The eRA CGAP Information Exchange

Functional Description and Technologies

Version 1.2

February 14, 2003

Prepared by:
Ekagra Software Technologies Ltd
Silicon Spirit
Z-Tech

Prepared for:
The National Institutes of Health
Office of Extramural Research
6705 Rockledge Drive
Bethesda, MD 20892

Revision History

Date	Version	Description of Change	Author
2/10/2003	1.1	Draft version after initial internal review	J-J Maurer
2/14/2003	1.2	Changed DUNS, CCR references	J-J Maurer

TABLE OF CONTENTS

1. Introduction

The Competitive Grant Application Process (CGAP) is part of the overall eRA project to create ***electronic*** processes for receipt of grant applications and subsequent business flow. The effort of the CGAP team will be focused on bridging the gap between the submitters and the receiver of applications to create an integrated process, based on the current technologies.

The process to be designed and developed is a system-to-system communication flow where applications are transmitted as streams of data and acknowledgement of receipt or status information is provided by NIH as an electronic system transaction. The system-to-system connections will bridge the internal and external information flows.

This document is a draft. It contains a summary of the ideas and approaches taken by the CGAP team in the first few weeks of the project. The document is written to document the vision for the project as well as the major technology choices being made. This document does not imply that NIH has reviewed or approved the ideas and standards presented.

This document is being written to obtain feedback from the entire community in the early stages of the project. Based on the feedback, the project team will modify some of the approaches, baseline the overall processes and technologies and proceed with the design and development of the system.

This document primarily addresses the scope of phase 1 of the project. Phase 1 will take most of calendar year 2003 to complete. Subsequent phases will expand on the concepts presented here. The scope of phase 1 is limited to the receipt and processing of simple competitive applications of type R01 without subprojects.

The document is organized in 4 major sections:

- The descriptions of the major issues with the receipt of e-applications
- The description of the overall solutions proposed including the technical approaches
- A detailed description of the anticipated process flow for information exchange
- Appendices describing the technologies evaluated

2. Issues with receipt of e-applications

There are several dominant characteristics of the receipt process for competitive applications that shape the design of the electronic receipts. This section describes these major aspects of the receipt process.

Deadlines and load curves

The receipt of competitive applications at NIH is scheduled in 3 major cycles per year. The applicants have to meet a deadline for submission. After the submission, the applications are referred to study sections, reviewed and scored. The applications are funded according to the score and other factors. Both during peer review and council review, the applications are evaluated against other applications in the submission pool. The fairness of the process and the practical constraints of peer review require that deadlines are preserved and met.

In the paper based process, truckloads of applications appear within a day or two of the deadlines. The applications are recorded and processed over a few weeks after the initial receipt.

In the electronic world it is anticipated that most submissions will be made in the last few hours. These major deadlines will cause a massive influx of application submissions in the last 2 to 4 hours of the submission cycle. Precedents of this behavior exist at NSF. It is unlikely that this fact of life will change in the near future.

The application transaction will contain a project plan document and potentially graphics and figures. The transaction can be quite large, much larger than for example a Tax Return submission or a Purchase Order.

The NIH is confronted with the problem of receiving up to 4000 large transactions of unpredictable size and in an unpredictable load curve in a few hours 3 to 4 times a year. Volumes are increasing every year.

This model of receipt would require a large infrastructure used only a few times a year. There would be delays and denials of service when capacity is reached.

The issue to be addressed is how to implement a receipt model that mitigates the peak volumes and controls the transmission of a large amount of information in a short amount of time.

Structured data and documents

The grant application is made up of structured data such as the face page, budgets and documents such as the research plan and the appendices.

The technology of choice to exchange information is XML due to its simplicity and wide acceptance and recommended use in the Federal government. However XML does not by itself allow for the presentation of the information to be preserved. An XML file is a text file. To reliably transport binary information such as formatted document, additional processing is needed.

Most applications for research grants will, at a minimum, include a project plan document. The XML structured data stream therefore needs to be expanded to transport also at least one or more documents. These documents are files with a binary content.

The issue to be addressed is how to transport the binary payload in the same transaction as the XML application data.

Standard transaction for coordination with E-GRANTS.

The primary focus of the E-GRANTS initiative is to present one face to the public for finding, applying for, and managing grants. The vision and goals of the E-GRANTS initiative are provided as Appendix 1: FDP and E-GRANTS principles. To fulfill its vision, E-GRANTS has defined a core set of grant application data elements as follows:

CORE = SF424 + DUNS as defined in the X12 194 transaction set

The definition of a core provides a common data collection set for all Federal grant programs. Agencies can extend the core to meet any data collection requirements beyond the SF424. These extensions can be categorized at four levels:

- Cross-Agency level, whereby multiple agencies collaborate to define common data sets
- Agency level, whereby an agency defines data sets common to all programs within the agency
- Program level, which is a data set specific to an agency program
- Funding Opportunity level, which is a data set specific to one funding opportunity

An E-GRANTS grant application package must include the core, but also can have cross-agency components and/or agency components and/or program components and/or funding opportunity components.

Over the course of several years, Federal research agencies collaborated to define a grant application standard. This standard is recognized by ANSI X12 as the 194 transaction set.

The process of defining the 194 transaction set has laid the groundwork for establishing a set of cross-agency components relevant to Federal research programs.

The Research & Related (R&R) subcommittee of the Inter-Agency Electronic Grants Committee (IAEGC) has defined a set of cross-agency research components for the grant application. The R&R Component Package contains 6 component groups:

- Research Cover Page,
- Key Person,
- Research Project,
- Organizational Assurances,
- Research Budget and
- General Components.

Additionally, the R&R has defined two additional components: Demographics and Population Study, that will be proposed to E-GRANTS as potential cross-agency components, but are considered too agency-specific to be included within the R&R Component Package.

The issues to be addressed for the CGAP project are how to define an XML schema that will:

- *meet NIH's data collection requirements, and*
- *allow other agencies to expand the Core data set or the R&R Component Package to meet their data collection requirements.*

\

Revised processes for grant application processing

The receipt of applications in electronic format and XML stream will change the receipt criteria, receipt business process and will affect the processing of the grant application throughout its life cycle. These changes will require system changes in eRA, business process changes in eRA and in ICs and policy changes to redefine the application and application receipt criteria.

The issue for the project is to define the new business flows, incorporate system changes in the overall eRA plans and work with the communities of users to exploit the opportunities for streamlining afforded by the new technologies.

3. The proposed solutions

The following sections describe a proposed solution to the many issues raised by electronic submission of applications. These solutions are described as functional building blocks and will be further refined as the project progresses.

3.1. Business to business exchange mechanism

To address the issue of large peaks of transactions, the proposed solution is to define an exchange mechanism with a controlled submission of the large files. The first component is to establish a computer-to-computer exchange; the second component is to define the rules by which the transactions are exchanged.

To allow computer-to-computer transactions and submission of electronic applications, the concept of business-to-business exchange is proposed for the design of the CGAP. The Business-to-Business or Business-to-Government exchange (simply B2B exchange) allows the computer systems of one organization to deposit or retrieve a predefined transaction (a message) from or to the computer system of the trading partner, without human intervention.

The B2B exchange is not the same interface employed by a user on a workstation submitting or entering data using a Web page or sending an email. The user would be interfacing only with the internal systems of his/her own organization or a service provider. The interface with NIH is done in the background via a computer transaction. In this case the transaction will be an augmented XML file. The computer-to-computer communications is bi-directional. For example the applicant system submits an application and the NIH responds with the system acknowledgement, acceptance notice, the referral assignment notice and other transactions.

A B2B exchange mechanism implies that the trading partners have a system “listening” to the external world and ready to receive transactions at any time. It usually means that a server running specific software is accessible through firewalls to authorized external partners. The trading partner needs to participate in the B2B exchange using the exact specifications stipulated for the Exchange.

It also implies that the trading partner systems are identified, certified and that the transactions to be exchanged are in a pre-approved structured format.

3.2. XML and standard transaction definition

To exchange structured information system to system, the security, transport mechanism, transaction packaging and most important the transaction content itself needs to be defined, agreed on and implemented at each node of the exchange.

For each type of transaction, a standard XML schema will be defined.

3.3. XML and binary attachments

The technology of choice in the present architecture of the Federal government is to use XML as the standard for information exchange between systems. The XML technology allows disparate systems to find a common information description and exchange standard. However an XML file is a text file where each element of the information payload is preceded and followed by a descriptive tag. A formatted document is a binary stream with characters that may interfere with the tag scheme of the XML stream. To reliably transport a binary document with XML:

- The document must be transformed or mapped from its binary characters to the printable characters of the ASCII character set

Or

- The document must be transported as an external attachment to the XML stream.

There are numerous ways of solving this problem. Some of the most used solutions are described in Appendix 2. The transaction is expected to be one physical file for the XML stream and the attached documents.

3.4. Controlled receipt business process model

To address the issue of large peaks of large transactions, the process of transferring the application transaction from the submitter to NIH has to be a controlled process. This controlled process can be accomplished within the framework of the information exchange by defining a transfer protocol and business process that allows for the control.

The submission process of the competitive grant application was broken down into 3 steps:
- The request for a submission initiated by the submitter
- The issuance of an acknowledgement of the submission request (the ticket)
- The controlled transfer of the application file.

Several models for the control process were considered. The model selection was based on early feedback and on the technical merits of the models. The model selected is as follows:
- The request for submission contains the file identification.
- The submitter places the file at the address agreed on in the in the trading partner agreement.
- The request ticket is placed in an NIH processing queue.
- When the place in the processing queue is reached, the NIH side of the exchange contacts the submitter and downloads the file.

The submitter of the application will be able to check on the status of the application submission by:

- Processing the automated messages provided by NIH as part of the process flow
- Submitting a request for status as a Web Service request to NIH.

3.5. Critical agreements with all trading partners

The creation of the exchange process described above requires participation of the external community as trading partners. The entire project is based on the following fundamental assumption about the process and the participants:

- The information exchange is based on a computer-to-computer exchange, where each trading partner has a device listening for messages at all times.

- The transfer of the body of the application, any other large document or any message that contains a binary attachment is a controlled process based on a submission request queue. The NIH initiates the transfer process.

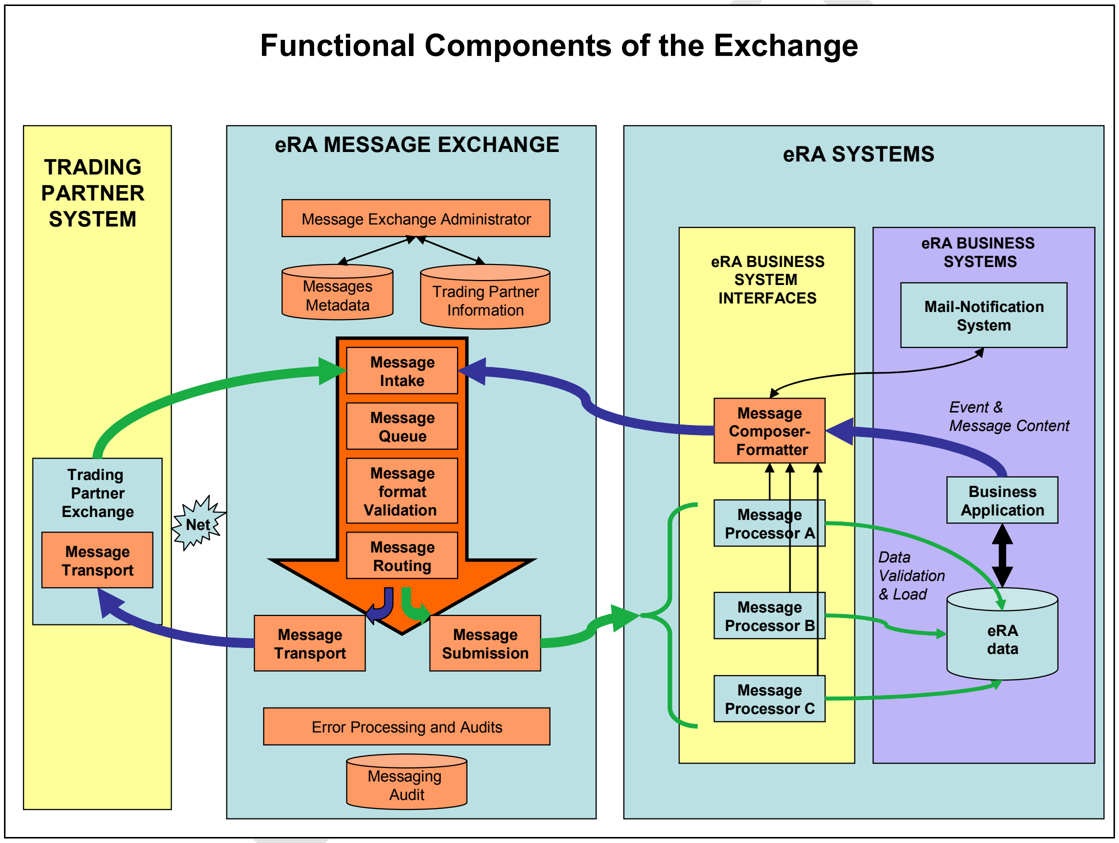
- The information exchange is based on an agreed on standard protocol and standard transaction definition. All trading partners will conform to these standards.

The selection of the business-to-business exchange solution conforms to the guiding principles established by the E-GRANTS initiative and the Federal Demonstration Partnership (FDP). These principles are provided in Appendix 1.

4. The technical approach and components

4.1. The exchange concept and components

The NIH will implement a business-to-business exchange infrastructure to send and receive message streams to and from trading partners. The following diagram illustrates the concepts and components of the exchange mechanism.



The eRA exchange message processing itself has the following components

- The message transport and intake mechanism
  The eRA exchange has a listener that takes in messages from registered trading partners
- A virus checker
  The messages are checked for binary content and viruses.
- A message queue or buffer
  Messages are stored in the exchange and queued for processing

- A component to validate the format of the message (not the content)
  Messages are retrieved from the queue and validated for format according to message type. No content is checked, no business rules applied.
- A component to determine the routing of the message
  The destination of the message is determined using the message type, message header and the exchange metadata.
- A component to submit the message to a specialized processor
  One of the dispositions of the message is to submit it to a specialized processor that deals with this specific type of message or message payload component.
- A mechanism to audit all messages
  All messages or attempts to submit a message are tracked.
- A component to transport outgoing messages.
  The component is similar to the intake mechanism.
- A component to trap and communicate errors
  Not represented in the flows are all the paths taken when an error occurs. For each action taken, the error processing path will be defined.

- Message Exchange administration component
  Information about trading partners, message structure and format is kept in a metadata repository. Most of the mechanism that drive the exchange are driven by the data stored in the administration module

- Trading Partner Exchange
  Each trading partner must also have some components of the exchange. At a minimum, compatible message transport component must be implemented.

- eRA Business Systems Interfaces
  The eRA business system interface is mainly composed of specialized message processors that interpret, validate and load the content of the message in the eRA databases.
  The outgoing messages generated by eRA systems are formatted into a standard message by a message formatter in the eRA business interface.

More details about the process flow are provided in section Process flows

4.2. The proposed standards

The following sections describe the standards tentatively selected by the CGAP team. The appendices describe alternative standards evaluated by the team. This section simply describes the principal characteristics of the technologies selected to implement the CGAP.

4.2.1. Transport protocols

The transport protocols available for securely transporting messages were briefly evaluated (see Appendix 2: The XML transport mechanism selection for more details). The CGAP team recommends the use of HTTPS as the protocol for version 1 of the implementation. Other protocols may be added over the course of the project. HTTPS is the secure version of the HTTP protocol. In HTTPS the data stream is encrypted. Most web sites used for financial and other confidential transactions use HTTPS as their transport protocol.

HTTP and HTTPS

Overview

Short for HyperText Transfer Protocol, HTTP is the underlying protocol used by the Web. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. HTTP is by far the most widely used protocol on the Web. Of the widely accepted protocols on the Internet, it is the one most often used by Web Services. Based on a stateless request – response system it is very lightweight. Because of its widespread implementation, HTTP has a variety of servers made for any server platform. It is also relatively simple to install and maintain. HTTP also has the ability to be piped across SSL. This ensures that the transmission is secure.

Advantages of HTTPS

HTTP is a building block of the web. This gives it a wide range of support, installation and configuration options. It can be tunneled through SSL, which is convenient for securing the transmission.

Disadvantages

HTTPS is not entirely reliable. For instance, a transmission could be interrupted at the source in the middle of a transmission and not generate an error. Also, depending on the application, its stateless nature could be a hindrance.
HTTP was primarily designed to handle serving static documents to a client. However, since then HTTP has been adapted to serve everything from dynamic content to streaming media. Because of this HTTP has been overloaded and could potentially cause a security concern.

Conclusion

Most Web Services examples use HTTP as the transport protocol. It is extremely simple to set up and get running. HTTP is well understood and omnipresent for web applications. It should be strongly considered for all implementations. However, HTTP's statelessness can create an additional burden on the Web Service implementation. The application is now required to maintain any state that it would need.

The CGAP project will start out using HTTPS for short message exchanges. Additional protocols will be introduced over time.

SCP

The Secure Copy protocol is commonly used to transfer files from one machine to another over the Internet. It will be considered as a second protocol for the transmission of the application files from the service providers. SCP provides more reliability and guarantees delivery.
However in the first phase of CGAP, the project team will try to limit implementation requirements at the service provider to one protocol.

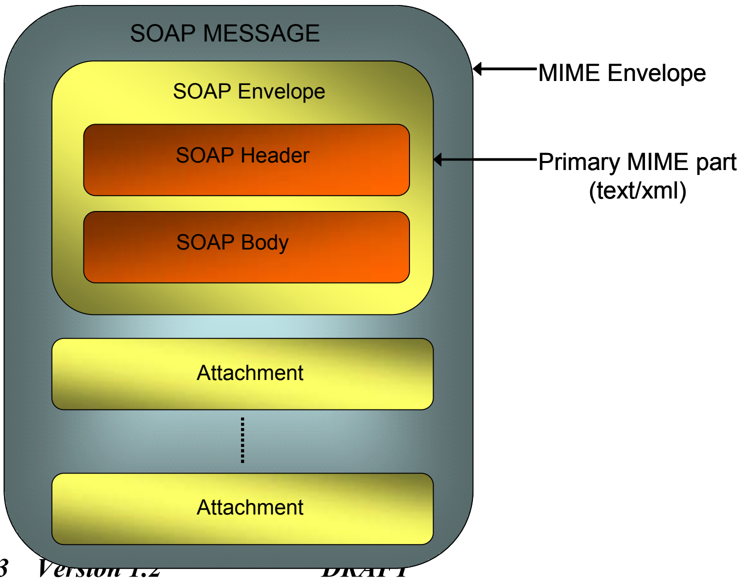
4.2.2. Packaging the transaction

The CGAP application receipt will contain structured data and documents. The documents will be external to the XML stream. The technology selected to package the XML file with the attachment files into one file/package is SOAP with attachments (SwA).

Simple Object Access Protocol (SOAP)

The SOAP protocol is used extensively in B2B exchanges to format messages. SOAP is a recommended practice in the Federal government. However SOAP by itself does not include attachments. The attachments in SOAP can only be defined as references using the `href` attribute. The `href` attribute allows pulling the embedded content out of the XML container, and replacing it with a link. That means that external references are essentially URLs. This technique is not suitable for most documents submitted to NIH through the XML stream. Therefore, this approach will not be appropriate as a generalized packaging solution.

SOAP with Attachments (SwA)



SOAP Messages with Attachments (SwA) is a fairly recent extension to the SOAP protocol. The SOAP message is wrapped in a MIME envelope. The first part of the multipart MIME (Multipurpose Internet Mail Extension) message is the XML SOAP document; the subsequent parts contain the attached data.

HTTP forms can be sent using MIME multipart; this means that all web servers should already have the proper MIME machinery built in.

Advantages

The advantage of this implementation is that it is very simple and easy to implement.

Disadvantages

A drawback to SwA is that it can not handle data streaming. This means that the client would have to buffer the entire attachment (for instance a multi-media data stream) before processing it. For the size of the CGAP application files and the type of processing, it is not anticipated that the components of the message have to start processing before the entire application is transmitted.

4.2.3. The Exchange standard – ebXML

There are a number of business-to-business exchange protocols and models defined. Many are proprietary commercial standards and products. A few exchange standards have been initiated by standard-setting bodies.

The CGAP team is proposing to use some components of the ebXML standard as a framework to develop the eRA Exchange mechanism.

ebXML

Overview

The ebXML (Electronic Business using eXtensible Markup Language) standard resulted from a combined effort of the UN (UN/CEFACT) and OASIS (the Organization for Advancement of Structured Information Standards). The organization was formed to provide an open XML-based infrastructure to enable the global use of electronic business information. ebXML currently works on all aspects of a standard, including an implementation framework, vocabularies, data dictionaries, and processes. ebXML is a modular suite of specifications that enables enterprises of any size and in any geographical location to conduct business over the Internet.

The mission of ebXML is to "*provide an open XML-based infrastructure enabling the global use of electronics business information in an interoperable, secure and consistent manner by all parties*".

The key components of ebXML are the Collaboration Protocol Profile (CPP), the Collaboration Protocol Agreement (CPA), the Business Process and Information Modeling, the Core Components, the Messaging and the Registry/Repository.

