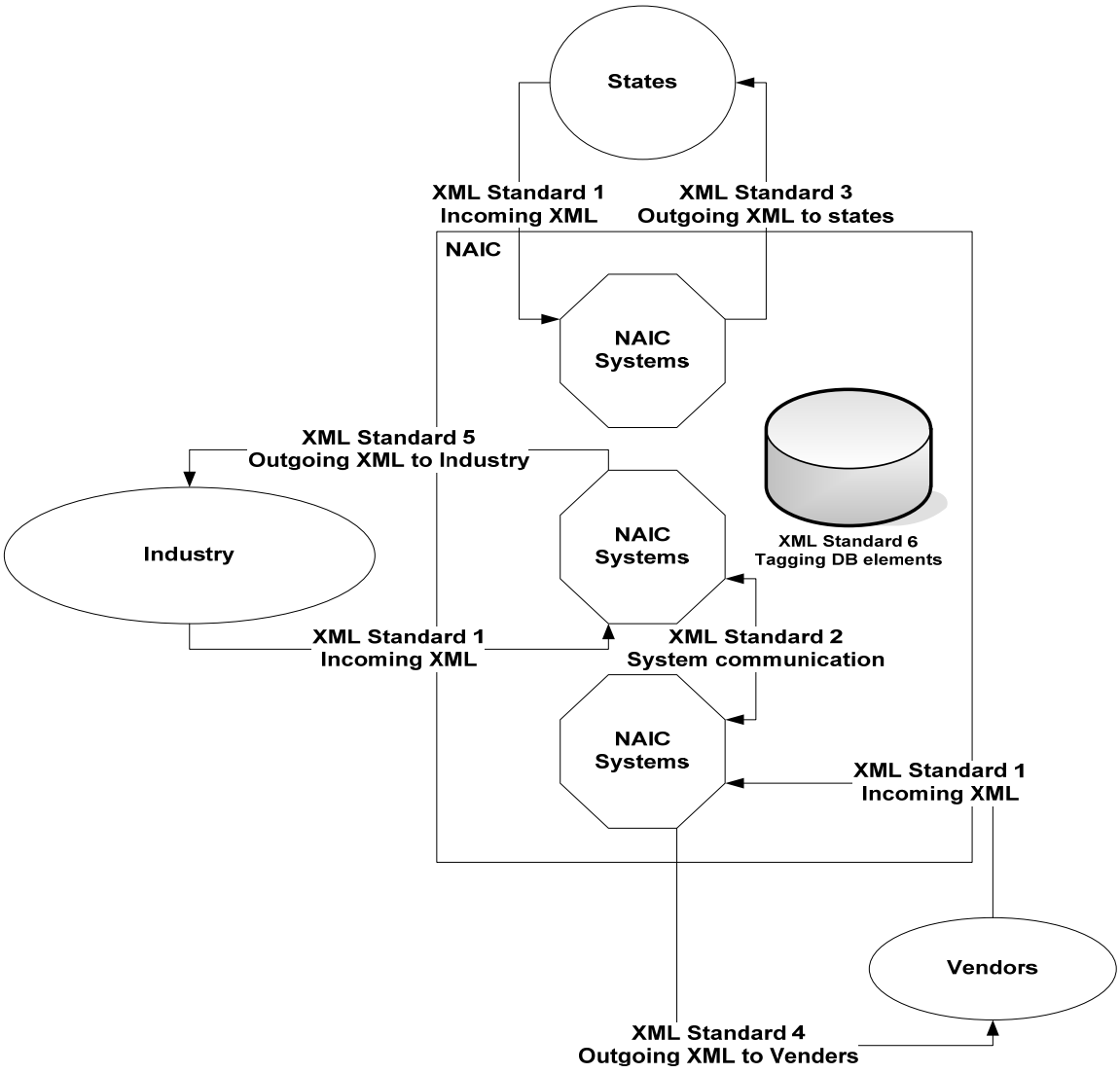
Container

Oracle Application Server

Clippings from NAIC Developer Guidelines

• • •

The NAIC is working on or plans to work on the standard implementation of XML in 6 areas outlined in the following drawing:



• • •

WSS

Description

WSS is a set of specifications that are becoming the standard for secure Web services communication. These specifications describe how to operate security concepts, including message integrity, message confidentiality and message authentication in an XML-based communication.