Collaboration Protocol Profile (CPP)

A CPP describes a company's offerings in a standard, portable way. Specifically, it describes the message-exchange capabilities and business collaborations that a company supports. It also describes the company's business processes, including how partners interact with this company. An interesting facet of a CPP is that a business collaboration includes both sides of a two-party B2B transaction. For example, in a buyer-seller situation, the CPP would describe not only the selling process and semantics of the seller, but also the buying process and semantics of the buyer.

Collaboration Protocol Agreement (CPA)

A CPA describes the exact requirements and mechanisms for the transactions that two companies perform with each other. It is formed from a manually or automatically derived intersection of their CPP's, which has been reviewed and agreed upon by both sides. This CPA becomes a contract between the two parties and specifies the "rules of engagement" for a particular collaboration.

Business Process and Information Modeling

ebXML also includes specifications for describing a business process in XML. This can include transactions, document flow, binary collaborations, data encapsulation formats, and more. These specifications are used by authors when constructing CPP's and are also used to describe and share business processes or information formats.

Core Components

Another crucial part of the ebXML standard is a set of ebXML schemas, also called components. These schemas contain formats for business data, such as dates, taxation amounts, account owner, exchange contract, and more. They are specific to business constructs and entities, but not aligned to any particular vertical industry.

Messaging

EbXML messaging is a format that encapsulates a message with all the related message-oriented middleware semantic (e.g., asynchronous/synchronous, reliability options). In particular, an ebXML message represents the visible part of the execution of a CPA, and has features specified to enforce the "Rules of engagement" specified therein. ebXML messaging is built on top of SOAP-encapsulated message-passing invocations (as opposed to RPC-style invocations). It extends the SOAP protocol by adding layered frameworks that support attachment, security and reliable delivery. The Message Services Specification can be found at http://www.ebxml.org/specs/ebMS2.pdf

Registry/Repository

The ebXML registry/repository is a service that stores CPP's, CPA's, ebXML core components and any other ebXML documents or fragments including WSDL documents, JAR files with Java code and even audio and video etc. It contains powerful query abilities to allow users to search for relevant components and potential business partners. The JAXR API can also be used to access ebXML registries. A supplemental report offers a way to locate ebXML registries using UDDI.

A key concept here is while UDDI provides a universal singleton source of references to Web service information, an ebXML Registry is a local container for the actual information itself. ebXML provides for publishing and discovery of almost any type of artifacts including CPP, Schemas, commonly used XML components, as well as Web services. Thus in a wide-area web service discovery scenario, a potential business partner would first search for a service in UDDI, which would contain a reference to a CPP or other documentation that is actually stored in an ebXML registry. The potential business partner then uses the information in UDDI to access the ebXML Registry where they subsequently discover the CPP for a business partner. They then use the CPP to initiate a partnership agreement for subsequent B2B transactions with the partner.

Advantages

- ebXML is a very comprehensive standard and it addresses all foreseeable business and technical B2B issues.
- ebXML is becoming a de facto industry standard.
- Its Messaging Services Specification, which integrates SOAP 1.1, is robust and comprehensive (RosettaNet, OAG and HL7 use this specification).
- Is an open standard, meaning no individual company owns or controls their development and they are freely available to all.

Disadvantages

- ebXML is very comprehensive and it is very complex to implement in its entirety.
- ebXML is a young technology. There are few implementations and the standards are likely to change.
- Does not define validation mechanisms for XML (does not refer to DTD, Schema or XSL files) for a vertical industry.

Recommendation

ebXML may be hard to put into operation in its entirety and it may be more than eRA needs. However it does provide a recognized and well documented framework to work with. Some components are more mature than others. For example the Messaging Services Specification offers recommendations for message packaging, reliable messaging, error handling, and security that provide a strong foundation.

The CGAP team recommends using this framework as a guideline for implementation of the eRA exchange. However if the standard is too complex for the size of the problem eRA is trying to solve, only the appropriate subsets of the standards will be implemented.

Reference
http://www.ebxml.org

4.2.4. CGAP XML Schema Strategy

The CGAP strategy for the design of the XML schema is a provisional solution dependant on the feedback from the E-GRANTS initiative. Once the E-GRANTS project has defined a standard transaction, the CGAP team will revise the XML transaction for the grant application.

4.2.4.1. Issues and goals of the XML schema strategy

The NIH CGAP team is working with the E-Grant initiative to define an approach to the packaging of grant applications into an XML-based data stream. There are other messages to be defined for the implementation of CGAP besides the grant application itself. However other messages such as outgoing aknowldgements will be much simpler and will not be discussed here.

The most difficult problem to be solved is the definition of the grant application transaction.

The E-Grant initiative will create a receipt process that is common to 26 agencies by finding a common ground for the submission process. The E-Grant process will act as a broker for the institutions, to all the agencies and programs participating in the project.

The NIH competitive grant form is form PHS 398. The data stream definition of the grant application in the EDI data set is the 194 transaction set. The grant application form approved by OMB for Federal use is the form 424.

The data on form 424 is considered core information. Not all core elements are mandatory. Each agency therefore has their own set of non-core data they will collect. Some non-core data is mandatory.

Given this complex problem:

What is the strategy for designing an XML transaction that allows all the non-core data to be represented?

The following section discusses the approach proposed by the CGAP team for the design of the XML transaction.

The team has opted to use XML Schemas rather than a DTD due to the flexibility and added capabilities of the schema.

As described in the previous section, the proposed technology for packaging the transaction is SOAP with attachments.

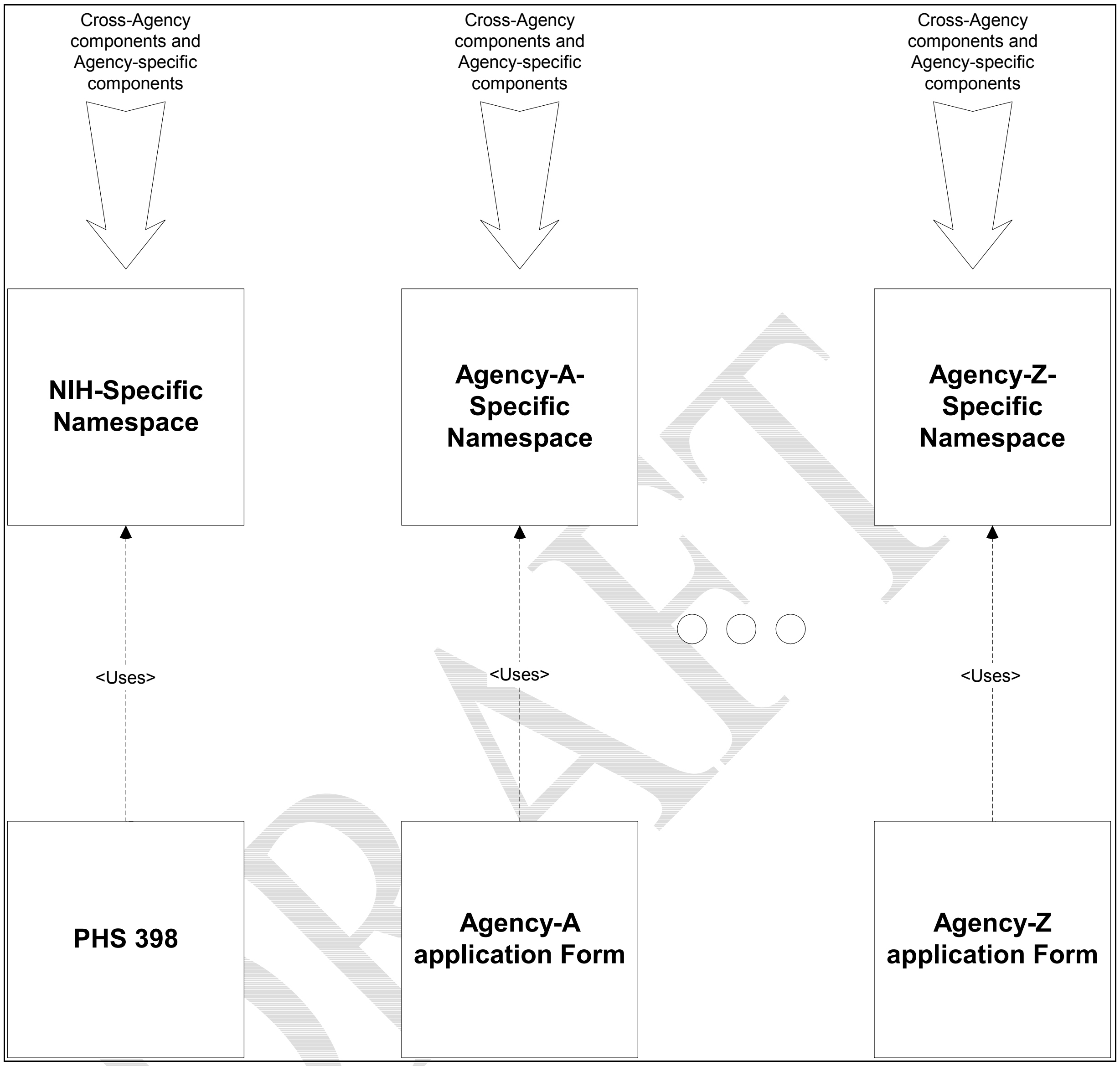
Part of the payload for the CGAP information exchange for a grant application will be a partial PHS398 application form, conveyed as an XML document. This fragment is referred to as a *partial* PHS398 application, since any portions of the PHS398 application kit which are expected to arrive in PDF format will be received as a separate attachment (within the same MIME envelope).

The goal at this stage of the project is to compose an XML schema, which will define the vocabulary (tag names) and data structure that will be expected in the PHS 398 datastream. The focus of the schema will be to enforce document structure almost exclusively. Some enforcement of constraints may be incorporated, wherever practical, if matters of value enumeration or data format patterns are obvious, and would be of unquestioned value to all agencies.

But merely producing a schema that meets CGAP's immediate needs is not the only goal. In devising the XML schema for the PHS 398, it is important to take full advantage of XML's object-oriented capabilities. Therefore the CGAP project team shall ensure that related data items within the PHS 398 application are grouped together sensibly to form the objects they represent, and that any commonality between objects will be extracted to form generic, reusable type definitions that can then be inherited and extended as needed.

As this effort unfolds, it is also important to extract and define vocabulary components (i.e., XML namespaces), which will be useful building blocks for *other* XML schemas, particularly when the NIH/CGAP initiative is extended to the cross-agency E-GRANTS arena. Therefore, as the CGAP project team works toward its initial goals, it shall also draw substantial inspiration from prior efforts (by E-GRANTS) to define points of commonality amongst various federal grant programs.

The following illustration presents the type of scenario that should be avoided, and forms the underlying argument in favor of the modular approach that will be undertaken by the CGAP project team. In the illustration below, it is supposed that no "core" or other cross-agency vocabularies are defined, and that each agency must define its own namespace to define not only its particular domain requirements, but those that happen to be common across agencies as well.


Cross-Agency components and Agency-specific components
Cross-Agency components and Agency-specific components
Cross-Agency components and Agency-specific components
NIH-Specific Namespace
Agency-A-Specific Namespace
Agency-Z-Specific Namespace
<Uses>
<Uses>
<Uses>
PHS 398
Agency-A application Form
Agency-Z application Form

This illustration is an over-simplification, but does serve to demonstrate the magnitude of the problem. If each agency defines its own agency-specific components, that part is fine. In fact, one should always expect that there will be agency-specific data collection requirements that cannot be sufficiently modeled in a "common" XML namespace. Without some "common" XML namespaces however, each agency must also undertake the task of modeling (separately) those data components that they all happen to share. What will inevitably result is a disparate set of vocabularies, all describing the same basic data elements, but each one with its own unique element names and inflections in the way the data is modeled, grouped, and associated. The unfortunate by-product of this scenario is that it would be incredibly difficult, if not impossible, to build an XML schema for eGrants, that would allow for the submission of a single non-repetitive datastream which can be easily parsed, transformed, and reassembled for the various agencies that it will ultimately serve.

The best way to ensure success for eGrants is to build upon the initiative that eGrants has already taken, to reach cross-agency consensus on common data structures and collection requirements. This consensus is reflected in the eGrants definition of the "Core" data elements, and in the draft specification of the "Research and Related Grant Application Component Package". The CGAP XML strategy is based on defining re-usable namespaces that will foster the definition of common building blocks of data across the participants in the E-GRANTS initiative.

The end results of using the common namespaces will be agency-specific schemas which employ much of the same element naming and data structuring, and which differ only in those ways in which the agencies themselves differ in collecting this data. This type of approach lends itself much more readily to the type of "one-face" approach to which eGrants is striving, and simplifies the job of parsing and transforming a single application datastream at the broker, into the specific variants that are required by each participating federal agency.

4.2.4.2. Strategy

Consequently, the CGAP strategy in defining an XML schema will be to build it in modular components. As a foundation, XML namespaces shall be defined that express the most common, cross-agency data structures. Then, a namespace shall be defined, which specifies the data structures specific to NIH data collection requirements. This NIH-specific namespace will define some unique structures of its own, but in many cases, it will define structures that actually inherit or build on and extend some of the foundation types that are defined in the cross-agency namespaces.

Finally, with all of these namespaces defined, a *PHS398 schema* shall be defined, which draws upon all of these namespaces, to specify the structure of the PHS398 application document. It is important to keep in mind that all of these components are really XML Schema Documents (.xsd files). However it is also important to note that only this last component, which is referred to as the *PHS398 schema,* will actually express the acceptable document structure that can be submitted via the CGAP datastream.

The intention of this modular approach is two-fold. First, many of the data items that are extracted into these component namespaces will be useful building blocks to inherit and extend, as the document structure of other NIH datastreams are defined later on. Second — and no less important — the cross-agency component namespaces should also serve as useful building blocks to agencies *other* than NIH, as each agency seeks to define its own agency-specific vocabularies and document structures.

4.2.4.3. Schema Components

4.2.4.3.1. Core Namespace

The purpose of an application *core namespace* is to provide a common vocabulary and data collection set that can be used for all Federal grant programs. The data that comprises the application "core", as defined by the E-GRANTS initiative, is as follows:

CORE = SF424 + DUNS, as defined in the X12 194 transaction set.

For the purposes of CGAP, the *core namespace* may also be expanded, to include any other generic data types that appear to be ideal base types for cross-agency use. An example of such a data type might be a type called "PostalAddressType" which is a type that can be conceivably used in any number of datastream settings, and does not appear to be limited to NIH, or even to just research-related applications. As the data in the Research and Related, Cross-Agency, and NIH namespaces are modeled (discussed later), opportunities will be taken wherever possible, to identify and extract generic foundation datatypes to the *core* namespace that seem to offer maximum reusability across the spectrum of cross-agency application document types.

4.2.4.3.2. Research and Related Namespace

The purpose of the *research and related namespace* is to provide a common vocabulary of cross-agency components that are relevant to federal research programs. The vocabulary that is expressed in this namespace shall conform to the initial recommendations contained in the "Research and Related Package" specification.

Wherever applicable, element, group, and type definitions that are contained in the *core namespace* shall be inherited and reused. Inherited *core* definitions will either be used to describe subelements within more complex *research and related* data structures, or will be extended within the *research and related namespace* to define a more robust data structure than the *core namespace* specifies.

4.2.4.3.3. Other Cross-Agency Namespace

The purpose of the *other cross-agency namespace* is to provide a common vocabulary of cross-agency components that are not necessarily relevant to all federal research programs. The definitions that populate this namespace are expected to grow over time, as the process of discovery continues and more cross-agency components are identified. Initially it shall contain the definitions suggested in the "Research and Related Package" specification, for "Other Components".

As is the case with the *research and related namespace,* element, group, and type definitions that are contained in the *core namespace* shall be build on and reused, wherever possible. Inherited *core* definitions will either be used to describe sub-elements within more complex *cross-agency* data structures, or will be extended within the *cross-agency namespace* to define a more robust data structure than the *core namespace* specifies.

4.2.4.3.4. NIH-specific Namespace

The purpose of the *NIH-specific namespace* is to provide a common vocabulary of components that are unique to NIH applications. Any data element that is collected on the PHS 398 form, which would require a new datatype, or an extension of an existing datatype from one of the other namespaces, will be defined here. These datatypes will be identified and added to this namespace as the structure of the PHS 398 document (see below) progresses, and it is determined that some portion of the document cannot be properly expressed using the vocabulary that is already defined in the common namespaces above (*core, research and related,* and *cross-agency).*

Element, group, and type definitions that are contained in the *core*, *research and related,* and *cross-agency* namespaces shall be inherited and reused, wherever possible. Inherited definitions will either be used to describe subelements within more complex *NIH-specific* data structures, or will be extended within the *NIH-specific namespace* to define a more robust data structure than the original namespace specifies.

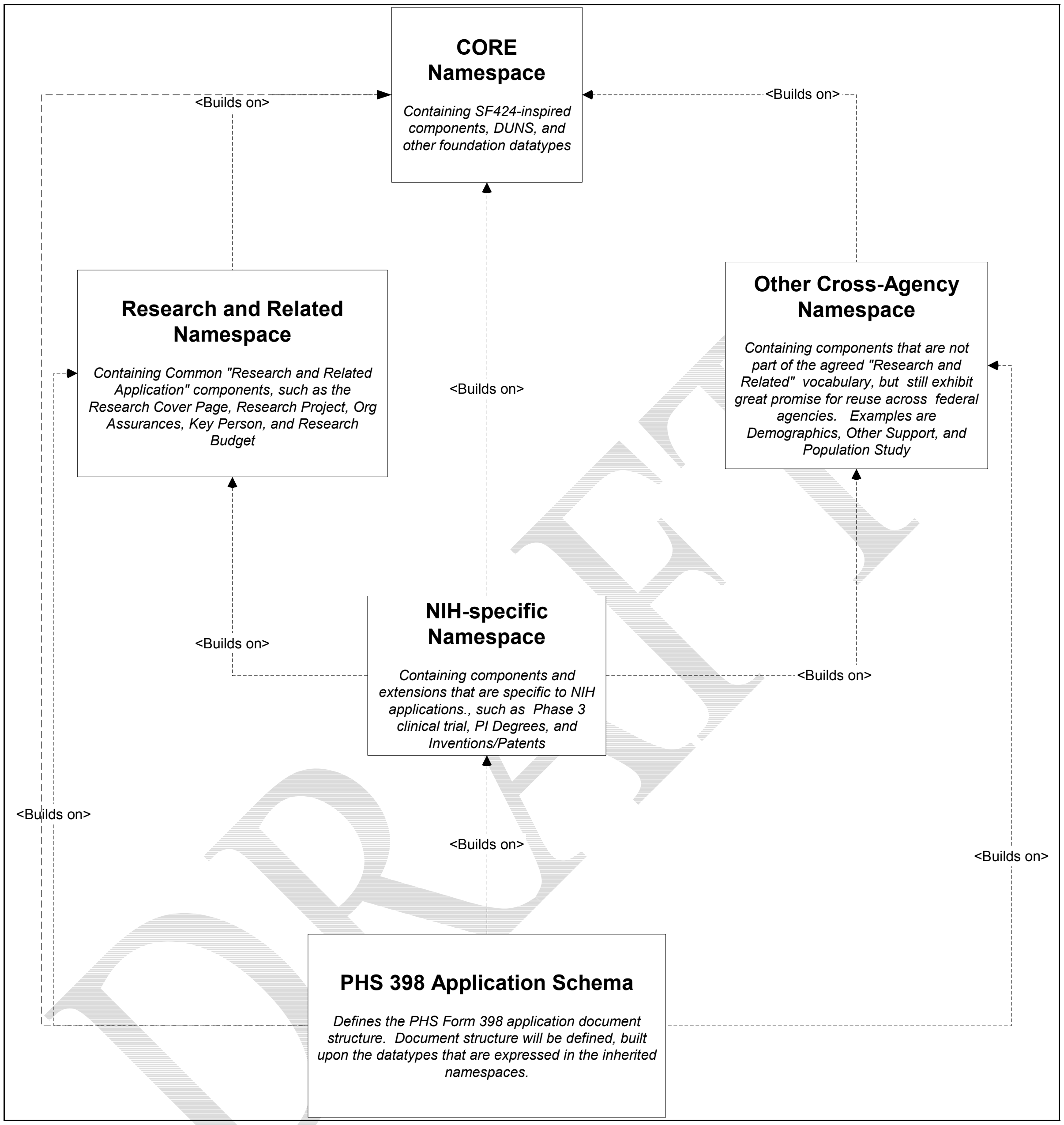
4.2.4.3.5. PHS 398 Application Schema

The PHS 398 document schema shall pull together all appropriate data definitions, found in the various namespaces, for the purpose of defining the PHS 398 application document structure. The document schema shall be mapped to the paper form and shall define the hierarchy of elements to which the electronic submission must comply. When the schema itself has been completed and published, a mapping will be included, so that each component of the paper PHS 398 form can be unambiguously traced to its counterpart in the XML document tree.

Element, group, and type definitions that are contained in the various namespaces shall be imported and used to define the schema. It should not be necessary for the schema itself to extend any existing namespace structures at this time, since the contents of the *NIH-specific* namespace should already reflect any complex datatypes that are peculiar to the PHS 398.

4.2.4.4. Illustration

The following illustration presents the various XML schema components that have been described in this section, and their relationships to one another. The illustration presents the most fundamental namespace (core) first, and demonstrates how increasingly specific namespaces will inherit from those that are more generalized in nature. At the bottom of the diagram is the actual document schema, which will govern the actual content and structure that is expected from the incoming application datastream.