Editor

Oracle JDeveloper

Container

J2EE

Clippings from NAIC Developer Guidelines

• • •

In addition to the standard Envelope and Body tags, SOAP allows a Header tag. The Web Services Security (WS-Security) specification for passing username and password utilizes the Header tag to send this information. The following example shows the security header.

```
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope">
  <S:Header>
        <se:Security xmlns:se="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wsswssecurity-secext-1.0.xsd">
            <se:UsernameToken>
                <se:Username>swa</se:Username>
                <se:Password>swapw</se:Password>
            </se:UsernameToken>
        </se:Security>
  </S:Header>
  <S:Body>
        ....
  </S:Body>
</S:Envelope>
```

The Security tag is the container for all the authentication tags. It can contain other security authentication tags than given in the example above. UsernameToken contains the Username and Password tags. An application uses the data in these two tags for authentication. This message is sent over SSL between the server and the client.

For SOAP-Messaging, the developer should publish a XML schema describing the SOAP body XML content. Optionally, an application can validate the incoming request against the schema. For a XML schema, the file should follow this layout:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:annotation>
      <xsd:documentation>
         [package].[method] - [request or response]
         [description of request or response]

         Copyright 2006 National Association of Insurance Commissioners
      </xsd:documentation>
   </xsd:annotation>

   <xsd:element name=[name]>
      <xsd:annotation>
         <xsd:documentation>
            [description]
         </xsd:documentation>
      </xsd:annotation>
   </xsd:element>
   ...
</xsd:schema>
```

Adding the annotation tags to the element tags provides a data dictionary for the XML and allows for a program to generate other documentation about the

• • •

WSDL

Description

WSDL or Web Services Description Language is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services).

Editor

Oracle JDeveloper

Container

Sun's Web Service Developer's Kit / Oracle Application Server

Clippings from NAIC Developer Guidelines

• • •

SOAP WSDL Binding Styles

WSDL supports both messaging and RPC using a binding style. The standard allows four styles, RPC/encoded, RPC/literal, Document/Encoded, and Document/Literal. There is also a convention for Document/Literal called wrapped Document/Literal. RPC/encoded is the SOAP RPC method discussed above. It follows the convention of the method name as the root tag and parameters as children. Each parameter has the xsi:type attribute set to the SOAP datatype. RPC/literal follows the same convention of RPC/encoded except the xsi:type attribute is not in the message body of the data. Document/Encoded is a binding style that is not used and might be removed from the standard. Document/Literal allows the WSDL schema to use any valid XML schema to describe the request, response, and error messages. Wrapped Document/Literal is just the same as Document/Literal except the root tag for the message is the name of the message. A good article describing WSDL binding is at http://www-106.ibm.com/developerworks/webservices/library/ws-whichwsdl/.

• • •

XML Schema

Description

XML Schema is a language for describing the structure and constraining the contents of XML documents.

Editor

Oracle JDeveloper

Container

Sun's Web Service Developer's Kit / Oracle Application Server

Clippings from NAIC Developer Guidelines

• • •

This section deals with schema and WSDL files and trying to minimize client issues when a web service interface must change. The goal of the developer is to provide zero to small changes in client code when a developer changes to a web service. A web service must define its interface in a schema or WSDL file. The directory structure and versioning of the files is presented in a later document. For message-based web services, the service developer should add optional tags after pre-existing tags. If the client developer wrote a SAX parser that ignores extra data in tags, the code does not change. Another option is to version the new interface by changing the top level tag with a version number. The code still supports the old and the new interface.

• • •

BPEL

Description

An XML-based language for the formal specification of business processes and business interaction protocols. BPEL extends the Web Services interaction model and enables it to support business transactions. It is the result of a cross-company initiative between IBM, BEA and Microsoft to develop a universally supported process-related language.

Editor

Oracle BPEL Process Manager

Container

Oracle Application Server

Clippings from NAIC Developer Guidelines

Yet to be developed

FTP

Description

A communications protocol governing the transfer of files from one computer to another over a network.

Editor

N/A

Container

MoveIt

Clippings from NAIC Developer Guidelines

• • •

B2. What commands does XFER support?
The XFER program supports the following commands.

about	exit	ls	pass	rmdir
cd	get	mdelete	prompt	user
close	help	mget	put	
debug	lcd	mkdir	pwd	

delete	ldir	mput	quit
dir	lls	open	rename

More help is available by typing Help All at the XFER prompt. A XFER User Manual is included in the XFER file downloads. The user manual lists the system requirements, installation, command line usage, commands and examples.

C. API

C1. Where can I find more information on the MOVEit API?

The MOVEit API file downloads are located at http://www.naic.org/moveit/. The Windows environment supports a COM component and Java. The Windows COM component is described in the DMZAPI_Win_Doc_40.zip file and code samples are included in the DMZ_API_Win_40.exe file. For other operating systems a Java API is offered.

Javadocs can be found in DMZAPI_Java_Doc_40.zip, DMZAPI_Java_Doc_40.tar.gz and example code can be found in the DMZAPI_Java_40.zip and DMZAPI_Java_40.tar.gz files.

Financial Data Web Services

NAIC financial data is currently available in XML format via the Pick A Page report, accessible from I-SITE or the NAIC Report Web Service (NRWS). The Pick A Page response XML represents one company's data for a specified page of the financial statement. See Appendix A for a sample of the current XML. A copy of the NRWS response schema can be referenced in Appendix B.

The Pick A Page XML data is generated using the current FDR data and meta data (i.e. company name, column name, column description, line number, row description). While this approach is relatively simple, taking advantage of data currently available, it does have some drawbacks. An evaluation of this approach follows:

Pros:

1. Financial tags are derived from existing FDR meta data.
2. A single XML schema is applied to every financial table. This single structure allows one client that can be used for multiple pages without modifications.
3. This web service exists today – no additional NAIC development required.
4. Users familiar with the current FDR database structure should be able to easily interpret this naming convention.

Cons:

1. The "Content" field of the service response contains a string labeled with the "CDATA" tag followed by Pick A Page's XML schema mentioned above. This approach was taken to provide one web service that accommodates multiple data formats, such as html, rtf, etc. This is not a recommended SOAP web service approach, as the data is hidden in the CDATA tag.
2. The financial data columns are not defined in the provided schema. Therefore the user must interrogate the XML document to determine what columns are being returned by the service.
3. The current schema does not include the database line code (LNCODE). (Note: It is recommended that LNCODE be used, rather than LINE_NO for queries that run from year to year.)

Recommendations:

1. Develop a new web service that eliminates the "CDATA" tag and provides pure XML.
2. In order to address con #2 defined above, research is being done to investigate the feasibility of using the ANY tag element. The ANY tag may provide a defined schema, where the ANY tag can be substituted with the column tags (i.e. the <Row> details in Appendix A) for the specified page. An alternative approach would be to create a unique schema and web service for each financial statement page.
3. The LNCODE should be added to the schema.

Appendix A – Current Pick-A-Page XML