CORE Namespace
Containing SF424-inspired components, DUNS, and other foundation datatypes
Research and Related Namespace
Containing Common "Research and Related Application" components, such as the Research Cover Page, Research Project, Org Assurances, Key Person, and Research Budget
Other Cross-Agency Namespace
Containing components that are not part of the agreed "Research and Related" vocabulary, but still exhibit great promise for reuse across federal agencies. Examples are Demographics, Other Support, and Population Study
NIH-specific Namespace
Containing components and extensions that are specific to NIH applications., such as Phase 3 clinical trial, PI Degrees, and Inventions/Patents
PHS 398 Application Schema
Defines the PHS Form 398 application document structure. Document structure will be defined, built upon the datatypes that are expressed in the inherited namespaces.
<Builds on>
<Builds on>
<Builds on>
<Builds on>
<Builds on>
<Builds on>
<Builds on>
<Builds on>

4.2.5. Security (early draft)

In the first phase of the project, the security implementation will be relatively conservative and simple. In subsequent phases more complex processes will be put in place.

For the first phase the following proposals are made:

- The trading partners are authenticated using Accounts and Passwords, both at the NIH site and for NIH access to the trading partner sites.
- The transmissions are encrypted using HTTPS
- The relationship between service provider and application submitter is not recorded in the NIH exchange (in phase 1). An Institution must have an IPF with a DUNS number, a PI must have a profile, a Service provider must be certified. NIH will not track if an institution has been authorized to submit via a specific service provider. Authentication and authorization for grant application submission is tied to the application.

In the second phase of the CGAP project the use of digital certificates will be evaluated for authentication and signatures.

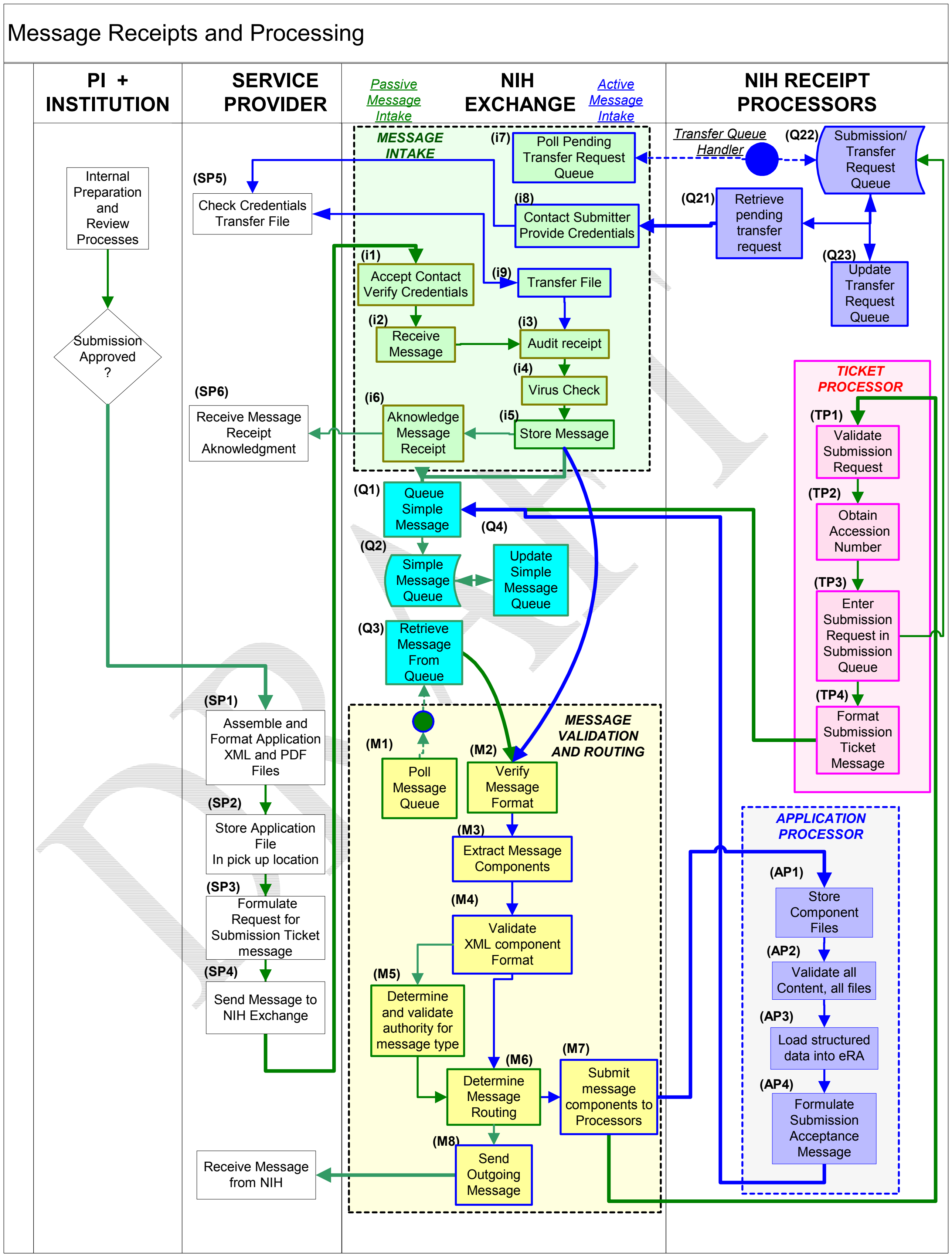
A proxy mechanism may be implemented to verify that a submitter must be authorized by the service provider. This may not be necessary for NIH but might be a self-service application for the Signing Officials of the Institutions.

A more complete evaluation of the following aspects of the security will be performed to address each aspect of security:
- encryption
- authentication
- authorization
- non-repudiation

4.3. Process flows

The following sections describe the functional flow of the CGAP process and of the exchange. This is a high level description that is intended to be an overall roadmap. Once the roadmap is agreed on, the CGAP team will develop prototypes for each component according to the general approach for the project plan.

4.3.1. The exchange process flow


Message Receipts and Processing
PI + INSTITUTION
SERVICE PROVIDER
Passive Message Intake
NIH EXCHANGE
Active Message Intake
NIH RECEIPT PROCESSORS
Internal Preparation and Review Processes
Submission Approved ?
(SP5) Check Credentials Transfer File
(SP6) Receive Message Receipt Aknowledgment
(SP1) Assemble and Format Application XML and PDF Files
(SP2) Store Application File In pick up location
(SP3) Formulate Request for Submission Ticket message
(SP4) Send Message to NIH Exchange
Receive Message from NIH
MESSAGE INTAKE
(i7) Poll Pending Transfer Request Queue
(i8) Contact Submitter Provide Credentials
(i1) Accept Contact Verify Credentials
(i9) Transfer File
(i2) Receive Message
(i3) Audit receipt
(i4) Virus Check
(i6) Aknowledge Message Receipt
(i5) Store Message
(Q1) Queue Simple Message
(Q2) Simple Message Queue
(Q4) Update Simple Message Queue
(Q3) Retrieve Message From Queue
MESSAGE VALIDATION AND ROUTING
(M1) Poll Message Queue
(M2) Verify Message Format
(M3) Extract Message Components
(M4) Validate XML component Format
(M5) Determine and validate authority for message type
(M6) Determine Message Routing
(M7) Submit message components to Processors
(M8) Send Outgoing Message
Transfer Queue Handler
(Q22) Submission/ Transfer Request Queue
(Q21) Retrieve pending transfer request
(Q23) Update Transfer Request Queue
TICKET PROCESSOR
(TP1) Validate Submission Request
(TP2) Obtain Accession Number
(TP3) Enter Submission Request in Submission Queue
(TP4) Format Submission Ticket Message
APPLICATION PROCESSOR
(AP1) Store Component Files
(AP2) Validate all Content, all files
(AP3) Load structured data into eRA
(AP4) Formulate Submission Acceptance Message
(SP7)

The diagram represents the functional components of the receipt and delivery of messages from and to external and internal partners.

The following sections are narratives describing the overall process. This process will be changed as the system goes through the analysis, design, and development process as well as from the feedback obtained through this draft from all external participants.

The PI and Institution may follow internal approval and submission processes that are not in the scope of the CGAP project.

Pre-requisites for electronic submission are:

1. The participating institution must have an Institution Profile in the Commons and DUNS number from the NIH eRA Commons.
2. The Principal Investigator has to have a Personal Profile and ID from the NIH eRA Commons.
3. The service provider transmitting the application must be a registered service provider with NIH and must have certified their software through a NIH test suite.

The Service Provider can be an integral part of the institution system.

The processes sketched out for the service provider are more end results than process. It will be up to the service provider to implement the processes to obtain these end results

The service provider will have to achieve these results:

<u>Service provider application preparation and ticket request steps</u>

- (SP1) Assemble and format application XML and PDF files
  The Service Provider system assembles into a single file the grant application with all its components. The file is packaged according to the standards defined in the technical sections of this document.

- (SP2) Store Application file in pick up location
  The method selected to transfer the file is for NIH to initiate the transfer of a file stored on the service provider machine.

  The service provider has to store the file in a location that will be given to the NIH exchange in such at way it can be transferred from the service provider to NIH based on the agreed on transmission protocol and process. The service provider has to resolve all firewall and access issues based on the exchange protocol and trading partner agreements. The security should such that only NIH can access the grant application files.

- (SP3) Formulate a request for submission ticket message

  The service provider system assembles a XML message to request a submission ticket for the grant application. The request has to contain the following information:
  - Identifying credentials for the service provider
  - Location of the application file
  - Check sum of the application file
  - DUNS for the submitting institution
  - Commons profile ID for the PI submitting the application
  - Accession number of the first submission if this is a revised submission within the same submission period.
  - Email address of a contact to be notified of the receipt of the submission

- (SP4) Send Message to NIH Exchange

  The Service provider has to initiate the contact with the NIH computers and send the XML stream for the ticket request. The Service Provider will expect an answer from the NIH exchange confirming the receipt of the message itself in the same session.
  This request is a conversation with the NIH exchange resulting either in a return message from NIH confirming that the message has been received or that the message failed. The ticket itself will be issued later as an asynchronous message.

<u>Message Intake Steps for messages initiated by the service provider</u>

These activities are considered passive message intake since the exchange will need to accept messages at the discretion of the sender.

Only messages with no binary payload or content will be received as passive messages.

- (I1) Accept contact and verify credentials

  The NIH exchange accepts the contact from the registered service provider opens a session and verifies the credentials of the service provider. This is part of the selected transmission protocol

- (I2) Receive the message

  The NIH exchange receives the message.

- (I3) Audit receipt

  The NIH exchange records in persistent storage that
  The transmission has been attempted,

The outcome of the transmission (received, failed, refused) and
When the transmission happened.

The audit records are used to diagnose problems, monitor security, activity and performance.

- (I4) Virus Check

  The NIH exchange will check if there are characters out of range in the message (no binary content for passive messages)
  The NIH Exchange will check if the content of the message is free from documented viruses (may not be needed after previous check).

- (I5) Store the message

  The NIH Exchange stores the message in persistent storage. The message is stored in the exchange in a queue for later processing.

- (I6) Acknowledge Message receipt.

  The NIH Exchange sends an acknowledgement message to the Service Provider to indicate success or failure of the transmission. This is part of the transmission protocol and is a system-to-system message.

<u>Message storage and queueing</u>

All messages are stored immediately after receipt, before their content is processed by NIH systems. This storage is a processing buffer or queue. Information about the message is stored in a queue mechanism. The messages are processed one by one in the order they were received according to processing capacity at NIH.

There probably will be separate queues for the incoming messages and outgoing messages. Only one queue is represented on the diagram for simplicity.

The message queue mechanism has the following components

(Q2) A place to store the queue entries in persistent storage
(Q1) A process to add a message into the queue
(Q3) A process to retrieve the next in line message from the queue for processing
(Q4) A process to update the message status in the queue after the message is processed, delete messages that have been completely processed.

A queue mechanism allows potential parallel processing of messages using multiple message validation and routing processors.

<u>Message validation and routing</u>
The message validation and routing component is designed to be symmetrical and process both incoming and outgoing messages. It is anticipated that many different

internal and external applications will submit messages through the exchange and therefore validation of outgoing messages will also be performed. For details of message components see the technical section of the SOAP with Attachments format.

- (M1) Poll the message queue

The NIH exchange has a process that check if any more messages are in the queue after it has completed processing the previous message. The processing of the messages is independent of the intake process.

- (M2) Verify message Format

The message format is validated. The message itself is complex with potentially multiple layers. This process will verify that the message itself is properly structured for subsequent processing.

- (M3) Extract message components

The NIH exchange processes the external layer of the message to extract the components. One of the components is expected to be an XML segment. Other payloads are optional.

- (M4) Validate XML component format

The NIH exchange verifies that the XML segment is a proper schema. The validation may have 2 phases:

1) The message is parsed for valid XML syntax and structure
2) The message is parsed to see if it meets the exact definition of the schema as described in the message metadata database of the exchange. Does this message correspond to a pre-determined message type and version.

- (M5) Determine and validate the authority for this message type

There may be a business rule check to see if the trading partner has the authority to submit this type of message. The checks are done against the metadata of the exchange not data contained in or derived from business processes. This step may be shifted to other point of processing. The types of checks that may be done are:

- Is this service provider registered to submit this type of message
- Is the service provider certified for this message type
- Is the outgoing message provider authorized for this message type

- (M6) Determine the message routing

The NIH exchange determines where the message is going. For incoming messages the process will look up the appropriate message processor for the message type and based on the message content. For outgoing messages the NIH exchange will determine of the requested destination is that of a service provider registered with the Exchange. Only messages going to certified exchange partners can be routed through the exchange.

- (M7) Submit message components to processors

The NIH exchange submits the incoming message component to the processor identified in the routing look up. This may be a call to pass the message or the message location to any number of systems or API designed to process certain types of messages. When a message has multiple components in its payload it is possible that more than one message processor is called for the different component types. Example: One processor routine may work on the XML portion of the grant application, a different processor may work on the PDF files payload. The make up of processors is a local design decision.

- (M8) Send outgoing messages

The NIH exchange processes outgoing messages by contacting the requested trading partner, presenting credentials, sending the message and waiting for confirmation that the message has been received. This process is similar to the message initiation process performed by the service provider.

Message processors

Message processors are business applications that work on a specific message type or message component. Each message type will be processed by a specific message processor. The message processor contains all the business rules for that type of message and typically will store the payload of the message in the appropriate table or location of the eRA database.

Ticket request message processor.

- (TP1) Validate submission request

The ticket processor will validate the application submission request.
The ticket processor will check if the message content contains the required information.

- The ticket processor will verify that the DUNS is for a valid institution that is a registered NIH eRA Commons user.
- The ticket processor will verify that the PPF id is a valid PPF for a PI registered in the NIH eRA Commons and affiliated with the submitting institution. The Commons User name may be used for the same purpose.
- The ticket processor will validate that in case the accession number is provided in the ticket request that the accession number exists and

corresponds to the submitting institution. Is a PI change allowed? The deadline must not be passed to submit substitute applications in version 1.

(TP2) Obtain Accession Number

The ticket processor obtains an accession number from the database. The accession number will be coordinated with the number series used in the paper receipts. The ticket processor may record the request for submission in the eRA tables (TBD)

(TP3) Enter submission request in submission queue

The ticket processor enter the request in the queue for transferring the application file itself. The queue will contain the information for the location and file characteristics to allow for retrieval.

(TP4) Format submission ticket message

The ticket processor assembles an XML message containing the date stamp of the submission and other identifying information of the submission. This outgoing message is a business message indicating that NIH acknowledges that the PI and Institution have submitted an application file to NIH. This is equivalent to the signature of receipt for the paper file at the loading dock. It does not mean the application has been accepted for review.

The outgoing message is submitted to the NIH Exchange of processing and transmittal to the submitter. Depending on the business rules and policies, the eRA e-mail notification system mail also be used to send an e-mail to inform the submitter of the receipt of the ticket.

Steps M1 through M8 are followed to process the outgoing message.

(SP7) Receive Message from NIH

The service provider receives a message from NIH. In this place in the process this is the message that represents the receipt submission ticket. It indicates that the deadline has been met for submission assuming that the file stored in the Service provier computer matches the characteristics of the file reported in the submission ticket.

Note: In version 1 it is not anticipated that the system will track the relationship between service providers and institutions. As long as the PI and Institution are registered in the COMMONS they can use any service provider to submit an application. The service provider itself must be registered and certified for this message type. However there is no tracing of delegation of authority from the PI or Institution to the service provider.

If in subsequent versions this type of control and tracking is necessary it will be build as a self-serving application for PI, Institutions and Service providers to use.

Transferring the application file from the service provider

A queue is set up to process file transfers requests. This queue is similar to message queues in its principles and mechanisms.

- (Q22) Submission transfer request queue

  The NIH exchange has persistent storage of the requests for file transfers as a separate queue. This queue contains information about the tickets to be processed.

- (I7) Poll pending transfer request queue

  This portion of the message intake is the active message processor.
  The message intake processor polls the queue to see if any more transfer requests are in the queue.

- (Q21) Retrieve pending transfer request

  The queue mechanism serves up the next transfer request in line.

- (I8) Contact submitter, provide credentials

  The message intake processor contacts the submitter of the transfer request and provides the NIH credential for accessing the external system and that particular file.

- (SP5) Check Credentials, transfer file

  The service provider accepts the contact and verifies the credentials submitted by NIH. The transfer protocol is initiated by NIH and the service provider system transfers the file to NIH.

- (I9) Transfer file

  The NIH message intake initiates the transfer of the file based on the standard protocol. At the end of the transmission the NIH message intake processor check the checksum of the file to match against the ticket information.

  NIH will always initiate the transfer of files that may contain a binary payload. This allows time to manage the processing time needed for NIH to check the content of the payload for potential viruses and reduces the security risks to the exchange.

- o Steps I3 through I6, SP6, Q1 are followed indicating that the file transfer is successful and an acknowledgement message is provided.

- o (Q23) Update transfer request queue

  The message transfer queue is update with the results (success or failure/retry) of the transfer.

The file received by NIH is processed as any other message through the Message Validation and Routing processor (steps M1 through M7). If successful the application file is then submitted to the Application Receipt processor(s).

<u>Application Receipt Processor</u>

The application Receipt processor is the software that checks the content of the grant application file against the business rules of the NIH, processes each component of the payload file and loads the data into the eRA database tables.

- o (AP1) Store Component files

  The Application receipt processor stores all the components of the message payload into the appropriate eRA tables and files. No transformation of data is made. The XML component of the payload is stored as an XML data type in the database.

- o (AP2) Validate all content, all files

  Each component of the payload is validated:
  - ▪ The PDF files are checked for page counts, average fonts and other presentation business rules required by NIH

  - ▪ The XML component is processed through a parser that extracts each data element of the message. Each data elements check according to the business rules for the grant application, constraints of the database. The errors are collected during processing and classified as cause for rejection, requiring a revised application, requiring additional information at a later date or simply informational

- o (AP3) Load structured data into eRA

  When the grant application data has passed the business rule filter , the data is laoded into the eRA tables.

- o (AP4) Formulate submission acceptance message

After the data is successfully loaded, a grant application acceptance message is formulated. This message is a business message informing that the application has been received and accepted by NIH, contingent on a QA review, additional information to be provided or other conditions.

The outgoing message is submitted to the NIH Exchange for processing.

Error Processing

At each process step there is the potential for errors. System messages will be generated and sometimes accumulated for transmission to the service provider or internal processors at NIH. There are probably 3 types of error messages

- Errors that occur during a synchronous session.
  These messages are transferred immediately and are usually part of the transmission protocol
- Errors that occur during internal processing in the exchange
  These errors may be accumulated and reported to the service provider as an asynchronous message, preformatted and coded or are reported to the exchange manager.
- Errors that occur during content processing and cause the business transaction to fail, even if the transmission and validation is successful.
  This type of message will be sent both as a system-to-system message but may also be sent as an e-mail business message notifying the contact provided in the ticket request of the problems encountered in processing the grant application.

4.3.2. The registration and trading partner agreements (early draft)

The process flows for registration and establishment of the trading partner agreements has not been developed yet.

These processes are not due to be designed and implemented until the second segment of the CGAP project. However some general concepts or rules are emerging and the following section summarizes these ideas.

- The ebXML standards will be used as guidelines to help define the eRA trading agreements. No full implementation of ebXML is planned in the first phase of CGAP.

- The NIH eRA Commons Institution Profile and Personal Profile will be expanded to register institutions and service providers participating in the CGAP program.

- Registration in the NIH eRA Commons is a pre-requisite for electronic submission both for the Institution, the PI and possibly the Key Personnel.

- The NIH will continue monitoring the E-GRANTS progress of using the Central Contractor Registration (CCR) system to be renamed the Business Partner Network (BPN), as a government wide registration service for grant applicants. The CGAP team will analyze the impact of using the BPN service as more details become available.

- The Service Provider will have to register with NIH and obtain an ID. The NIH most likely will specify the terms and conditions for the trading agreement. The Service Provider will have to sign up to these terms for each transaction type.

- The NIH more than likely will require a human contact and verify the identity of the SP applicant through mail or phone contact.

- The NIH will develop a test suite of applications to be processes through the service provider system and submitted as messages to the NIH.

- The NIH will establish a test environment where the Service Provider can test their software and interfaces to the NIH exchange.