```
<?xml version="1.0" encoding="UTF-8" ?>
- <Report type="PickAPage" cocode="19232" year="2005" table="P2005ASSETS">
  <PageTitle>2005 ASSETS PAGE - 002</PageTitle>
  <CompanyTitle>19232 - Allstate Ins Co</CompanyTitle>
- <ColumnLabels>
  - <Row>
      <LINE_NO />
      <LINE_NO_DESCRIPTION>Line</LINE_NO_DESCRIPTION>
      <ASSTS_CURR_YR>Assets Current Year</ASSTS_CURR_YR>
      <NONADMITTED_ASSTS_CURR_YR>Nonadmitted Assets Current Year</NONADMITTED_ASSTS_CURR_YR>
      <NET_ADMTD_ASSTS_CURR_YR>Net Admitted Assets Current Year</NET_ADMTD_ASSTS_CURR_YR>
      <NET_ADMTD_ASSTS_PR_YR>Net Admitted Assets Prior Year</NET_ADMTD_ASSTS_PR_YR>
    </Row>
</ColumnLabels>
- <MainTable>
  - <Row>
      <LINE_NO>01</LINE_NO>
      <LINE_NO_DESCRIPTION>Bonds</LINE_NO_DESCRIPTION>
      <ASSTS_CURR_YR>27278785418</ASSTS_CURR_YR>
      <NONADMITTED_ASSTS_CURR_YR>0</NONADMITTED_ASSTS_CURR_YR>
      <NET_ADMTD_ASSTS_CURR_YR>27278785418</NET_ADMTD_ASSTS_CURR_YR>
      <NET_ADMTD_ASSTS_PR_YR>27019631350</NET_ADMTD_ASSTS_PR_YR>
    </Row>
  - <Row>
      <LINE_NO>02.1</LINE_NO>
      <LINE_NO_DESCRIPTION>Preferred stocks (stocks)</LINE_NO_DESCRIPTION>
      <ASSTS_CURR_YR>354085767</ASSTS_CURR_YR>
      <NONADMITTED_ASSTS_CURR_YR>0</NONADMITTED_ASSTS_CURR_YR>
      <NET_ADMTD_ASSTS_CURR_YR>354085767</NET_ADMTD_ASSTS_CURR_YR>
      <NET_ADMTD_ASSTS_PR_YR>354072019</NET_ADMTD_ASSTS_PR_YR>
  </Row>
  - <Row>
      <LINE_NO>02.2</LINE_NO>
      <LINE_NO_DESCRIPTION>Common stocks (stocks)</LINE_NO_DESCRIPTION>
      <ASSTS_CURR_YR>9823355880</ASSTS_CURR_YR>
      <NONADMITTED_ASSTS_CURR_YR>1074949</NONADMITTED_ASSTS_CURR_YR>
      <NET_ADMTD_ASSTS_CURR_YR>9822280931</NET_ADMTD_ASSTS_CURR_YR>
      <NET_ADMTD_ASSTS_PR_YR>9616484538</NET_ADMTD_ASSTS_PR_YR>
  </Row>
  - <Row>
      <LINE_NO>03.1</LINE_NO>
      <LINE_NO_DESCRIPTION>First liens - mortgage loans on real estate</LINE_NO_DESCRIPTION>
      <ASSTS_CURR_YR>506754765</ASSTS_CURR_YR>
      <NONADMITTED_ASSTS_CURR_YR>0</NONADMITTED_ASSTS_CURR_YR>
      <NET_ADMTD_ASSTS_CURR_YR>506754765</NET_ADMTD_ASSTS_CURR_YR>
      <NET_ADMTD_ASSTS_PR_YR>381388003</NET_ADMTD_ASSTS_PR_YR>
  </Row>
  - <Row>
      <LINE_NO>03.2</LINE_NO>
      <LINE_NO_DESCRIPTION>Other than first liens - mortgage loans on real estate</LINE_NO_DESCRIPTION>
```