- Once the Service Provider has validated their software output with the NIH test suite, the service provider will be certified for that version of the exchange and authorized to submit applications in behalf of its clients.

- Each message type will have a definition stored in the exchange. This definition will be under configuration management and accessible to the registered users.

- The trading partners will have to provide specific information for their systems and as their contacts. Some of the information will be self maintained by the Service provider. Some of the changes will require the test suite to be exercised again and the provider re-certified.

4.3.3. Interaction of the NIH exchange and the COMMONS IPF and PPF

The Institution and Personal Profiles will be maintained in a central place independent of the application. With each application, elements of the profiles will be re-used.

The interaction of the profile stored in the NIH eRA Commons, potential profile database stored at the institution and service providers has not been designed yet. The resolution of that problem has been deferred until after the basic exchange has been designed.

More than likely a set of transactions or remote procedure calls will be implemented to allow the service provider systems to access the NIH profile databases, retrieve and update information. The exchange mechanism and technology will support this type of transaction but the process flow will need to be defined first.

4.4. Operational and maintenance considerations (TBD)

The following list of items needs to be worked on:

- Exchange management processes and support
- User support
- Testing
- Partner certification
- Scaling and performance testing
- Change control and configuration management

4.5. Draft of the proposed technologies and technical architecture (TBD)

The technical architecture of the NIH exchange, test suite, development environments and security aspects have to be designed.

The technical architecture will specify such detail as the number of message queues, number of processors needed for each component, the redundancies and fail over capabilities and other infrastructure issues.

For each component of the functional architecture a specific technology will be selected.

For each component a technology prototype will be build and the version of a product or technology standard will be baselined.

5. CGAP processes after initial intake at NIH (TBD)

At this point in the project the CGAP team has focused on the intake of messages and grant application files.

The processing steps after receipt have not been analyzed fully. Business process changes and system changes will need to be designed and implemented to take advantage of electronic receipts.

In particular additional generic capabilities will need to be put in place to work with electronic applications. Some of these capabilities are:

- Annotation of documents
- Versions of documents
- Workflow of e-applications
- Formatting for display
- Formatting for printing
- Print, CD on demand

The business processes to be evaluated are:

- Receipt
- Multiple types of revisions of initial application
- Referral
- Receipt of appendices
- Review
- Council
- Grant management
- Program management
- Records management
- Progress reporting
- Renewals
- Multiple types of changes, supplements, amendments
- Close outs

Both the eRA processes and the NIH Institutes and Centers processes will need to be considered.

6. Appendix 1: FDP and E-GRANTS principles

E-GRANTS Vision

The E-GRANTS project will:

- Produce a simple, unified "storefront" for all customers of Federal grants to electronically find opportunities, apply, and manage grants.
- Facilitate the quality, coordination, effectiveness, and efficiency of operations for grant makers and grant recipients.

This combination of customer-facing vision and Federal internal improvement, as defined by the consensus of the grant-making agencies, provides a solid foundation for the efforts of the E-GRANTS initiative.

E-GRANTS Goals

Four goals for the E-GRANTS initiative were defined by consensus among the grant-making agencies:

1. Eliminate the burden of redundant or disparate electronic and paper-based data collection requirements.

2. Define and implement simplified standard processes and standard data definitions for Federal grant customer interactions.

3. Protect the confidentiality, availability, and integrity of data.

4. Standardize the collection of financial and progress report data in support of audit and performance measurement activities.

E-GRANTS Objectives and Timetable

Six major objectives for the E-GRANTS initiative were defined by consensus among the grantmaking agencies, along with the dates for completion of those objectives:

1. Finalize the E-GRANTS Business Case in support of partner requirements and other participant input (4/15/02).

   - Include defined categories of grants, solution concepts of simplified processes, solution concepts for standardized data, concept for achieving the goals, Program Management plan, organizational structure

2. Pilot a simple, unified way to find Federal grant opportunities via the Web (7/1/02)

   - Include standardized format and data elements

3. Evaluate the use or expansion of interagency and agency specific capabilities for discretionary grant programs (6/1/02)
   - Including COTS packages

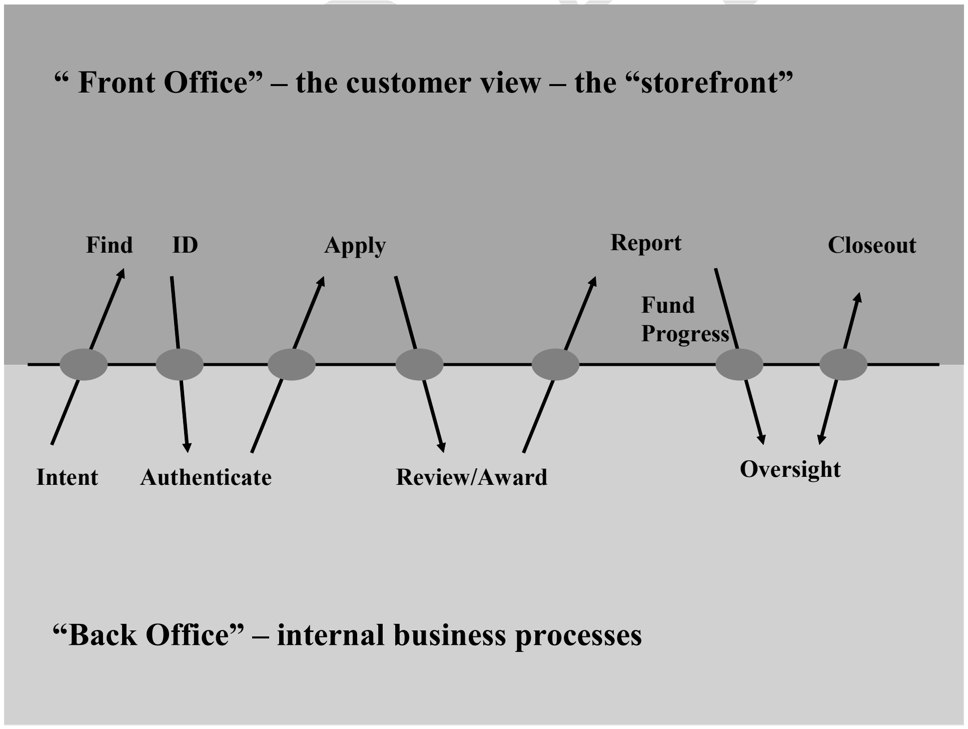
4. Work with E-Authentication PMO and privacy groups (ongoing)

5. Define application data standards (10/1/02)

6. Deploy simple, unified application mechanism (10/1/03)

E-GRANTS Scope: Focus on the "Front Office"

In taking immediate and measurable steps to address the vision, goals, and objectives defined by the consensus of the grant-making agencies, the E-GRANTS initiative will focus on the "Front Office" – the customer-facing aspects of the grant lifecycle. The following figure represents a two-sided view of the grant lifecycle that identifies parts of the lifecycle that are customer-facing ("Front Office") as well as those that are solely internal agency processes ("Back Office"):


" Front Office" – the customer view – the "storefront"
Find
ID
Apply
Report
Closeout
Fund Progress
Intent
Authenticate
Review/Award
Oversight
"Back Office" – internal business processes

By focusing on the “Front Office” aspects of the grants lifecycle, the E-GRANTS initiative will make a unified, simplified single set of processes available to potential applicants, applicants, and awardees. This will become the one-stop, electronic “storefront” where emerging e-business technologies and best practices are used to give grantees full service grants processing across all functions in the grants life cycle. The E-GRANTS storefront will be the single point of entry for grantees, offering both full general information exchange and secure e-business transaction processing.

FDP principles

This appendix is a copy of the core principles of the Federal Demonstration Partnership program. It has been reformatted to fit in the document. These principles are guidelines used on the CGAP Exchange project.

Federal Demonstration Partnership
Core Principles for Electronic Commerce in Grants Administration

- **Principle 1:** The interests of the funding entities, recipient organizations, and general public are best served when initiatives that involve electronic interactions between a federal funding entity and recipient organization are undertaken within the context of the implementation of PL 106-107
  - Electronic systems that maximize convenience to a single department, agency, or program without contributing to a government-wide solution should not be initiated
  - Any such programs already in existence should have a plan for converting to one of the government-wide solutions
  - Whenever possible, funds that would have been expended for a non-standard solution should be redirected toward a government-wide solution

    Advantage to funding entities:
    - avoids costs of one-of-a-kind solutions by leveraging inter-agency efforts

    Advantage to recipient organizations:
    - avoids need to learn to utilize and support a plethora of electronic commerce solutions

- Principle 2: Consultation with recipients early in system development increases both utility and satisfaction
  - The FDP 106-107 committee has reviewed several systems that have not been developed with recipient community consultation, and they do not address the needs of the recipient community.
  - Consultation can and should include beta testing with volunteer partners from the recipient community.

    Advantage to funding entities:
    - improve the quality of service and systems

    Advantage to recipient organizations:
    - reduced workload

- Principle 3: Interoperable systems for the electronic exchange of data in support of grants administration processes are possible only when government-wide data standards, such as the ANSI X-12 transaction sets, are used

- Funding entities should refrain from requiring data that is not included in the government-wide data standards
- Data definitions should be standard across funding entities, such that any given data element means the same thing for all funding entities where non-standard data is required by statute, funding entities should devise a mechanism for collecting this data that doesn't compromise the use of data standards (e.g., collect the information as part of the narrative enter this information into internal funding entity systems as necessary)
- Funding entities should take into account the time and expense incurred by recipient organizations when there are changes to the data standards

Advantage to funding entities:

- allows use of shared, government-wide systems

Advantage to recipient organizations:

- allows the use of one system for all federal funding entities

- Principle 4: The promise of electronic commerce in grants administration depends on each funding entity and recipient organization implementing one of the approved data exchange mechanisms.

- Funding entities should provide recipients the option to use any one of the approved data exchange mechanisms, and use the Federal Commons for transactions that use an exchange mechanism other than that supported by the entity
- Recipient organizations must commit to the use of one of the approved data exchange mechanisms
- Business-to-business transactions are appropriate for funding entities and recipients who have a sufficiently large number of grants to merit the investment in electronic grants administration systems
- Consumer-to-business transactions are appropriate for funding entities or recipients with smaller volumes of grants
- Electronic submissions are in lieu of paper, not in addition to

Advantages to funding entities and recipients:

- Each can use the transmission mechanism most appropriate to their needs, and
- Electronic transmission eliminates the need for most paper
- Provides an electronic record of the transaction

- Principle 5: Funding entities must respect recipient organizations' need to know what is being proposed by individuals and groups within the organization, and the role organizations play in providing quality control over proposal submissions, and monitoring deliverables

- Electronic systems should incorporate institutional approval of proposals, and should provide access to technical report submissions
- The electronic identity of the authorized institutional officials has to be

established and used

- Any statutorily mandated, proposal-specific assurances, representations, or certifications can be provided only when the organization is involved in the submission process

Advantages to funding entities:

- Ineligible proposals do not reach the funding entity,
- Appropriate assurances, representations, or certifications are obtained, and
- The assistance of recipient organizations is obtained for monitoring deliverables

Advantages to recipient organizations:

- Recipients are not placed in the position of choosing between accepting awards based on proposals it would not have supported (at least not without revisions) and standing between a principal investigator and her/his funding.
- Organizational approval of proposals reduces the negotiation necessary to reach agreement after an award is made
- Recipients have the information necessary to address delinquency of technical reports before funding to other investigators is threatened or audit findings are reported

- Principle 6: Data should be collected once, at its inception, without redundant data entry

  - Data ought to be entered only once and updated as needed, by the designated person(s) at the responsible organization authorized to provide it. The submitting organization is the entity responsible for assigning responsibility and authority to its individuals within the context of the roles agreed to by the parties. Once entered, data may be updated only in accordance with established standards (e.g., proposal data may not be updated after a submission deadline except for administrative corrections and with the consent of the affected agency).

Advantage to funding entities and recipient organizations:

- elimination of redundant data entry

- Principle 7: Electronic data must be secure from unauthorized access during transmission, storage, and subsequent use by the funding entity

  - Security has to be provided to protect the technical narrative of proposals, demographic information, proprietary information, and other information protected under law

Advantage to funding entities and recipient organizations:

- Providing data security enables all parties to avoid the liability associated with failing to observe legal requirements for protecting personal and technical information.
- Research results and ideas are safe from misappropriation by unscrupulous individuals and organizations

- Principle 8: The utility of systems to be used by recipients is greatly enhanced when the systems are pre-loaded with as much funding agency legacy data as possible.

  Advantage to funding entities and recipient organizations:
  - elimination of redundant data entry

- Principle 9: Data integrity is essential

  - Data transmission has to be confirmed as accurate and complete
  - The source and content of the transmission has to be non-refutable
  - Where funding entity actions are based on graphical information, such as a technical proposal narrative or report that includes images or other graphics, the funding entity must assure that it does not allow images to be degraded in a way that could result in decisions adverse to the recipient

  Advantage to funding entities and recipient organizations:
  - A high degree of data integrity increases the extent to which both faculty and program personnel rely on the system, rather than resorting to paper
  - Reduces the probability that deserving research goes unfunded because of poor or inaccurate representation

- Principle 10: System performance must be adequate for the processes and applications supported

  - System capacity sufficient to handle peak loads
  - System response time is acceptable

  Advantage to funding entities and recipient organizations:
  - A high degree of system reliability increases the extent to which both faculty and program personnel rely on the system
  - A reliable system avoids problems with making exceptions for late proposals, reports, etc.

- Principle 11: Funding entities are responsible for providing training materials and support of recipient organization personnel

  - There is a help desk, or equivalent means for providing ready assistance to users
  - There are training materials and training opportunities
  - There is documentation and instructions for users

  Advantage to funding entities and recipient organizations:
  - Training and support increase user efficiency

7. Appendix 2: Selection of XML parsers

Overview

The purpose of this document is to describe the various ways an XML document can be parsed and validated against a given set of rules. The rules checked apply to the structure of the XML document as well as content.

References

http://java.sun.com/xml/jaxp/index.html
http://www.jdom.org
http://java.sun.com/xml/jaxb/index.html
https://www6.software.ibm.com/developerworks/education/x-xschema/x-xschema-1-1.html

7.1. XML Parsing Techniques

This sections covers low-level-based APIs that deal directly with an XML document's content. The two most common APIs are SAX, the Simple API for XML and DOM, the Document Object Model. Additionally JDOM, and JAXP Sun's Java API for XML Processing will be covered.

JAXB, Sun's Java Architecture for Data Binding is an example of a high level API that can be used for the marshaling and un-marshaling of XML data will be discussed as well.

SAX

SAX is the *Simple API for XML parsing*. SAX was the first widely adopted API for XML in Java, and is a "de facto" standard. The current version is SAX 2.0. SAX is an **event-based API**, which reports parsing events (such as the start and end of elements) directly to the application through callbacks, and does not build an internal tree representation of the XML document. The application implements handlers to deal with the different events much like handling events in a graphical user interface.

SAX is an interface that must be implemented by a third part driver/parser. Although you can have the SAX2 interfaces without a driver, that's not useful; it would be like using JDBC without a database driver. Some of the commonly used SAX 2.0 compliant parsers are:

- Apache Xerces
- IBM XML4J
- Sun Crimson
- Oracle XML Parser

Most current Java programming environments include SAX2 interfaces in the core of their XML support and some include a parser as well. JDK 1.4 ships with the crimson parser as a default parser. You can bootstrap SAX to use a specific parser of your choice without having to change your code. SAX bootstrapping options are covered in Appendix A.

SAX processing model consists of the following steps (See Figure 1):

1. Implement a class that extends the DefaultHandler or implements the ContentHandler interface and hold callback methods.
2. Instantiate a new SAX parser, set the parser to your implementation of the handler, parse the XML document.
3. The parser reads the XML document sequentially (no random access). The parser fires events as it encounter XML elements and content. The class implemented in step 1 handles the events.

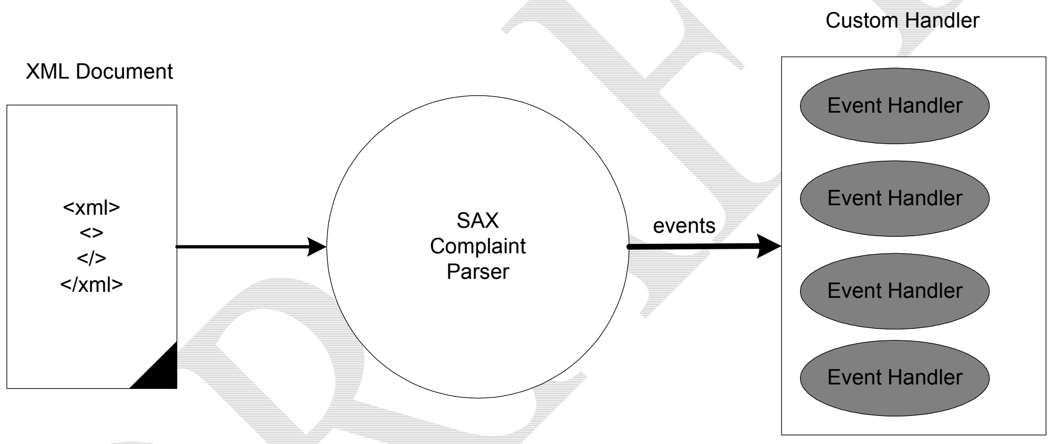


Figure 1 SAX Parser Model

Example:

The Handler Class

```
import org.xml.sax.Attributes;
import org.xml.sax.ContentHandler;
import org.xml.sax.Locator;

class SAXHandler implements ContentHandler {
        public void startDocument() {
                System.out.println("Start Document");
        }
        public void endDocument() {
                System.out.println("");
                System.out.println("End Document");
        }
        public void setDocumentLocator(Locator locator) {
        }
        public void startElement(
                String namespace,
                String name,
                String qName,
                Attributes atts) {
                System.out.print("<" + name + ">");
        }
        public void endElement(String namespace, String name, String qName) {
                System.out.print("</" + name + ">");
        }
        public void characters(char[] ch, int start, int length) {
                String textData = new String(ch, start, length);
                System.out.print(textData);
        }
            public void processingInstruction(String target, String data) {
        }
        public void startPrefixMapping(String prefix, String uri) {
        }
        public void endPrefixMapping(String prefix) {
        }
        public void ignorableWhitespace(char[] ch, int start, int length) {
        }
        public void skippedEntity(String name) {
        }
}
```

The Parser Code

```
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.XMLReaderFactory;

public class TestSAX {
        public static void main(String[] args) throws Exception {
                XMLReader parser = XMLReaderFactory.createXMLReader();
   parser.setContentHandler(new SAXHandler());
   parser.parse("c:\\file.xml");
        }
}
```

DOM

The Document Object Model, a W3C standard, is a platform- and language-neutral interface that will allow programs to dynamically access and update the content, structure of an XML document. The XML document is represented as tree structure (hierarchy) of Node objects. Some of these Node objects may have child nodes of various types, while others are 'leaf' nodes, which can have nothing below them. Most of the SAX parsers mentioned above provide DOM level 2 implementations. DOM as a tree-based API allows an application to navigate the tree, i.e., they provide random access to the xml structure, while SAX is sequential in nature and allow only one pass through the XML document.

Representing XML documents as tree structures in memory can be useful for a wide range of applications, but they normally put a great strain on system resources, especially if the document is large. Furthermore, if the application needs to build its own strongly typed data structures rather than using a generic tree corresponding to an XML document, then it becomes inefficient to build a tree of parsed nodes, only to map it onto a new data structure and then discard the original. In this case using SAX with a content handler would or even JAXB would be more efficient.

DOM processing consists of the following steps:

1. Get a DOM parser.
2. Parse the XML document.
3. Get Document object form the parser.
4. Get the root Element of the Document.
5. Now you can navigate throw the DOM tree elements and modes. Various get methods are available to extract the XML data and attributes from the tree elements.

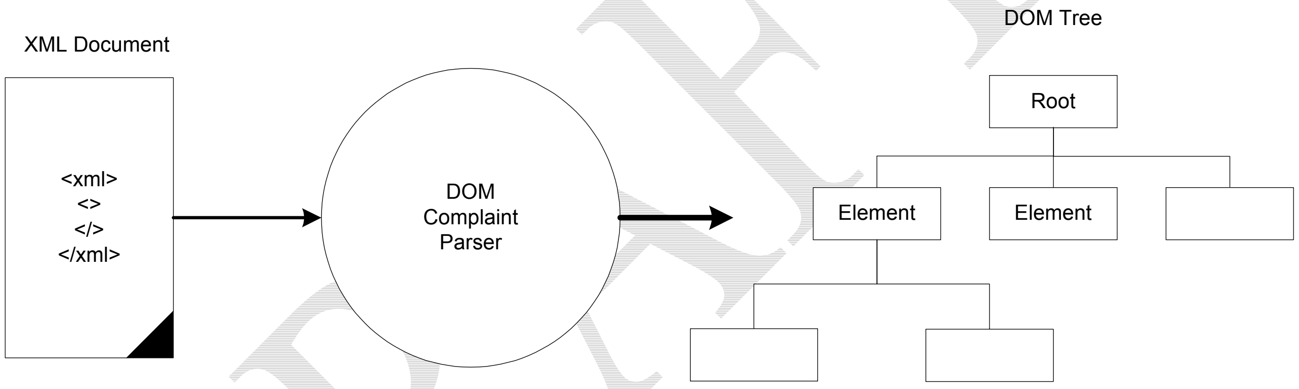


Figure 2 DOM Model

Example:

```
public class ParseDoc {
 public static void main (String[] args) throws Exception {
  DOMParser parser = new DOMParser();
  parser.parse("file.xml");
  Document doc= parser.getDocument();
  Element root = doc.getDocumentElement();
  exploreNode(root);
 }
 static public void exploreNode(Node node) {
  if (node.getNodeType() == Node.ELEMENT_NODE) {
   System.out.println("<" + node.getNodeName() + ">");
   NamedNodeMap attr = node.getAttributes();
   for (int x=0;x < attr.getLength();x++) {
    System.out.print(" Attribute ");
    System.out.print(attr.item(x).getNodeName());
    System.out.print(" has a value of ");
    System.out.println(attr.item(x).getNodeValue());
   }
   NodeList children = node.getChildNodes();
   for (int i=0;i<children.getLength();i++) {
    exploreNode(children.item(i));
   }
  }
  }
}
```

JDOM

Since DOM was developed as language neutral, it does not take into consideration and java language specific features. It also uses DOM dictated objects to represent the XML tree, which is seen by Java developers to be complex, or hard to use. JDOM was designed as an alternative to DOM. JDOM is both Java-centric and Java-optimized it also uses Java collections to represent node lists. JDOM also provides some extra features that DOM does not. One example of that is, the ability to output XML tree structure into XML text.

JDOM is not an interface, but rather an implementation that provides direct services to represent the XML document as a tree of elements and nodes. So underneath the hood, JDOM uses a SAX or DOM parser to parse the XML document and build the JDOM tree. Using SAX with JDOM is the preferred option since SAX is more efficient and faster than DOM.

JDOM processing consists of the following steps:

1. Get a SAX or DOM builder/parser
2. Build the XML document
3. Get the root Element of the Document
4. Now you can navigate throw the JDOM tree elements and modes. Various get methods are available to extract the XML data and attributes from the tree elements.

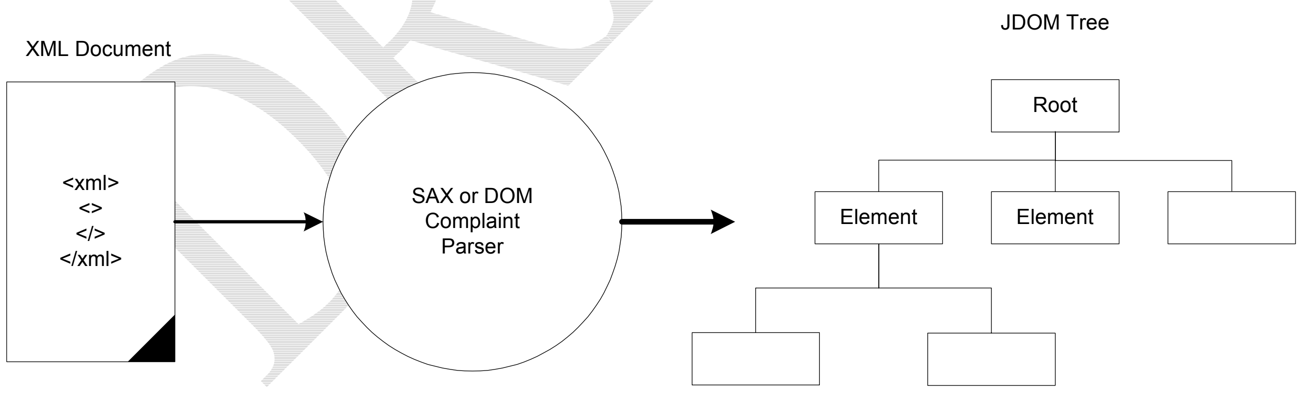


Figure 3 JDOM Model

Example:

```
try {
      //Creating the XML document form an input source
      SAXBuilder builder = new SAXBuilder();
      builder.setValidation(true);
   TestErrorHandler errorHandler = new TestErrorHandler();
      builder.setErrorHandler(errorHandler);
      Document doc = builder.build("C://personal.xml");
      if (errorHandler.errors.size() > 0) {
      System.out.println("Error(s)" + errorHandler.errors);
      }
   else{
      Element root = doc.getRootElement();
      // do something with the root element

         //Writing the XML document to an output source
            XMLOutputter outp = new XMLOutputter();
            outp.setTextTrim(true);
            outp.setIndent("  ");
            outp.setNewlines(true);
            outp.output(doc, System.out);
      }
}
```

JAXP

JAXP supports processing of XML documents using DOM, SAX, and XSLT. JAXP enables applications to parse and transform XML documents independent of a particular XML processing implementation. Depending on the needs of the application, developers have the flexibility to swap between XML without making application code changes. The latest JAXP release is 1.2. This new release adds support for XML schema and an XSLT compiler (XSLTC).

JAXP requires access to either the DOM or SAX APIs. Although JAXP is an API itself, you must still install one of these two APIs and make them accessible to JAXP before you can use the JAXP API with an XML parser. Some installations of the DOM and SAX APIs interact with the underlying XML parser in ways specific to the accessed parser. Accessing the parser using these types of proprietary methods can lead to greater difficulty when you want to update or modify your code or the underlying XML parser. Because JAXP provides a method of interfacing with the DOM and SAX APIs, it allows you to perform the same functions without having to create proprietary techniques for interacting with the parser.

JAXP also provide an interface to allow for XML transformation, the purpose of this interface, again, is to provide a vendor-neutral XML transformation capabilities to the Java language. Figure 4 shows an overview of the JAXP API model. The table below lists some of the common JAXP 1.1 compliant parsers and transformation engines.

Parsers

- Apache Xerces
- IBM XML4J
- Sun Crimson
- Oracle XML Parser

Transformation Engine

- Apache Xalan-J

Figure 4 JAXP Model

Example

```
//SAX Example
public class Main{

  public static void main(String argv[])
  {
    // Use the validating parser
    SAXParserFactory factory = SAXParserFactory.newInstance();
    factory.setValidating(true);
    factory.setNamespaceAware(true);

    try {

      // Parse the input
      SAXParser saxParser = factory.newSAXParser();
      saxParser.parse( new File(argv[0]), new MyContentHandler());

    } catch (Exception e) {

    }
//DOM Example
public class Main{

  public static void main(String argv[]){
   DocumentBuilderFactory factory =
      DocumentBuilderFactory.newInstance();
    //factory.setValidating(true);
    //factory.setNamespaceAware(true);
    try {
      DocumentBuilder builder = factory.newDocumentBuilder();
      document = builder.parse( new File(argv[0]) );
    } catch (Exception e) {
    }
```

JAXB

Java Architecture for XML Binding (JAXB) provides an API and tools that automate the mapping between XML documents and Java objects.

JAXB makes XML easy to use by compiling an XML schema into one or more Java technology classes. The combination of the schema derived classes and the binding framework enable one to perform the following operations on an XML document:

- Un-marshal XML content into a Java representation
- Access, update and validate the Java representation against schema constraint
- Marshal the Java representation of the XML content into XML content

JAXB gives Java developers an efficient and standard way of mapping between XML and Java code. Java developers using JAXB are more productive because they can write less code themselves and do not have to be experts in XML.

To build a JAXB application, start with an XML schema. After obtaining an XML Schema, you build and use a JAXB application by performing these steps:

1. Generate the Java source files by submitting the XML Schema to the binding compiler. One can use custom binding declarations to override the default binding of XML Schema components to Java representations.
2. Compile the Java source code.
3. With the classes and the binding framework, write Java applications that:

    - Build object trees representing XML data that is valid against the XML Schema by either unmarshalling the data from a document or instantiating the classes you created.
    - Access and modify the data.
      Optionally validate the modifications to the data relative to the constraints expressed in the XML Schema
    - Marshal the data to new XML documents.

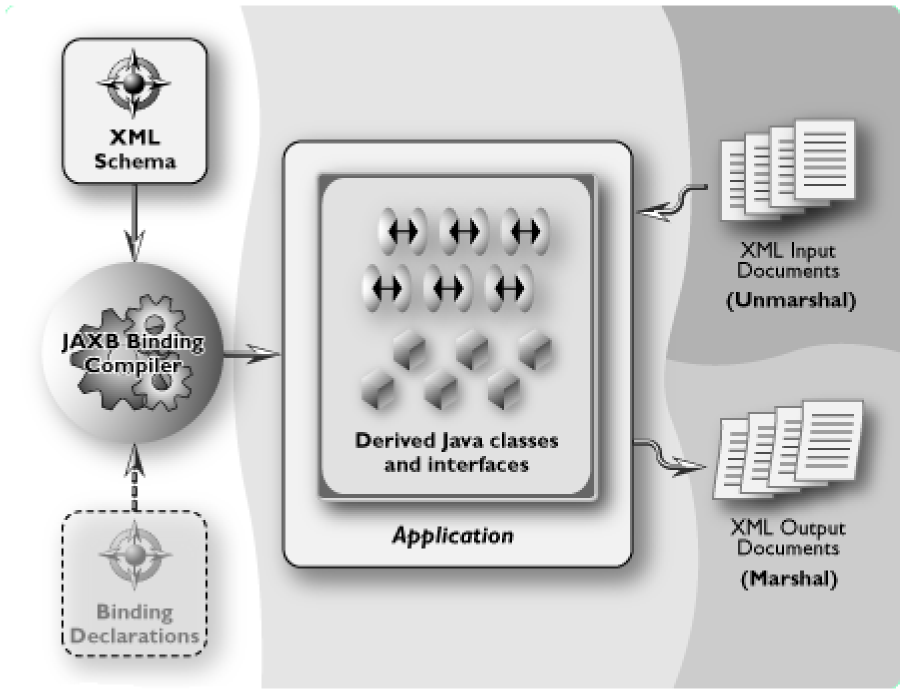


Figure 5 JAXB Model

JAXB is a new specification and so far only a reference implementation form Sun is available. Commercial and third party implementation would properly be available in the coming months.

Example:

```
   1. Create the Schema file
   2. Generate the schema objects
       xjc -d srcDir -p test.jaxb.model schemaFileName.xsd
   3. Compile the code.
   4. Here is a code sample using the generated objects to read and
      write an XML file confirming to the schema.
   try {
            JAXBContext jc = JAXBContext.newInstance("test.jaxb.model");
            Unmarshaller u = jc.createUnmarshaller();

      FileInputStream fis = new FileInputStream("C://personal.xml");

      Object o = u.unmarshal(fis);;

            //create a Marshaller and marshal to a file
            Marshaller m = jc.createMarshaller();
            m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
Boolean.TRUE);
            m.marshal(o, System.out);
   }
```

7.2. XML Validation

XML documents can be validated in two ways, DTD and Schema. Schema is the newer of the two and it provides better validation features over DTD. So this document will focus on validation using XML Schema only.

Keep in mind that XML Schema support is new in many parsers and it is provided as a vender added feature. Which will require the application to use a parser specific feature or methods to validate against an XML schema, which can make your code less portable. JAXP 1.2 specifications try to address this problem by providing a uniform access to scheme validation, which we will discuss in the next section

Validating when using SAX, DOM, JDOM and JAXP

SAX, DOM, JDOM and JAXP processing models provide a setFeature() and setAttribute() methods that allow us to pass on to the parser a set of validation instructions. For example to turn validation on, you can use the following method:

```
 parser.setFeature("http://xml.org/sax/features/validation", true);
```

This option by default will turn DTD validation for the document as specified by the XML 1.0 standards. Since schema validation is provided by vendors as an option, there is no standard way to turn schema validation on a parser. For example the apache Xerces parser uses the following method to allow for schema validation.

```
parser.setFeature("http://apache.org/xml/features/validation/schema",tru
e);
```

Other parsers might use different URI to describe their schema validation feature or provide some other methods. Using JAXP 1.2 compliant parser can make life a little bit easier since JAXP 1.2 provides a standard way of configuring XML schema validation for both SAX and DOM parsers. See examples below:

Using SAX Parser

```
try {
      SAXParserFactory spf = SAXParserFactory.newInstance();
      spf.setNamespaceAware(true);
      spf.setValidating(true);
      SAXParser sp = spf.newSAXParser();
      sp.setProperty(
            "http://java.sun.com/xml/jaxp/properties/schemaLanguage",
            "http://www.w3.org/2001/XMLSchema");
      sp.setProperty(
            "http://java.sun.com/xml/jaxp/properties/schemaSource",
            "http://www.example.com/Report.xsd");
      DefaultHandler dh = new DefaultHandler();
      sp.parse("http://www.wombats.com/foo.xml", dh);
}
```

Using DOM Parser

```
try {
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        dbf.setNamespaceAware(true);
        dbf.setValidating(true);
        dbf.setAttribute(
              "http://java.sun.com/xml/jaxp/properties/schemaLanguage",
              "http://www.w3.org/2001/XMLSchema");
        dbf.setAttribute(
              "http://java.sun.com/xml/jaxp/properties/schemaSource",
              "http://www.example.com/Report.xsd");
        DocumentBuilder db = dbf.newDocumentBuilder();
        Document doc = db.parse("http://www.wombats.com/foo.xml");
}
```

Validation Error Handling

Typically the validation is done while the parser is marshaling the XML document, any structure or content errors are passed on to an error handler. Most parsers' default error handlers throw a SAXParseException when an error is encountered, and the processing of the XML document stops. This behavior might be acceptable when encountering a structure type error that prevents the parser form completing the parsing of the XML document. As for content errors, it is preferred that errors are queued and send back to the application in one pass through the document. All the processing models above allow the application to set a custom error handler that can accommodate the specific error handling needs of the application.

Example:

Custom Error handler that can be used with SAX, DOM, JDOM and JAXP

```
public class TestErrorHandler implements ErrorHandler {

      Collection errors = new ArrayList();
      Collection warnings = new ArrayList();
      Collection fatalErrors = new ArrayList();
      /**
       * @see
org.xml.sax.ErrorHandler#warning(org.xml.sax.SAXParseException)
       */
      public void warning(SAXParseException arg0) throws SAXException {
            warnings.add(arg0.getMessage());
      }
      /**
       * @see
org.xml.sax.ErrorHandler#error(org.xml.sax.SAXParseException)
       */
      public void error(SAXParseException arg0) throws SAXException {

            errors.add(arg0.getMessage());
      }
      /**
       * @see
org.xml.sax.ErrorHandler#fatalError(org.xml.sax.SAXParseException)
       */
      public void fatalError(SAXParseException arg0) throws
SAXException {
```

```
            fatalErrors.add(arg0.getMessage());
      }
}
```

Validating when using JAXB

JAXB uses an XML schema (no DTD) to generate java based object representation of the XML structure, with easy to use marshaling, un-marshaling and validation functions. JAXB still relies on a parser, but only for un-marshaling of the XML documents. A JAXB application can perform structure and content validation with Java classes that it generates from a schema.

Regarding error-reporting JAXB behaves the same way as SAX parsers and reports validation errors as they occur. JAXB allow the application to assign the Unmarshaller object a custom validation error handler. The error handler must implement the ValidationEventHandler interface.

JAXB Validation Error handler

```
public class JAXBErrorHandler implements ValidationEventHandler {

      public boolean handleEvent(ValidationEvent arg0) {
            return false;
      }
}
```

JAXB Validation Example

```
 try {
          JAXBContext jc = JAXBContext.newInstance("test.jaxb.model");
          Unmarshaller u = jc.createUnmarshaller();

          //create the error handler
          JAXBErrorHandler eh= new JAXBErrorHandler();
          u.setEventHandler(eh);

   //Turn validation on while parsing the XML document
          u.setValidating(true);
          FileInputStream fis = new FileInputStream("C://personal.xml");

   Personnel personnel = (Personnel) u.unmarshal(fis);;
          List persons = personnel.getPerson();
          Person p = (Person) persons.get(0);

   //Set salary to wrong value
   p.setSalary(new BigInteger("-100"));

          // create a Validator
          Validator v = jc.createValidator();
          v.setEventHandler(eh);
          // validate the object model prior to marshaling
          boolean valid = v.validateRoot(personnel);

          if(valid){
          //create a Marshaller and marshal to a file
```

```
		Marshaller m = jc.createMarshaller();
		m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
		m.marshal(personnel, System.out);
		}
}
```

Business Rules and Inter-Field Validation

It is common for applications to have the need to apply validation rules to a certain field depending on the result of validation on another field. For example, an application might validate a ZIP code field only if a user specifies a U.S. address. Currently the XML schema rules have no provisions for inter field validation or complex business rules validation. Such validation needs can be accomplished in three ways:

1. Hard code the business rules/validation in the application business methods.
2. Using third party XML schema extensions, like schematron.
3. Using an external rule/validation engine.

The first method requires no external tools and currently being used in our J2EE applications, supplementing the Meta Model validation service, which provides filed level validation only. Further study of option 2 and 3 should be conducted to assess their viability and level of integration with our framework.

7.3. Recommendation

Parsers

All of the parsing methods mentioned above use an XML parser to parse XML documents. The parser we chose must at least support the following features:
- JAXP 1.1 or 1.2 compliant if we want to have a portable schema validation code.
- DOM Level 2 and SAX 2.0 compliant.
- XML Schema compliant.

One of the most commonly used parser in the Java environment is the Xerces2 Java Parser. The JAXP 1.2 reference implementation ships with the Xerces2 parser and the Crimson parser. Our project so far has been using the Xerces2 parser without any problems. As for XSLT processors Apache Xalan-J is the recommended processor.

Oracle Parser Option:
TBD

Parsing Models

All of the above mentioned parsing models have their strength and weaknesses. The functional requirements of the components being built could dictate one method over the other. Here are some of the general guidelines that we can follow:

A. Use JAXP 1.1 or 1.2 compliant parsers when possible, since it provides better portability. JAXP still gives us access to a SAX or DOM parser without losing any functionality. JAXP also provides a transformation interface.
B. SAX is appropriate for high-speed processing of XML because it does not produce a representation of the data in memory and can automatically perform structure validation of XML documents.
C. DOM/JDOM, on the other hand, produces an in-memory data representation, which allows an application to manipulate the contents in memory. DOM and JDOM should be used carefully specially when processing large XML documents, due to their extra demand on memory resources of the system. JDOM claims to use less memory resources than DOM for the same XML document.
D. If you need to map the XML document into java objects, then you can use one of the following two methods:
    i. If the content of a document is dynamic and not well constrained, SAX is more appropriate than DOM or JAXB. The content handler will build the java object graph based on the XML document processed.
    ii. If the content of a document is static, well constrained and described by an XML schema, then JAXB is more appropriate. Since it can map XML data to java objects with far less code than SAX.

Validation:

SAX Parsers Based Validation

Advantages	Disadvantages
Ability to validate XML document without having to convert the XML structure into a tree or java object based representation.	Can't validate in memory or on marshaling
Fast parsing and small memory footprint.	

JAXB Validation

Advantages	Disadvantages
Can validate at the object level during run time, which can be done before marshalling to file/storage	Memory overhead of managing the object model representation of the XML structure, especially if the object model is not needed.
Quick and easy way to provide XML structure to java object binding. Far less code than using SAX.	No open source or commercial implementations yet.
Supports only schema validation	

SAX and JAXP BootStrapping

SAX

There are three ways to bootstrap your SAX parser the first two provide the most flexibility without coupling your code to a specific parser. The third option hard codes the parser name within your code.

1. Use the META-INF/services/org.xl.sax.driver system resource
2. Use the java VM argument -Dorg.xml.sax.driver

When running your application use the following command argument to tell SAX which parser you would like to use

```
java -Dorg.xml.sax.driver=com.example.xml.SAXDriver MySAXApp
```

The table below lists some of the common SAX parser and their main SAX driver

Parser Name	Class Name	Notes
Oracle	oracle.xml.parser.v2.SAXParser	Optionally validates; proprietary
Crimson	org.apache.crimson.parser.XMLReaderImpl	Optionally validates; used in JDK 1.4; Open Source
Xerces	org.apache.xerces.parsers.SAXParser	Optionally validates; Open Source

3. Alternatively, if you don't mind coupling your application to a specific SAX driver, you can use its constructor directly or use the XML XMLReaderFactory.createXMLReader(Driver Name) method. Assume that the SAX driver for your XML parser is named com.example.xml.SAXDriver, the following code snippets illustrate these two options

```
//Using the factory class
String parserName = "com.example.xml.SAXDriver"
XMLReader parser= XMLReaderFactory.createXMLReader(parserName);

//Direct Instantiation
XMLReader xr = new com.example.xml.SAXDriver ();
```

JAXP

These are some of the ways you can bootstrap JAXP to use a specific XML driver:

- Install the JAXP-compliant driver in $JAVA_HOME/jre/lib/ext.
- You can modify the CLASSPATH to include the jar files of the XML driver

- You can also set up $JAVA_HOME/jre/lib/jaxp.properties file and set the following attributes to use the driver you chose:

  ```
  javax.xml.parsers. SAXParserFactory = compliantSAXParserFactory
  javax.xml.parsers. DocumentBuilderFactory =
  compliantDocumentBuilderFactory
  javax.xml.transform. TransformerFactory = compliantTransformerFactory
  ```

- Alternatively, if you don't mind coupling your application to a specific JAXP driver, you can use its constructor directly or use its factory implementation class

8. Appendix 3: eRA and XML Streams

8.1. Introduction

The eRA Competitive Grant Application Project CGAP is design processes and selecting technologies for the electronic receipt of applications. The eRA system will accept XML streams for grant applications and other types of exchange with the grantee institutions. The requirements are not yet defined; however, the following assumptions have been made:

- The solution should **allow binary data** to be conveyed.
- The **transmissions** should be **secured**: a stream may contain confidential information.
- The solution of choice should **favor Open Standards** as opposed to proprietary technologies.
- All data or documents should be grouped together (for a given application), so that **only one transmission** is necessary.
- The solution retained should be **scalable**: a large volume of data could be submitted through XML streaming.
- Applications may be received from a Service Provider that helps Institutions prepare grant applications or manage their grant portfolio.
- Applications may be received through the federal E-GRANTS broker in a standard XML format.
- The submission of a XML stream to NIH should not require using the services of an Application Service Provider (ASP) or *Trusted Broker*. Every external institution may act as its own service provider.