```
        <ASSTS_CURR_YR>0</ASSTS_CURR_YR>
        <NONADMITTED_ASSTS_CURR_YR>0</NONADMITTED_ASSTS_CURR_YR>
        <NET_ADMTD_ASSTS_CURR_YR>0</NET_ADMTD_ASSTS_CURR_YR>
        <NET_ADMTD_ASSTS_PR_YR>0</NET_ADMTD_ASSTS_PR_YR>
    </Row>
    - <Row>
        <LINE_NO>04.1</LINE_NO>
        <LINE_NO_DESCRIPTION>Properties occupied by the company (less $0 encumbrances) (real
        estate)</LINE_NO_DESCRIPTION>
        <ASSTS_CURR_YR>279006786</ASSTS_CURR_YR>
        <NONADMITTED_ASSTS_CURR_YR>0</NONADMITTED_ASSTS_CURR_YR>
        <NET_ADMTD_ASSTS_CURR_YR>279006786</NET_ADMTD_ASSTS_CURR_YR>
        <NET_ADMTD_ASSTS_PR_YR>326515854</NET_ADMTD_ASSTS_PR_YR>
    </Row>
....
    - <Row>
        <LINE_NO>26</LINE_NO>
        <LINE_NO_DESCRIPTION>Totals</LINE_NO_DESCRIPTION>
        <ASSTS_CURR_YR>47407587154</ASSTS_CURR_YR>
        <NONADMITTED_ASSTS_CURR_YR>2164111846</NONADMITTED_ASSTS_CURR_YR>
        <NET_ADMTD_ASSTS_CURR_YR>45243475308</NET_ADMTD_ASSTS_CURR_YR>
        <NET_ADMTD_ASSTS_PR_YR>44711745721</NET_ADMTD_ASSTS_PR_YR>
    </Row>
</MainTable>
- <WriteInTable>
    - <Row>
        <LINE_NO>0901</LINE_NO>
        <LINE_NO_DESCRIPTION>Security lending principal</LINE_NO_DESCRIPTION>
        <ASSTS_CURR_YR>19874545</ASSTS_CURR_YR>
        <NONADMITTED_ASSTS_CURR_YR>0</NONADMITTED_ASSTS_CURR_YR>
        <NET_ADMTD_ASSTS_CURR_YR>19874545</NET_ADMTD_ASSTS_CURR_YR>
        <NET_ADMTD_ASSTS_PR_YR>24121524</NET_ADMTD_ASSTS_PR_YR>
    </Row>
    - <Row>
        <LINE_NO>0902</LINE_NO>
        <LINE_NO_DESCRIPTION>Call options</LINE_NO_DESCRIPTION>
        <ASSTS_CURR_YR>1657501</ASSTS_CURR_YR>
        <NONADMITTED_ASSTS_CURR_YR>0</NONADMITTED_ASSTS_CURR_YR>
        <NET_ADMTD_ASSTS_CURR_YR>1657501</NET_ADMTD_ASSTS_CURR_YR>
        <NET_ADMTD_ASSTS_PR_YR>0</NET_ADMTD_ASSTS_PR_YR>
    </Row>
    - <Row>
        <LINE_NO>0999</LINE_NO>
        <LINE_NO_DESCRIPTION />
        <ASSTS_CURR_YR>21532046</ASSTS_CURR_YR>
        <NONADMITTED_ASSTS_CURR_YR>0</NONADMITTED_ASSTS_CURR_YR>
        <NET_ADMTD_ASSTS_CURR_YR>21532046</NET_ADMTD_ASSTS_CURR_YR>
        <NET_ADMTD_ASSTS_PR_YR>24121524</NET_ADMTD_ASSTS_PR_YR>
    </Row>
</WriteInTable>
</Report>
```


Appendix B – NRWS Response Schema

```
  <xsd:element name="Report" type="ReportType" />

  <xsd:complexType name="ReportType">
  <xsd:sequence>

     <xsd:element name="content" type="xsd:string">
        <xsd:annotation>
           <xsd:documentation>The actual report
contents.</xsd:documentation>
         </xsd:annotation>
     </xsd:element>

     <xsd:element name="content_type" type="xsd:string" minOccurs="0"
maxOccurs="1">
        <xsd:annotation>
           <xsd:documentation>The content type of the report. Example
for HTML - text/hmtl.</xsd:documentation>
         </xsd:annotation>
     </xsd:element>

  </xsd:sequence>
 </xsd:complexType>
```