Purpose and Scope

This document describes technological options and contains a high-level architecture of the different prototypes realized. It assumes that the reader is already familiar with the XML specifications and Internet development practices.

This document is intended to answer basic questions related to XML streaming technology for eRA and will evolve as the requirements are refined. This document simply describes the technologies considered as preliminary work to the technical architecture of the CGAP project. It is not intended to be a guide to technologies.

Definitions, Acronyms, and Abbreviations

Refer to the eRA (Enterprise) Glossary for a list of definitions, acronyms, and abbreviations.

References

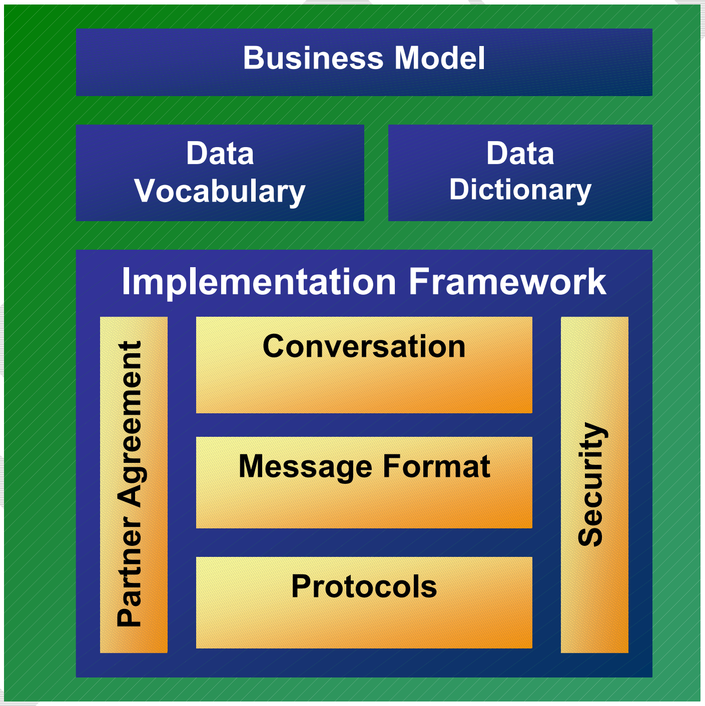
- http://java.sun.com/xml/
- http://www.w3.org/2002/ws/
- http://www.w3.org/2000/xp/Group/
- http://www.w3.org/Signature/
- http://www.w3.org/2001/XKMS/
- http://www.w3.org/Encryption/2001/
- http://otn.oracle.com/tech/xml/content.html

8.2. B2B Overview

The concepts of Business to Business (B2B) or Business to Government information exchanges are used as guiding ideas for the CGAP project. The objective is to achieve a system-to-system flow of information. The following section is an overview of the B2B exchange concepts.

Most B2B systems are based on loosely couples systems where an middle ground system (the exchange) acts as an intermediate system between the trading partners.

The technologies most commonly used to package the transactions are EDI or XML. The concept of the B2B exchanges are not limited to the definition of the standard transaction like the schema or DTD in XML. The following figure illustrates the components that require standardization to establish an efficient B2B implementation.



The following sections describe the components of a B2B exchange.

Business Model

The business model specified how the trading partners will interact in general and for a particular transaction. It specifies the roles, responsibilities and expected activities for each trading partner to accomplish a particular business process. A business transaction follows a process that may require multiple messages to be transmitted and multiple decisions to be made by each trading partner. These decisions are accomplished as business level rules, not at the transmission protocol level. In some of the frameworks, the business model and rules are defined as metadata can be by dynamically interpreted by the trading partners. These advanced concepts of meta-models are designed to establish a common language for the trading partner with enough specificity and rigor to eliminate ambiguity. Such models are still in their infancy and are complex to implement. To support the metadata, a business model repository is needed that can give dynamic access to the rules to the trading partners. While these models are beginning to be formulated, the implementations are complex and only feasible for large commercial products.

The existing exchanges usually have implemented specific business models that are not formally described and are not dynamic. While certainly this formal model approach is more rigorous and shows promise, the lack of implementations does not make it a good candidate for the CGAP project. The RosettaNet standard is probably the more advanced specification for business models but is entirely focused on traditional business transactions and does not include definitions for grant applications

In the CGAP context, it is expected that the business model will be a document describing the process for each transaction type. The notation will use the UML notation. The narrative part of the business process flow will use a Use Case format. It is not anticipated that a specialized standard for business modeling other than UML will be used in the first version of the CGAP system.

Data Vocabulary

The definition of a standard transaction is at the core of the exchange. This definition of a standard requires that the data element have the same meaning for all partners and that the transactions be defined as a standard for the exchange. These data definitions constitute a precise vocabulary for the exchange. The vocabulary communicates the structure and grammatical rules that the message content must adhere to. Given these established rules and definitions the transactions can then be interpreted by each trading partner for processing in their own systems. The metadata of the exchange must satisfy 2 goals:

- Provide human readable documentation
- Provide a schema readable by the systems

The human readable descriptions must describe both the definitions of each data elements and the structure of the transaction. If business rules are embedded in the transaction schema, they must be described for humans. The system-oriented definition must be a completely unambiguous and machine readable definition. It essentially must be the template for the transaction. When XML is selected for encoding, two schema choices are available for use: DTD and XML-schema. The XML-schema is more recent and has more capabilities than the DTD and is more suitable for descriptions of complex transactions.

In the eRA context, the notion of data vocabulary will be extended to the metadata describing the transactions, protocols and partner agreements, in addition to the specific vocabulary used in the XML schema.

Data Dictionary

In the context of CGAP and eRA the exchange data dictionary will be the repository for the metadata for the exchange. It is therefore a more limited interpretation that the classical meaning of a data dictionary that would include the providing of the semantic meaning of the transactions and data elements. It is also a more expanded meaning since it extends to the information needed to make the exchange process work. The data dictionary in CGAP will contain the information necessary for the trading partners to use the transaction templates, the definitions of the data elements used and other metadata needed to drive the exchange such as certain access rights and status of certifications of trading partners.

The data dictionary may also contain list of valid values for some variables or valid ranges for data elements.

Implementation Framework

The following sections describe the components needed to implement the exchange mechanism. The primary problem to be solved is to allow trading partners to create a seamless end-to-end process flows. The business process is integrated and the systems must interoperate. A number of very concrete technical problems need to be resolved to interoperate systems. The implementation framework defines all the technical information needed to implement the information flow between trading partners. The following subsections describe the elements of the implementation framework. Established standards such as ebXML and RosettaNet provide specifications for theses components with various degrees of completeness.

Protocols

The protocol is the layer of message transport on top of the physical connection layer. Logically it is the lowest layer in the framework. All other software relies on a specific protocol. The protocol defines the specific sequence of system events and formats for the transport of a message. Examples of protocols are HTTP, Simple Mail Transport Protocol (SMTP), and FTP. An exchange may support multiple protocols to accommodate multiple trading partners or multiple types of messages. Some B2B standard frameworks also support multiple protocol options.

Message Format

The message itself needs to be formatted in a standard way. The message is usually broken down into the following sections.

- **Header:** The description of the type of message and content. The content of the payload may be represented in a manifest that acts as a table of contents; the header usually carries security information, processing instructions, and routing information.
- **Payload:** The body of the message. One or more sections in a message that contain the business information.
- **Envelope:** The outer wrapper that comprises the payload and the header

The header portion of the message contains the additional information beyond the business information in the payload that enables those payloads to be transported between two services.

The business content in the payload needs to be structured in such a way it can be interpreted by the trading partners. The transaction contained in the payload is encoded using a pre-determined method such as binary, XML, HTML, EDI (positional text), and unstructured text format.

The envelope is a simple container used to bind together the header and the payload in the message body and provide some means to identify the standard to which the message adheres. The most common methods of packaging the message are Multipurpose Internet Mail Extensions (MIME) and XML.

Conversation

The word conversation is used in some standards to define the sequence of events that allows a business transaction to be completed. These events are a series of messages and specific information exchanged such as message receipt acknowledgments or errors. If the standard requires the reliable delivery of messages, the conversation defines the sequence of messages and acknowledgments over the selected protocols. The definition of the conversation allows each participant to check on the transaction state. Conversation is a technical implementation of the transaction delivery and completion. The business model represents business practices. The conversation definition allows complex messages and essentially a small business process flow to be represented in the exchange between trading partners.

Security

Both in commercial and government applications, the information integrity and privacy must be protected through implementation of security measures. The security aspects of the exchange are defined as part of the framework. The security measures address the issues of: encryption, authentication, authorization, non-repudiation.

Partner agreement

The partner agreement defines the structure of the information required to establish an electronic link with a trading partner. The trading partner agreements defined in most standards strive to automate the maintenance of the exchange by using metadata to describe the trading partners and their participation in the exchange. The more options available to the individual partner, such as protocol and message header, the more complex the partner agreement must be to enable users to specify those choices. Standards for partner profiles and agreements have been developed by organizations such as ebXML and UDDI (Universal, Description, Discovery, and Integration).

In the eRA context, the partner agreement will also define roles and responsibilities of the exchange participants. For example, it is likely that some sort of certification process will be required. The agreement will specify under what circumstances a re-certification is necessary. Accessing federal government computers also is subject to certain practices and responsibilities that need to be agreed on.

8.3. Implementation Framework

Transport Protocols

BEEP (or BXXP)

Overview

BXXP is essentially a toolkit that developers can use to create protocols for a range of applications including instant messaging, file transfer, content syndication, network management and metadata exchange. BXXP uses a peer-to-peer architecture. BXXP sets up & maintains a connection over the Internet between User X and User Y to carry data and information back and forth between the two parties. One special feature of a BXXP connection is it can carry multiple simultaneous exchanges of data - knows as channels. In BXXP's architecture, the two users alternate between acting as clients and servers. BXXP uses XML to frame the data images and information exchanged between the two users. At the core of the BXXP framework is a framing mechanism that permits simultaneous and independent exchanges of messages between peers. Messages are arbitrary MIME content, but are usually textual XML.

BXXP runs on top of TCP & acts as an alternative to HTTP or a custom-made data exchange protocol. HTTP was designed to handle the transport of HTML documents and is ideal for Web browsing. HTTP doesn't work well for the transfer of XML data; also it does not support multiple simultaneous exchanges between users. For these types of applications, developers have to create their own special-purpose protocols. Now they can use BXXP to speed that process. Blocks is a new protocol designed to move (XML) data on the Web smoothly and quickly.

BEEP has several development efforts underway that includes libraries for C, Java, and TCL.

Advantages:

BEEP has been designed with Web Services in mind. HTTP, the most popular transport method for Web Services is lacking in some respects, and BEEP addresses these problems. BEEP creates a stateful connection between the client and server, which minimizes the risk of errors, and can be more efficient as only one transport handshake would need to take place for a transaction.

Disadvantages:

BEEP is still in its infancy. There are only a handful of libraries for a few programming languages. This could limit the technologies that Client may implement. While BEEP’s stateful communication has many benefits, this can incur additional load for the application, as the transport is a much heavier layer.

Conclusion

BEEP is a good choice for Web Services. Its stateful nature is ideal for handling two-way Web Service communication. However, BEEP does not have a widely installed base, or a robust user community. Implementing this technology could be an extra burden for the client and potentially limit their choice of technology.

Reference

http://www.beepcore.org/

HTTP(S)

Overview

Short for HyperText Transfer Protocol, the underlying protocol used by the Web. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. HTTP is by far the most widely used protocol on the Web. Of the widely accepted protocols on the Internet, it is the one most often used by Web Services. Based on a stateless request – response system it is very lightweight. Because of its widespread implementation, HTTP has a variety of servers made for any server platform. It is also relatively simple to install and maintain. HTTP also has the ability to be piped across SSL. This ensures that the transmission is secure.

Advantages:

HTTP is a building block of the Web. This gives it a wide range of support, installation and configuration options. It can be tunneled through SSL, which is convenient for securing the transmission.

Disadvantages:

HTTP does not guarantee delivery. For instance, a transmission could be interrupted at the source in the middle of a transmission and not generate an error. Also, depending on the application, its stateless nature could be a hindrance.

HTTP was primarily designed to handle serving static documents to a client. However, since then HTTP has been adapted to serve everything from dynamic content to streaming media. Because of this HTTP has been overloaded and could potentially cause a security concern.

Conclusion

Most Web Services examples use HTTP as the transport protocol. It is extremely simple to set up and get running. HTTP is well understood and omnipresent for web applications. It should be strongly considered for all implementations. However, HTTP's statelessness can create an additional burden on the Web Service implementation. The application is now required to maintain any state that it would need. The limitations of reliability can be overcome for the CGAP processes by verifying that the transfer has been completed based on the ticket information.

Reference

HTTP Pocket Reference, O'Reilly.

SMTP

Overview

Short for Simple Mail Transfer Protocol, a protocol for sending e-mail messages between servers. SMTP is one of the earliest transport protocols on the Internet. It is tied closely with MIME, which is a format for encoding information over the protocol. SMTP and mime allow for multiple documents in the payload.

Like HTTP, SMTP has a very large install base. It is also one of the most widely used protocols on the Internet. SMTP has also been adopted for use in Web Services. It is the ideal protocol for handling message-based systems.

Advantages:

SMTP is natively message based, as it was designed for sending text messages with attachments over the Internet. Because of this it works well for Message Based web services.

Disadvantages:

Cannot be piped over SSL for security, but there may be a way to encrypt the MIME message, which would be sufficient. Not effective for handling large amounts of data per transaction.

Conclusion

SMTP is an excellent protocol for handling message-based Web services. This could be an excellent way of interfacing with a queue. Also, the response, which would be another SMTP message, could contain both a SOAP and HTML attachment. This could allow the CGAP exchange to send responses back to the client with a single API.

Reference
http://www.pocketsoap.com/specs/smtpbinding/

FTP

Overview

File Transfer Protocol (FTP) is the preferred method for sending files over the Internet. It differs from HTTP, which also sends files, in that it allows the client to browse the server's file system. HTTP only allows requests for a particular file, and relies on a referencing system (hyperlinks) for document navigation. Also, FTP uses a smaller packet size then HTTP, so is more efficient in transferring large amounts of data.

FTP uses a simple username/password challenge for security. It also has the ability to allow for anonymous access for distributing files to an unlimited number of clients. However, FTP over SSL has not caught on in the industry; therefore the file transmission is left unencrypted.

Advantages:

FTP is an excellent protocol for transferring large documents across the Internet. Due to its large installed base, there is a lot of information on this topic from general information to security.

Disadvantages:

FTP does not tunnel over SSL, so an encrypted transmission is not an option. FTP is singled out by the FBI as one of the more vulnerable protocols.

Conclusion

It naturally supports file transfer. However, security requirements have to be carefully evaluated when this protocol is used.

References

http://www.bisonftp.com/RFCS/rfc959.txt

SSH/SCP

Overview

Secure Copy is an adaptation of the UNIX CP utility with encryption and is used extensively in the UNIX world.

SSH is can be seen as a secure version of telnet, whereas SCP can be considered as a secure FTP. When using those protocols, all traffic is encrypted and prevents network-level attacks.

Advantages:

The protocol is easy to implement using tools.

It does not require tunneling or other mechanisms to guarantee security.

Disadvantages:

Windows client implementation is home made.

Conclusion

SCP should be considered as an option for transferring files.

Jabber

Overview

Jabber is an open XML protocol for the real-time exchange of messages and presence between any two points on the Internet. The first application of Jabber technology is an asynchronous, extensible instant messaging platform, and an IM network that offers functionality similar to legacy IM systems such as AIM, ICQ, MSN, and Yahoo. However, Jabber offers several advantages over legacy IM systems:

- Open—the Jabber protocol is free, open, public, and easily understandable, and multiple open-source implementations exist for Jabber servers, clients, and development libraries.
- Extensible—using the power of XML namespaces, anyone can extend the Jabber protocol for custom functionality; to maintain interoperability; common extensions are managed by the Jabber Software Foundation.
- Decentralized—anyone can run their own Jabber server, enabling individuals and organizations to take control of their IM experience.
- Secure—Any Jabber server may be isolated from the public Jabber network, many server implementations use SSL for client-server communications, and numerous clients support PGP/GPG for end-to-end encryption; more robust security using SASL and session keys is under development.

Advantages:

Excellent protocol for performing message based XML communications. Ideal for small messages such as: Instant Messaging, stock quotes, and news services. Jabber also integrates well with security schemes.

Disadvantages:

Jabber is not designed with B2B in mind. Originally developed as an Instant Messaging service, Jabber is quick with small payloads, but gets bogged down with large transfers. Jabber is an excellent alternative to SMTP as a messaging protocol.

Conclusion

Jabber is a technically superior replacement for SMTP on the messaging side of Web Services. However, Jabber cannot match the install base and amount of resources available with SMTP. It may not be ideal for prototyping our application, but should be kept in mind for the long run.

Reference

http://www.jabber.org/

Reliable HTTP (HTTPR)

Overview

HTTPR is a protocol for the reliable transport of messages from one application program to another over the Internet, even in the presence of failures either of the network or the agents on either end. It is layered on top of HTTP. Specifically, HTTPR defines how metadata and application messages are encapsulated within the payload of HTTP requests and responses. HTTPR also provides protocol rules that make it possible to ensure that each message is delivered to its destination application exactly once or is reliably reported as undelivered.

Messaging agents use the HTTPR protocol and some persistent storage capability to provide reliable messaging for application programs. This specification of HTTPR does not include the design of a messaging agent, nor does it say what storage mechanisms should be used by a messaging agent; it does specify what state information needs to be stored safely and when to store it, for a messaging agent to provide reliable delivery using HTTPR.

Advantages:

HTTPR picks up where HTTP left off. HTTP was designed for serving static pages, while HTTPR is an extension to HTTP with support for applications.

Disadvantages:

HTTPR is not widely supported. Developed solely by IBM, it does not have very much community support. As a result, there are limited APIs and servers for this technology.

Conclusion

While HTTPR is a capable protocol, and could be preferred to HTTP, it does not have a widely installed base and has little community support.

References

http://www-106.ibm.com/developerworks/webservices/library/ws-phtt/
http://www-106.ibm.com/developerworks/webservices/library/ws-httprspec/

8.4. XML Streams and Binary Data

An application for an NIH grant consists of a complex set of information, and may include pictures, PDF and/or Word documents. This section details the mechanisms available to convey binary data with an XML stream.

8.4.1. Embedding Binary Data in XML

Basic Approach

The basic approach would be to embed the byte values of binary data between tags within the XML document. However, this does not work because the binary data may contain bytes with a special value like 0x3C (less than) or 0x26 (ampersand) that has a specific meaning according to the XML 1.0 specifications. The receiving XML processor would attempt to interpret the byte sequence following the UTF-8 or UTF-16 encodings. This most likely causes the parser to encounter invalid sequences and fail.

Using CDATA Sections

The naïve answer to the problem stated above is to use CDATA constructs. This approach only makes the trouble less likely, rather than removing it. It is possible that the CDATA terminator string "*]]>*" appears in the binary data and corrupts the rest of the XML document.

Using Hexadecimal Representation

To avoid the misinterpretation of a random sequence of bytes, binary data must therefore be encoded into the valid character set before being embedded into an XML document. It implies a processing cost for encoding and then decoding the data.

A direct approach consists of converting each binary data byte into its two character hexadecimal representation. The 256 possible byte values can be encoded using a valid character set [0-9, A-F].

For instance, the byte `00111100` is represented `0x3C` in (java) hexadecimal notation. This byte corresponds to the character < (less than) and would have been misinterpreted by an XML parser if embedded between XML tags. The encoding of this byte would be the string "`3C`".

The drawback of this technique is that the size of the binary data is doubled. For each byte in the original binary file, two characters appear in the resulting XML document.

Base64 Encoding

Base-64 encoding is another technique to transform binary data to a valid character set. It is a well-known technique that has been employed for a long time and many implementations are available for free over the Internet. The encoding algorithm processes a byte stream in 3-byte sequence. Each 3-byte sequence parcels into four 6-bit data units. Each 6-bit unit then encodes into the character stream as the corresponding character from the character set [A-Z, a-z, 0-9, +, /]. RFC 2045[1] describes the algorithm in more detail.

In short, this approach encodes 3 data bytes using four characters. The encoded document is therefore 33% larger than the original document.

The EDI-X12 (including the X-12 194 transaction set for grant applications) standard uses Base64 transformations for transfer encoding.

Huffman Encoding

For some binary data, a histogram of each byte value's occurrence frequency within the data would show an uneven distribution, where some bytes are used very frequently, while others are used rarely or not at all. Huffman encoding uses this statistical property to reduce the average code length. The most frequently used bytes are represented using single characters or short character sequences, whereas the least frequently used correspond to longer character sequences.

When most data sets share similar statistical properties, a fixed mapping of the 256 possible byte values can be defined for all the exchanges. Alternatively, it is possible to calculate the optimal mapping and embed it within each transmission; this is particularly useful for large data files, where encoding efficiency is most critical.

This approach is highly effective for binary data where the distribution is biased toward a relatively small byte value subset; the resulting encoded document could be almost equal in size to the original binary document. It is not as effective for cases where the distribution is fairly uniform; the size overhead depends on the mapping utilized.

Recommendation for Embedding Binary Data in XML

The **preferred solution is Base-64** encoding. It is a standard algorithm and the size overhead of 33% of the encoded data is acceptable.

- The *hexadecimal representation* approach does not provide an additional benefit and the size overhead is higher.
- Huffman encoding is not as widely known as Base64. A custom implementation would have been required; with the inherent risks associated to it. Finally, the advantages of this approach are not obvious since NIH does not control the data

[1] RFC2045 is the Request For Comment for the MIME standard - part one. It is available at http://www.ietf.org/rfc/rfc2045.txt.

sets and therefore can not guarantee the unevenness of the byte values distribution.

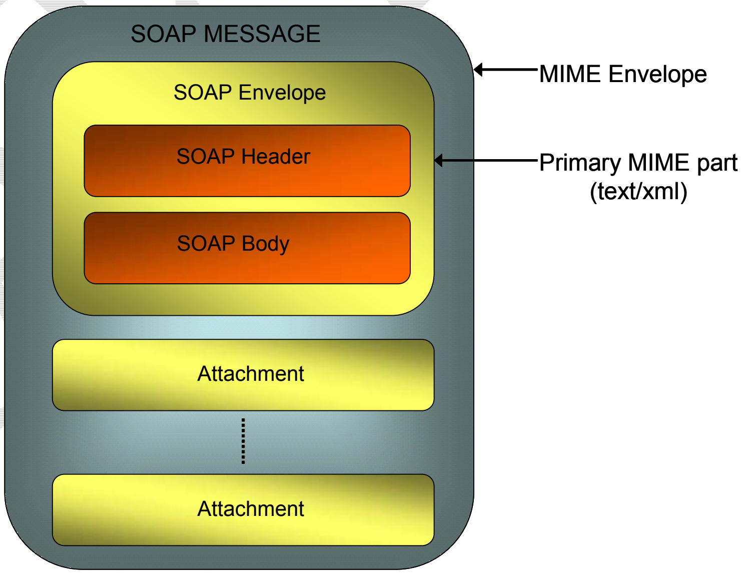
Note: Browsers can not display Base64-encoded data in a *readable* format. Preprocessing is required to decode the binary data before displaying them.

8.4.2. Keeping Binary Data Outside of XML

Simple Object Access Protocol (SOAP)

SOAP defines the `href` attribute that allows pulling the embedded content out of the XML container, and replacing it with a link. Unfortunately, most documents submitted to NIH through the XML stream will not be available as a URL; therefore, this approach will not be appropriate.

SOAP with Attachments (SwA)



SOAP Messages with Attachments (SwA) is simple: the first part of the multipart MIME (Multipurpose Internet Mail Extension) message is the XML SOAP document; the subsequent parts contain the attached data.

HTTP forms can be sent using MIME multipart; this means that all web servers should already have the proper MIME machinery built in.

The advantage of this implementation is that it is very simple and easy to implement.

A drawback to SwA is that it cannot handle data streaming. This means that the client would have to buffer the entire attachment (for instance a multi-media data stream) before processing it.

BEEP

BEEP stands for *Block Extensible Exchange Protocol*. Its primary inventor is Marshall Rose. Like SOAP, BEEP is described in a transport-neutral manner and is defined in RFC 3080[2].

[2] The memo is available at http://www.ietf.org/rfc/rfc3080.txt

BEEP is a "generic application protocol kernel for connection-oriented, asynchronous interactions". In other words, it's like an application level transport protocol. One of the most interesting implications of BEEP's design is that it supports multiplexing – multiple channels communicating over a single transport stream.

Simple attachments are embedded in a multipart MIME, following the same convention as described above for SwA.

DIME

DIME stands for *Direct Internet Message Encapsulation*. Its primary inventors are IBM and Microsoft. Like SwA or MIME, DIME is a message format, as opposed to a network protocol like HTTP.

DIME has a binary format: fields have fixed size, numbers often have fixed size, bytes are written in a specified order, and so on. A DIME message consists of a series of one or more records joined together to make a single application message. Records are not numbered; their position is implied by their position in the data stream. Therefore, DIME requires a stream protocol like TCP and is unsuitable for UDP.

Recommendation

The preferred solution is SOAP with Attachments (SwA). The XML streams submitted to NIH do not require (yet?) embedding data stream (like streamed audio or video). Moreover, the mechanisms of SwA are simple to implement. Finally DIME and BEEP may not be as mature and widespread as the technology which SwA is based upon.

Recommendation for Binary Data Transmission

The recommendation for binary data transmission is to use **SOAP with Attachments (SwA)** – described in the section 0. It conforms to standards that the Internet community is familiar with.

In addition, embedding binary data as a string in XML could have implications in terms of scalability. The use of common XML parsing techniques would require the server to store (temporarily) in memory the whole binary document (base64-encoded and therefore 33% larger) and then decode it. For SwA, MIME toolkits allow treating an attachment as a stream, reducing the memory footprint.

Cross-referencing of attachments with SwA could be achieved using a naming convention comparable to the one employed for email messaging. A *Message-ID* should be generated for each transmission. The main property of this *id* is that it is globally unique. It also identifies the organization or individual who originated the transmission (this information however should not be trusted nor considered reliable). The name of each part of the MIME message consists of the *message-id* prefixed by the part number. An example of this mechanism is illustrated below:

```
Content-Type: multipart/related; type=text/xml;
  boundary="xXxXxXx";
  start="<start-AA11234455.22@www.mysbir.com>"

Here is the latest Research Accomplishments for
the application #XYZ.

--xXxXxXx
Content-Type: text/xml; charset="UTF-8"
Content-ID: <start-AA11234455.22@www.mysbir.com>

<SOAP-ENV:Envelope>
 <SOAP-ENV:Body>
  <tns:AppNumber>XYZ</tns:AppNumber>
  <tns:ActionType>Update</tns:ActionType>
  <tns:ItemType>Research Accomplishment</tns:ItemType>
  <tns:ItemValue                             href="cid:part1-
AA11234455.22@www.mysbir.com"/>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

--xXxXxXx
Content-Type: application/msword
Content-Transfer-Encoding: 8bit
Content-ID: <part1-AA11234455.22@www.mysbir.com>

..... [file Research_ABC_XYZ.doc]

--xXxXxXx--
```

Further investigation and prototyping is however necessary before finalizing this recommendation.

8.5. Packaging protocol

SOAP/XML

Overview

SOAP is a simple and lightweight protocol for exchanging structured and typed information. It has been designed with distributed computing in mind. SOAP is based on XML Schema.

Advantages:

Lightweight, should not incur too much processing or bandwidth overhead. It is based on XML Schema, so data is structured and typed. This should lead to easy integration with Java Objects. It is the most widely supported Packaging protocol, as it is backed by large organizations such as IBM, Sun, and Microsoft, and a W3C Standard.

Conclusion

SOAP is widely supported and is considered a standard. There is a wealth of books on SOAP and on using SOAP in Web Services. With such a broad scope of documentation and community support SOAP is an excellent candidate as a packaging protocol for our application.

References

www.w3c.org
Programming Web Services with SOAP, O'reilly
Web Services Essentials, O'reilly

XML-RPC

Overview

Precursor to SOAP. Uses DTDs for structured data, but has little support for typing. Standardized, but not officially supported by a standards body (such as the W3C).

Advantages:

Simple, as it relies on DTDs.
Has a lot of community support.

Disadvantages:

Does not have strong data typing.
Not officially supported as a standard.

Conclusion

XML-RPC should be viewed as SOAP in its infancy. All of the issues surrounding XMP-RPC have been addressed with SOAP.

References
www.xmlrpc.org
Web Services Essentials, O'reilly

Jabber

Overview

Jabber is intended as an extremely lightweight message protocol. It is very well suited for P2P applications. Standardized, but not officially supported by a standards body (such as the W3C).

Advantages:

Jabber is extremely lightweight.
Also, it has its own transport protocol, so would integrate nicely with that.

Disadvantages:

Designed for P2P not B2B.
Not officially supported as a standard

Conclusion

While Jabber is an excellent Packaging and Transport Protocol, it is not suited for B2B applications with large amounts of data. Therefore, not recommended for our application.

References

www.jabber.org
Programming Web Services with SOAP, O'reilly.

DIME

Overview

DIME is a lightweight document encapsulation format. Not necessarily suited for structured data transport, but an excellent medium for transporting multiple documents of arbitrary type for processing. DIME has been developed jointly by IBM and Microsoft.

Advantages:

Lightweight
Efficiently encapsulates multiple documents.

Disadvantages:

There is potential for Microsoft distributing a broken distribution for an unfair advantage. This could create version incompatibilities.

Conclusion

Since we will need attachments for our applications, DIME may be an excellent way to encapsulate them in a single entity for transport. This would make the data retrieval from the client less error prone. We would still have the ability to use another Packaging protocol (like SOAP) as an internal document for data processing.

References

Programming Web Services with SOAP, O'Reilly.

8.6. XML Streams and Security

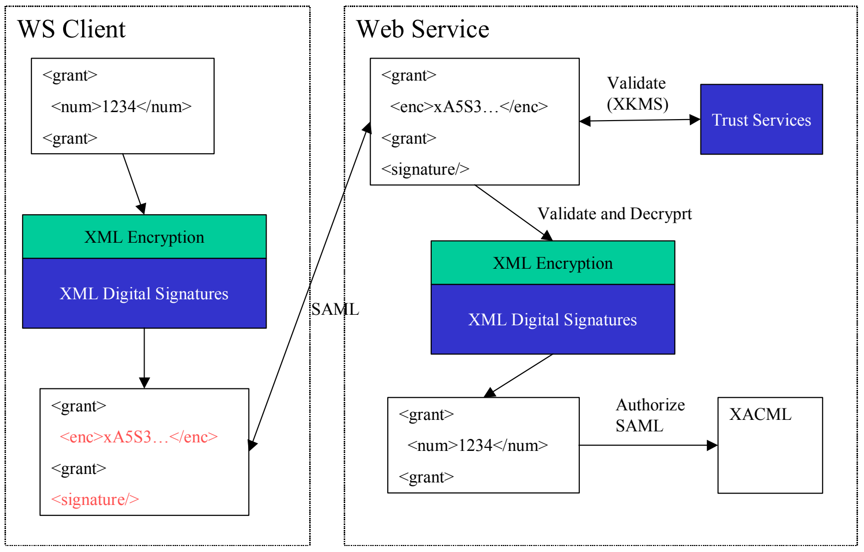
NIH will receive XML streams from the Internet; therefore the transmissions are exposed to hackers and other pirates. The most important factors to guarantee security are the following features:

- **Authentication**: NIH must be able to identify the source of a stream. Also, the originator of the stream must be able to identify the recipient of the data as NIH.
- **Confidentiality**: the transmission of the data must be encrypted to prevent eavesdropping.
- **Integrity**: a mechanism that guarantees the non-alteration of the data during the transmission must be implemented.
- **Non-Repudiation** is not a significant aspect for now. Non-repudiation is a way to guarantee that the sender of a message cannot later deny having sent the message and that the recipient cannot deny having received the message.

Security imposes additional restrictions on the technology. Traditionally security adds an additional layer of complexity. This complexity can increase overhead in the software development life cycle. For instance, it is not enough for the client to produce a document; they must also encrypt it and digitally sign it. Once this is done, there must be a system in place for decrypting it and validating and managing the signatures. The level of security needs to be determined for the project before a complete security recommendation can be made.

Proposed Standards for XML Security

The most popular *proposed standards* for XML security are presented in this section. Most of these standards work in tandem to complete the security landscape. For instance, XML Encryption only encrypts the data that handles confidentiality, while XML Signature only guarantees who you are communicating with. The following diagram illustrates how these technologies work together to form a secure Web Service.



XML Encryption (Xenc)

Overview

This standard has been proposed by W3C and IETF. It allows encryption of selected sections, with the idea that only sensitive information needs to be protected. Encrypting portions of a document with different keys would allow the distribution of the same XML document to various recipients, but the recipients would only be able to decrypt the parts relevant to them. Also, because the data itself is encrypted but not the file, it can still be recognized by XML parsers and handled accordingly.

XML Encryption is already a W3C standard and already has several supporting APIs. Due to its standardization and selective encryption, this security scheme would be ideal the Confidentiality and Integrity of the documents transmitted.

XML Encryption is still a relatively new specification. While there are several commercially available toolkits, they are only available for a limited set of technologies. This could potentially limit the technology choices of the client.
XML Encryption has potential to handle the Confidentiality and Integrity of the data being submitted.

References

http://www.w3.org/Encryption/2002/02-xenc-interop
http://phaos.com/products/xml/xml.html

http://www.baltimore.com/keytools/xml/

XML Signature

Overview

This standard has been proposed by W3C and IETF. XML signatures are closely related to encryption. Similar in concept to security certificate signatures, XML signatures are used to ensure that the data sent is the same as the data received.

To help compensate for typographical variations from file systems and parsers, XML signatures depend on a concept *canonicalization*.

XML Signature is a recommendation from the W3C and already has several Java APIs. This protocol works very well with XML Encryption to provide privacy, authenticity, integrity and non-repudiation.

This is an excellent way to handle authenticity, integrity and non-repudiation within Web services.

References

http://www.w3.org/Signature/

XML Key Management Specification (XKMS)

Overview

This standard has been proposed by W3C and IETF. XKMS defines a way to distribute and register the public keys used by the XML Signature specification. XKMS consists of two parts:

- XML Key Registration Service Specification (X-KRSS): to register public keys.
- XML Key Information Service Specification (X-KISS): to resolve the keys provided in an XML signature.

XKMS is a foundational specification for secure Web services, enabling Web services to register and manage cryptographic keys used for digital signatures and encryption. When combined with WS-Security, XKMS makes it easier than ever for developers to deploy enterprise applications in the form of secure Web services available to business partners beyond the firewall.

References

http://www.w3.org/TR/xkms/

eXtensible Access Control Markup Language (XACML)

Overview

This standard has been proposed by OASIS. XACML is used in conjunction with SAML (presented below). It provides a means for standardizing access control decisions for XML documents. It is used to authorize requested access to a resource, whether it is an entire document, multiple documents or a portion of a document. As opposed to XML encryption, access control information is stored in a physically separate repository that is referenced when a request is made.

XACML is excellent for addressing fine grained control of authorized activities, the effect of characteristics of the access requestor, the protocol over which the request is made, authorization based on classes of activities, and content introspection (i.e., authorization based on both the requestor and potentially attribute values within the target where the values of the attributes may not be known to the policy writer).

References

http://www.oasis-open.org/committees/xacml/

Security Assertion Markup Language (SAML)

Overview

This standard has been proposed by OASIS. A SAML request is sent via SOAP over HTTP. It contains information such as authentication username and password, or other details about the individual making the request. This information is then delivered to an application designed to process it with the intended goal of using XACML to allow or deny access to an XML resource.

SAML is used for learning about the business credentials and works well with XKMS.

References

http://www.oasis-open.org/committees/security/

WS-Security and WS-License

Overview

They are two proposals from Microsoft defining how to carry authentication, encryption, and digital signatures within a SOAP Envelope. WS-Security joins several security standards currently being developed within OASIS. Other specifications include SAML for authentication and authorization, XACML for access control, XrML for rights management, SPML for exchanging provisioning information, and XCBF for describing biometrics data. It has recently been proposed to Oasis.

References

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/wssecurspecindex.asp

SSL Approach

Instead of providing security at the data level, as with the NIH eRA application, securing the transmission itself satisfies all the requirements defined earlier in this chapter. SSL is widely adopted for Internet applications and most application/web servers propose toolkits or features to handle this protocol.

Securing the transmission using SSL

Secure Socket Layer (SSL) is a protocol residing in the network layer on top of TCP-IP. Tunneling HTTP through SSL is a regular practice also known as HTTPS.

SSL uses public key cryptography, a technique that uses a pair of asymmetric keys for encryption and decryption. Each pair of keys consists of a public key and a private key. The public key is made public by distributing it widely. The private key is never distributed; it is always kept secret.

Data that is encrypted with the public key can be decrypted only with the private key. Conversely, data encrypted with the private key can be decrypted only with the public key. This asymmetry is the property that makes public key cryptography so useful.

Confidentiality and data integrity are guaranteed for SSL communication. Data can be encrypted using 40-bit or 128-bit encryption (US only). Authentication is discussed in the next section.

SSL: One-Way Authentication vs. Two-Way Authentication

One-Way Authentication

One-way authentication, also known as server authentication, is the most common configuration. The server presents a certificate to the client, but the client is not required to submit any certificate. This configuration guarantees the identity of the server with which the client communicates.

This configuration requires the server to obtain and install an X509 certificate. In most cases, the certificate must be issued by a Certificate Authority (CA).

Authentication of the client is not performed at the network (SSL) layer, but at the application layer. For instance, a simple username/password scheme would satisfy the authentication requirements for the client. This level of security is sufficient for almost all B2B E-Commerce applications; Federal legislations such as HIPAA[3] consider this type of client authentication adequate.

Two-Way Authentication

Two-way authentication requires a digital certificate to be installed on the client side. This solution guarantees a higher level of security and attests the identity of the client. In the context of eRA, a client could obtain a certificate in two ways:

- ***From a Certificate Authority***
  For a fee, a CA can deliver a certificate that guarantees the identity of the client. This certificate is valid for a limited time (usually one year); it is recognized by all systems and can be used universally.

- ***From a certificate generator maintained by NIH***
  In this case, NIH has to implement a formal process to generate and deliver a certificate—a weak process would compromise the validity of those certificates. As NIH is not a trusted CA, only NIH eRA-specific applications will recognize and accept this certificate.

Recommendation

SSL is the recommended solution; it provides a robust and proven mechanism to secure XML streams. Also, installing certificates on the client side is not necessary, at least for the first stages of the project.

Early solutions are available for XML security standards but these implementations have not reached a sufficient level of maturity to be used in Enterprise applications. In addition, some specifications of the standards are not yet finalized.

Requirements will drive the final choice for the implementation of Security. The cost of implementation often increases significantly as those requirements become sophisticated.

[3] HIPAA is the Health Insurance Portability & Accountability Act of 1996. It defines, among other topics, the security standards protecting the confidentiality and integrity of "individually identifiable health information". This type of authentication is accepted if implemented in parallel with a specific password expiration policy.

8.7. B2B Standards

ebXML

Overview

The ebXML (Electronic Business using eXtensible Markup Language) standard resulted from a combined effort of the UN (UN/CEFACT) and OASIS (the Organization for Advancement of Structured Information Standards). The organization was formed to provide an open XML-based infrastructure to enable the global use of electronic business information. ebXML currently works on all aspects of a standard, including an implementation framework, vocabularies, data dictionaries, and processes. ebXML is a modular suite of specifications that enables enterprises of any size and in any geographical location to conduct business over the Internet.

The mission of ebXML is to "*provide an open XML-based infrastructure enabling the global use of electronics business information in an interoperable, secure and consistent manner by all parties*".

The key components of ebXML are the Collaboration Protocol Profile (CPP), the Collaboration Protocol Agreement (CPA), the Business Process and Information Modeling, the Core Components, the Messaging and the Registry/Repository.

Collaboration Protocol Profile (CPP)

A CPP describes a company's offerings in a standard, portable way. Specifically, it describes the message-exchange capabilities and business collaborations that a company supports. It also describes the company's business processes, including how partners interact with this company. An interesting facet of a CPP is that a business collaboration includes both sides of a two-party B2B transaction. For example, in a buyer-seller situation, the CPP would describe not only the selling process and semantics of the seller, but also the buying process and semantics of the buyer.

Collaboration Protocol Agreement (CPA)

A CPA describes the exact requirements and mechanisms for the transactions that two companies perform with each other. It is formed from a manually or automatically derived intersection of their CPP's, which has been reviewed and agreed upon by both sides. This CPA becomes a contract between the two parties and specifies the "rules of engagement" for a particular collaboration.

Business Process and Information Modeling

ebXML also includes specifications for describing a business process in XML. This can include transactions, document flow, binary collaborations, data encapsulation formats, and more. These specifications are used by authors when constructing CPP's and are also used to describe and share business processes or information formats.

Core Components

Another crucial part of the ebXML standard is a set of ebXML schemas, also called components. These schemas contain formats for business data, such as dates, taxation amounts, account owner, exchange contract, and more. They are specific to business constructs and entities, but not aligned to any particular vertical industry.

Messaging

EbXML messaging is a format that encapsulates a message with all the related message-oriented middleware semantic (e.g. asynchronous/synchronous, reliability options). In particular, an ebXML message represents the visible part of the execution of a CPA, and has features specified to enforce the "Rules of engagement" specified therein. EbXML messaging is built on top of SOAP-encapsulated message-passing invocations (as opposed to RPC-style invocations). It extends the SOAP protocol by adding layered frameworks that support attachment, security and reliable delivery. The Message Services Specification can be found at http://www.ebxml.org/specs/ebMS2.pdf

Registry/Repository

The ebXML registry/repository is a service that stores CPP's, CPA's, ebXML core components and any other ebXML documents or fragments including WSDL documents, JAR files with Java code and even audio and video etc. It contains powerful query abilities to allow users to search for relevant components and potential business partners. The JAXR API can also be used to access ebXML registries. A supplemental report offers a way to locate ebXML registries using UDDI.

A key concept here is while UDDI provides a universal singleton source of references to web service information, an ebXML Registry is a local container for the actual information itself. ebXML provides for publishing and discovery of almost any type of artifacts including CPP, Schemas, commonly used XML components, as well as web services. Thus in a wide-area web service discovery scenario, a potential business partner would first search for a service in UDDI, which would contain a reference to a CPP or other documentation that is actually stored in an ebXML registry. The potential business partner then uses the information in UDDI to access the ebXML Registry where they subsequently discover the CPP for a business partner. They then use the CPP to initiate a partnership agreement for subsequent B2B transactions with the partner.

Advantages:

- Addresses all possible business and technical B2B issues

- Is becoming a de facto industry standard.
- Its Messaging Services Specification, which integrates SOAP 1.1, is robust and comprehensive (RosettaNet, OAG and HL7 use this specification)
- Is an open standard, meaning no individual company owns or controls their development and they are freely available to all

Disadvantages:

- Very complex to implement.
- ebXML is a young technology. Some aspects need improvement, this means that it may change
- Does not define validation mechanisms for XML (does not refer to DTD, Schema or XSL files) for a vertical industry.

Conclusion

ebXML may be hard to put into operation in its entirety and it may be more than eRA needs. However it does provide a recognized and well-documented framework to work with. Some components are more mature than others. For example the Messaging Services Specification, which offers recommendations for message packaging, reliable messaging, error handling, and security, provides a strong foundation for B2B exchanges.

The CGAP team recommends using this framework as a guideline for implementation of the eRA exchange. However if the standard is too complex for the size of the problem eRA is trying to solve; only the appropriate subsets of the standards will be implemented.

Reference

http://www.ebxml.org

RosettaNet

Overview

RosettaNet is a non-profit consortium working to create, implement and promote open e-business process standards. Developed by means of an industry-wide partnership, RosettaNet standards address the Information Technology (IT), Electronic Components (EC) and Semiconductor Manufacturing (SM) supply chain, including manufacturers, distributors, resellers, shippers and end users. It integrates data and processes, utilizing XML-based messaging and internet standards.

The mission of RosettaNet is to drive collaborative development and rapid deployment of internet-based business standards, creating a common language and open e-business processes that provide measurable benefits and are vital to the evolution of the global, high-technology trading network

RosettaNet Standards:

- **PIPs:** RosettaNet Partner Interface Processes (PIPs) define business processes between trading partners.
  In short, a PIP specifies the structure and format of the business document payload for a particular business activity; it also specifies the exchange protocol between trading partners for that activity. It consists of the activities, decisions, and roles for each trading partner involved in a particular business activity.

- **PIP Directory:** The PIP Directory provides you with faster access to the PIP information you are seeking.
  There are eight high-level categories of business functions of PIP, called clusters. Each cluster is then broken down into segments. A segment may contain multiple PIPs. The PIP specification uses the open-EDI Reference Model (ISO/IEC 14662 concepts) and provides multiple views:
  - Business Operational View (BOV): for the business aspects of business transactions
  - Functional Service View (FSV): for technical aspects of business transactions
  - Implementation Framework View (IFV): for RosettaNet implementation aspects.

- **Dictionaries:** RosettaNet dictionaries provide a common set of properties for PIPs. Their purpose is to ensure the consistency of information exchanged between trading partners when executing PIPs and to capture semantics. RosettaNet Technical Dictionaries provide properties for defining products. The RosettaNet Business Dictionary designates the properties used in basic business activities.

- **RosettaNet Implementation Framework (RNIF):** It specifies how to transport RosettaNet Objects and Business Messages between trading partners' network applications. The RNIF specifies an XML-based and MIME-based "envelope" in which to put the business document payload. RNIF Core Specification is the packaging, routing, and transport of all PIP messages and business signals.

- **Product & Partner Codes:** Product and partner codes in RosettaNet standards expedite the alignment of business processes between trading partners. The codes are defined are:
  - Partner Identification: DUNS (Data Universal Numbering System) are used to uniquely define a trading partner and its location
  - Product Identification:
    - Product Number: a GTIN (Global Trade Item Number) is a unique product identifier
    - Product Classification: The United Nations Standard Products and Services Code (UN/SPSC) is a classification convention that is used to numerically identify products and services.

- Product Description (dictionaries)

The basic techniques utilized by RosettaNet are: XML files, DTD validation, HTTP, HTTPS, SMTP protocols, MIME packaging, signature and encryption, and unique identifiers.

Advantages:

- One of strength of RosettaNet is in developing vertical business process standards in various business domain areas. Their PIPs are very complete and well thought out.
- The architecture is mature
- Many tools support RosettaNet

Disadvantages:

- PIPs are not easy to read and process. RosettaNet is not suited for small companies
- The enterprise readiness effort for RosettaNet implementations is often underestimated.
- There is no PIP dedicated to the *grant industry*.
- Does not define registries for partner discovery

Conclusion

RosettaNet is a good candidate but would eliminate small companies from the picture. It appears as a compromise between EDI and SOAP. It is probably the most mature and complete standard defined for B2B. However, the standard seems inflexible and does not address the grant application process.

Reference
http://www.rosettanet.org/

xCBL - UBL

Overview

The Common Business Library (xCBL) is a set of XML building blocks and a document framework that allows the creation of reusable XML documents for e-business. Although XML provides a self-describing document, stronger data typing and validation for e-business transactions is needed. In addition, there is a proliferation of industry-specific elements that would potentially lead to problems. xCBL is a combination of the leading XML industry initiatives and most common cross-industry XML elements. Fundamental documents, such as orders and invoices, and elements, such as country codes and currencies, have been captured in xCBL. Since the applications are heterogeneous, integration would have to take place at a semantic level where components communicate through a common language, vocabulary, and business concepts. xCBL is not a single standard but a collection of common business elements that underlie all EDI and Internet protocols. xCBL has been developed after EDI semantics, such as EDIFACT. Its reusable components should speed the implementation and facilitate interoperation by providing a common semantic framework. xCBL essentially serves as the *mother code*, providing one language that all participants in eCommerce can understand. Application profiles allow businesses to support the same set of fields for use within particular areas of application. Support for the W3C Schema is provided with version 3.5 of xCBL.

The OASIS Universal Business Language (UBL) will not replace xCBL in the short term; it is just starting down its development path. It will be some time before UBL contains the depth and breadth of components and documents that are in the current version of xCBL. The purpose of UBL is to quickly develop a standard XML business library by taking an already existing library (xCBL 3.0) as a starting point and modifying it to incorporate the best features of other existing XML business libraries.
In the mean time demand for new XML documents will continue while UBL is developed.

Advantages:

- Provides easily understood transition from traditional EDI and paper-based business practices
- Gets small businesses on board easily
- Defers the rocket science for later

Disadvantages:

- Only covers the vocabulary and dictionary parts of B2B. It does not address messaging, packaging, security, registries, partner agreements and business process aspects.
- It is not a standard, but more a shared library.

Conclusion
It is not a good candidate for eRA as it is limited to only one dimension of B2B.

Reference
http://www.xCBL.org
http://www.oasis-open.org/committees/ubl/

Other Standards/Specifications

This section describes briefly other players in B2B.

cXML

The commerce XML standard is an Ariba-created specification comprising a vocabulary and simple envelope defined for the express purpose of document exchange between trading partners using the Ariba product set and the Ariba Network. cXML is a streamlined protocol intended for consistent communication of business documents between procurement applications, e-commerce hubs and suppliers. The protocol does not include the full breadth of interactions some parties may wish to communicate. However, through the use of extrinsic elements and newly defined domains for various identifiers, it is easily expanded by such applications. This expansion is the limit of point-to-point configurations necessary for communication. cXML is DTD based.

More information can be found at: http://www.cxml.org/

fpML

fpML (financial products Markup Language) is the industry-standard protocol for complex financial products. It is based on XML (Extensible Markup Language), the standard meta-language for describing data shared between applications. The fpML language is a vocabulary for financial document exchange.

More information can be found at http://www.fpml.org/

OAG

The Open Application Group (OAG) standard consists of a large vocabulary of XML documents covering a broad range of application-to-application and business-to-business transactions. The mission of the Open Applications Group is to define and encourage the adoption of a unifying standard for e-Business and Application Software interoperability that reduces customer cost and time to deploy solutions.

More information can be found at http://www.openapplications.org/

ICE

ICE is a specialized transaction protocol for automating the distribution of content among business partners. It uses XML to define the format of messages that are sent between servers to negotiate a content transaction, which typically involves downloading a designated package of content for an agreed-upon price. An ICE syndication server runs in tandem with a content management system, from which content is extracted for distribution, and into which incoming content is integrated. ICE does not specify the format of the content itself, which could be anything from plain text to HTML or XML. ICE is a bi-directional, request-response protocol in which XML-based messages are exchanged between two different systems to conduct the transaction.

On the same day as the ICE press conference, the World Wide Web Consortium acknowledged that the ICE submission was under review.

More information can be found at http://www.xml.com/pub/a/98/10/ice.html

9. Glossary

Acronym	**Definition**
AIM	AOL Instant Messenger.
ANSI	American National Standards Institute. ANSI is a voluntary organization that creates standards for the computer industry.
API	Application Programming Interface. An API is a set of functions that can be used to work with a component, application, or operating system.
Ariba	Ariba is a software vendor.
ASCII	American Standard Code for Information Interchange. ASCII is a code for representing English characters as numbers
ASP	Application Service Provider
B2B	Business-to-Business. It refers to inter-enterprise integration applications, automation of business processes and communication of business information with trading partners.
B2G	Business-to-Government.
Base64	Base 64 is the data encoding system defined in RFC 2045: "Multipurpose Internet Mail Extensions (MIME), Part One: Format of Internet Message Bodies". This encoding is used as a standard to send binary information through network connections or by e-mail.
BEEP	Blocks Extensible Exchange Protocol. BEEP has been proposed by IETF. It is an application protocol framework for connection-oriented, asynchronous, request-response interactions.
BOV	Business Operational View. BOV is one of the views of the PIP standard of RosettaNet. It defines the business aspects of business transactions.
BXXP	BXXP is an XML-based messaging framework for building application protocols that was used as a starting point for the IETF BEEP Working Group's efforts.
C	General-purpose programming language that was originally designed by Dennis Ritchie of Bell Laboratories and implemented there in 1972
CA	Certificate Authority. A CA is an authority in a network that issues and manages security credentials and public keys for message encryption.
CDATA	Character Data. CDATA is a type of a section of an XML document defined in a DTD. They are letters, numbers and other symbols that are used exactly as they are typed. They are not parsed or processed, or treated as if they have any special meaning.
CGAP	Competitive Grant Application Process. The NIH eRA effort to receive and process competitive grant applications electronically.
COMMONS	The NIH eRA Commons. A Web-based, client-server system that enables data exchange between the external research community and NIH. Only registered organizations can access the working modules.
CPA	Collaboration Protocol Agreement. CPA is a subset of ebXML. A CPA describes the exact requirements and mechanisms for the transactions that two companies perform with each other.
CPP	Collaboration Protocol Profile. CPP is a subset of ebXML. A CPP describes a company's offerings in a standard, portable way. Specifically, it describes the message-exchange capabilities and business collaborations that a company supports.

Acronym	Definition
cXML	commerce XML. cXML is a streamlined protocol intended for consistent communication of business documents between procurement applications, e-commerce hubs and suppliers.
DIME	Direct Internet Message Encapsulation. DIME is a lightweight document encapsulation format.
DOE	Department Of Energy
DOM	The Document Object Model A W3C standard, is a platform- and language-neutral interface that allows programs to dynamically access and update the content the structure of an XML document.
DTD	Document Type Definition. A DTD provides the schema and underlying framework for a standardized XML document.
DUNS	Data Universal Numbering System. DUNS are used to uniquely define a trading partner and its location.
ebXML	Electronic Business XML. ebXML, sponsored by UN/CEFACT and OASIS, is a modular suite of specifications that enables enterprises of any size and in any geographical location to conduct business over the Internet
EC	Electronic Components.
EDI	Electronic Data Interchange. the transfer of data over a network between different companies.
EDIFACT	Electronic Data Interchange For Administration, Commerce and Transport.
eRA	Electronic Research Administration The NIH initiative and systems to promote enterprise level, paperless processing and management of research administration functions
FDP	Federal Demonstration Partnership. The FDP is a cooperative initiative among federal agencies and institutional recipients of federal funds.
fpML	financial products Markup Language. FpML is the industry-standard protocol based on XML for complex financial products.
FSV	Functional Service View. FSV is one of the views of the PIP standard of RosettaNet. It defines the technical aspects of business transactions.
FTP	File Transfer Protocol. FTP is a method of transferring computer files from one computer to another.
GTIN	Global Trade Item Number. GTIN is an identifier used for the unique identification of trade items worldwide.
GPG	GNU Privacy Guard. GPG is a complete and free replacement for PGP.
HIPAA	Health Insurance Portability & Accountability Act of 1996. The goal of HIPAA is to protect the security and confidentiality of electronic patient health information and is intended to cut the cost of print-based record transactions.
HL7	Health Level Seven. HL7 is an ANSI approved standard for exchanging information in the health industry.
HTML	HyperText Markup Language. HTML is the authoring language used to create documents on the World Wide Web.
HTTP	HyperText Transfer Protocol. The underlying protocol used by the World Wide Web.
HTTPR	Reliable HTTP. HTTPR is a protocol proposed by IBM that offers the reliable delivery of HTTP packets between the server and client.
HTTPS	HTTP Secure.

Acronym	Definition
	HTTPS tunnels HTTP communications through SSL.
IC	Institutes and Centers. The 27 institutes and centers that comprise NIH - for instance the National Cancer Institute.
ICE	Information and Content Exchange. ICE is a specialized transaction protocol for automating the distribution of content among business partners.
ICQ	ICQ stands for "I Seek You". It is a popular way to chat on the Internet.
IEC	International Electrotechnical Commission. It's an organization of 50 countries that was created "to promote international cooperation on all questions of standardization and related matters.
IETF	Internet Engineering Task Force. IETF is the main standards organization for the Internet.
IFV	Implementation Framework View. IFV is one of the views of the PIP standard of RosettaNet. It defines RosettaNet implementation aspects.
IM	Instant Messenger. IM refers to programs such as MSN Messenger, ICQ, Yahoo Messenger and so forth.
IP	Internet Protocol. IP is a way of addressing and routing data packages from a sender to a receiver.
IPF	Institutional Profile File. NIH specific numbering and identification system for grantee organizations. Contains the DUNS
ISO	International Organization for Standardization. ISO is an international organization composed of national standards bodies from over 75 countries.
IT	Information Technology.
JAR	Java ARchive. JAR is a file format used to bundle all components required by a Java application.
Java	Java is a high-level object-oriented programming language developed by Sun Microsystems.
Java VM	See JVM
JAXB	Java Architecture for XML Binding. JAXB provides an API and tools, developed by Sun Microsystems, that automate the mapping between XML documents and Java objects.
JAXP	Java API for XML Processing. JAXP, an API defined by Sun Microsystems, supports processing of XML documents using DOM, SAX, and XSLT.
JAXR	Java API for XML Registries. JAXR provides a uniform and standard Java API, defined by Sun Microsystems, for accessing different kinds of XML Registries
JDK	Java Development Kit.
JDOM	Java DOM. JDOM is an implementation of the DOM interface that provides direct services to represent an XML document in Java as a tree of elements and nodes.
JVM	Java Virtual Machine.
MIME	Multipurpose Internet Mail Extensions. MIME a specification defined by IETF for formatting non-ASCII messages.
NIH	National Institute of Health.
NSF	National Science Foundation.
OAG	Open Applications Group. OAG is a non-profit consortium focusing on best practices and process based XML

Acronym	Definition
	content for e-Business and Application Integration.
OASIS	Organization for the Advancement of Structured Information Standards. OASIS is a global consortium that drives the development, convergence and adoption of e-business standards.
OMB	Office of Management and Budget. Part of the Executive Office of the President.
P2P	Peer-to-Peer.
PDF	Portable Document Format. PDF is a file format developed by Adobe Systems.
PGP	Pretty Good Privacy. PGP is a technique for encrypting messages.
PHS398	Grant Application Form 398 of the Public Health Service.
PI	Principal Investigator. A qualified person designated by the applicant institution to direct the research and be responsible for the proper conduct of the project or program of a grant.
PIP	Partner Interchange Process. PIP, a standard of RosettaNet, defines business processes between trading partners.
PPF	Person Profile File.
RFC	Request For Comments
RNIF	RosettaNet Implementation Framework. RNIF specifies how to transport RosettaNet Objects and Business Messages between trading partners' network applications
RosettaNet	RosettaNet is a non-profit consortium working to create, implement and promote open e-business process standards.
RPC	Remote Procedure Call.
SAML	Security Assertion Markup Language. SAML is a security standard proposed by OASIS.
SASL	The Simple Authentication and Security Layer. SASL is a method for adding authentication support to connection-based protocols.
SAX	Simple API for XML. SAX is an event-based API used to parse XML data.
SCP	Secure CoPy. SCP is based on SSH for data transfer. It copies files securely between hosts on a network.
SF424	The Standard Form 424 is the Application for Federal Assistance form used by grant applicants
SM	Semiconductor Manufacturing
SMTP	Simple Mail Transfer Protocol. SMTP is a protocol used for sending e-mail messages between servers.
SOAP	Simple Object Access Protocol. SOAP is a platform independent protocol that allow remote applications to communicate with each other.
SPML	Service Provisioning Markup Language. Open standard for defining and exchanging identity provisioning requests in XML.
SSH	Secure Shell. SSH can be defined roughly as a secure version of Telnet.
SSL	Secure Socket Layer. SSL is a protocol residing in the network layer on top of TCP-IP which secures data transmissions.
SwA	SOAP with Attachments. A W3C standard that allow a SOAP message to be bundled with binary files in a MIME multipart/related message.

Acronym	Definition
TCL	Tool Command Language. An interpreted programming language.
TCP	Transmission Control Protocol. TCP is a reliable networking layer on top of IP.
UBL	Universal Business Language. UBL is a standard library of XML business documents developed by OASIS.
UDDI	Universal Description, Discovery, and Integration. UDDI specifications define a way to publish and discover information about web services.
UDP	User Datagram Protocol. UDP is an unreliable networking layer which sits at the same level of the networking stack as TCP.
UML	Unified Modeling Language. UML is a general-purpose notational language for specifying and visualizing complex software in object-oriented projects.
UN/CEFACT	UN/CEFACT is the United Nations Centre for Trade Facilitation and Electronic Business.
UN/SPSC	United Nations Standard Products and Services Code
UNIX	UNiplexed Information and Computing System. UNIX is a popular multi-user, multitasking operating system.
URI	Uniform Resource Identifier. A URI is the generic term for all types of names and addresses that refer to objects on the World Wide Web.
URL	Uniform Resource Locator. A URL is a kind of URI. It is the global address of documents and other resources on the World Wide Web.
UTF	Universal Transformation Format. UTF is a method of converting Unicode characters, which are 16 bits each, into 7- or 8-bit characters.
W3C	World Wide Web Consortium. W3C is an international consortium of companies involved with the Internet and the Web.
X12	Set of EDI standards approved by ANSI.
X509	Most common type of digital certificate. Identity verification can be carried out using either secret-key techniques or public-key techniques.
XACML	eXtensible Access Control Markup Language. Standard proposed by OASIS, which is used in conjunction with SAML. It provides a means for standardizing access control decisions for XML documents.
XCBF	XML Common Biometric Format. XCBL, developed by OASIS, defines a common set of secure XML encodings for biometrics.
xCBL	XML Common Business Library. xCBL is a set of components that allows the creation of robust, reusable XML documents to facilitate global trading.
Xenc	XML Encryption. Xenc is a standard of W3C and IETF which allows encryption of selected sections of an XML document.
XKMS	XML Key Management Specification. XKMS defines a way to distribute and register the public keys used by the XML Signature specification.
X-KRSS	XML Key Registration Service Specification. X-KRSS is a subset of XKMS that allows public keys registration.

Acronym	Definition
X-KISS	XML Key Information Service Specification. X-KISS is a subset of XKMS that helps resolving the keys provided in an XML Signature.
XML	Extensible Markup Language. XML is a text-based meta-language that uses tags, elements, and attributes to add structure and definition to documents.
XML-RPC	XML with RPC. XML-RPC allows disparate operating systems in different environments to make procedure calls over the Internet.
XrML	eXtensible rights Markup Language. XrML provides a universal method for specifying rights and issuing conditions (licenses) associated with the use and protection of content.
XSL	eXtensible Stylesheet Language. XSL is an extension of XML that enables the automatic transformation of XML data into specified presentation formats such as HTML or PDF.
XSLT	XSL Transformations. XSLT is a part of XSL. XSLT stylesheets convert XML documents or data into the desired end formats.
XSLTC	XSLT Compiler initially developed by Sun Microsystems and donated to the Xalan-Java project on Apache.org.