Exchange Network Node Implementation Guide v1.0

FINAL

April 25, 2003

TABLE OF CONTENTS

LIST OF TABLES

INTRODUCTION

The Node 1.0 Implementation Guide provides Exchange Network (Network) Partners a resource to *supplement* the Network Exchange Protocol v1.0 (Protocol), the Network Node Functional Specification v1.0 (Specification), the Network WSDL v1.0, the Network Security Guidelines v1.0 (Security Guidelines), and Demonstrated Node Configurations (DNCs). Network implementers should always refer to the Network website, www.exchangenetwork.net, for the most up-to-date version of all Network documents.

The purpose of the Implementation Guide, as compared with the other Node 1.0 products, is to focus on the *implementation process* itself. The Guide provides Node Implementers with additional information which, due to either format, timing or content, is not included in the other Network guidance documents.

This document contains 3 sections:

- Section 1 introduces and describes the Node 1.0 project, Protocol and Specification, Network Security, and the basic Network Technologies.
- Section 2 identifies generic requirements and recommendations for Node building, anticipated and available Network resources, and lessons learned from the implementation of pilot Nodes.
- Section 3 provides specific information and a description of each of the Node 1.0 participants Node implementation with more specific information on how to install a Node contained in the DNCs.
- Appendix– Node Process Flow Diagrams
- Glossary– Definitions of terms and acronyms

The Node 1.0 team thanks the following individuals and companies for their participation, valuable insights, and hard work in making this a successful project.

State Participants

Dennis Burling (State Co-chair), Nebraska Department of Environmental Quality
David Blocher, Maine Department of Environmental Protection
Harry Boswell, Mississippi Department of Environmental Quality
Dan Burleigh, New Hampshire Department of Environmental Services
Frank Catanese, New Hampshire Department of Environmental Services
Ken Elliott, Utah Department of Environmental Quality
Dave Ellis, Maine Department of Environmental Protection
Karen Knox, Maine Department of Environmental Protection
Renee Martinez, New Mexico Environment Department
Tom McMichael, New Mexico Environment Department
Melanie Morris, Mississippi Department of Environmental Quality
Dennis Murphy, Delaware Department of Natural Resources and Environmental Control
Brent Pathakis, Utah Department of Environmental Quality

Brian Shows, Mississippi Department of Environmental Quality
Chris Simmers, New Hampshire Department of Environmental Services
Michael Townshend, Delaware Department of Natural Resources and Environmental Control
Robert Williams, Maine Department of Environmental Protection

EPA Participants

Connie Dwyer (EPA Co-chair), Office of Environmental Information, Central Data Exchange
Chris Clark, Office of Environmental Information, Central Data Exchange
Patrick Garvey, EPA NSB Executive Staff

Environmental Council of States

Molly O'Neill, ECOS NSB Executive Staff

Node 1.0 Project Contractors/Consultants

Kochukoshy Cheruvettolil, Ross & Associates Environmental Consulting, Ltd.
Tom Potter, Computer Sciences Corporation
Andrea Reisser, Concurrent Technologies Corporation
Louis Sweeny, Ross & Associates Environmental Consulting, Ltd.
Glenn Tamkin, Computer Sciences Corporation
Rob Willis, Ross & Associates Environmental Consulting, Ltd.
Yunhao Zhang, Computer Sciences Corporation

State Contractors/Consultants

Calvin Li, Oracle
Brad Loveland, Venturi Technology Partners
Chris McHenry, Integro
Tony Pruitt, Ciber Federal Solutions
Brett Stein, XAware Inc.
Steven Wu, enfoTech & Consulting Inc.

SECTION 1

1.1 Node 1.0 Project Description

The Node 1.0 project was a follow-on to two previous pilot projects. The purpose of the projects was to learn as much as possible about building Nodes, using Nodes on the Network, and to develop basic Network technical guidance documents. In June 2002, at the onset of the Node 1.0 project, the eight participants, Delaware, Maine, Mississippi, Nebraska, New Hampshire, New Mexico, US EPA, and Utah in conjunction with the Technical Resource Group (TRG) received formal training on basic Network technologies. In addition, the Node 1.0 group held weekly conference calls for the duration of the project and hosted seven nation-wide Node Knowledge Calls. Over the life of the project the participants created and implemented three versions of Nodes and documentation finally arriving at the version 1.0, ready for immediate Network implementation.

The following are the products from the Node 1.0 project:

- Network Exchange Protocol v1.0 (Protocol)
- Network Node Functional Specification v1.0 (Specification)
- Network WSDL file v1.0
- Network Security Guidelines and Recommendations v1.0
- Network Node 1.0 Implementation Guide
- Node 1.0 Demonstrated Node Configurations
- Node 1.0 team recommendation to the NSB

1.2 Node 1.0 Test Service Requests

Service requests and responses are the currency of the Network. Partners are expected to use the Protocol and Specification, the Flow Configuration Document (FCD), Trading Partner Agreements (TPA), Network Schema, and other Network resources to design and implement Network service requests and responses. In some cases, such as standardized Network Flows, many of the exchange details will already be solidified and all a Partner has to do is conform to the pre-created Flow definition. In others, the partners themselves will specify these. In any event, it is the responsibility of the Partners involved in any given exchange to assure that all essential details of an exchange are specified. Interoperability of the Network depends on it. The NSB will support this process by chartering follow-on guidance development work and communicating the best practices established by early Flows.

In order to realistically test the Protocol, the Node 1.0 group established the Test Service Requests shown in Table 1. The Test Service Requests were only used for the Query and Solicit Web Methods (defined below). All other Network Web

Methods used dummy documents. It is imperative to note that these *service requests were created simply for the Node 1.0 project and should not be considered formal Network Flows.* The Node 1.0 Project service requests should only be used as reference examples. Until Flow/exchange guidance, such as the FCD, are available, all exchanges over the Network will have to use a similar, informal process for establishing exchanges.

Table 1: Node 1.0 Project Test Service Requests (Test requests only, do not implement)

Test Service Request Name	Parameter 1	Parameter 2	Parameter 3	Response Schema
GetFacilityByName	**State USPS**	**Facility Name**		FRS Schema
Expected Parameter Value Format	*XX*	*String*		
GetFacilityByID	**State USPS**	**State Facility ID**		FRS Schema
Expected Parameter Value Format	*XX*	*String*		
GetFacilityByChangeDate	**State USPS**	**ChangeDate**		FRS Schema
Expected Parameter Value Format	*XX*	*YYYY-MM-DD*		
GetFacilityBySICcode	**State USPS**	**SIC**		FRS Schema
Expected Parameter Value Format	*XX*	*Number*		
GetAllNEIByYear	**State USPS**	**Year**		NEI Schema
Expected Parameter Value Format	XX	YYYY		
GetNEIByStateFacilityIDByYear	**State USPS**	**StateID**	**Year**	NEI Schema
Expected Parameter Value Format	*XX*	*String*	*YYYY*	
GetNEIByEPAIDByYear	**State USPS**	**EPAID**		NEI Schema
Expected Parameter Value Format	*XX*	*Number*		
GetNEIByAirPRogramIDByYear	**State USPS**	**AirProgramID**	**Year**	NEI Schema
Expected Parameter Value Format	*XX*	*Number*	*YYYY*	

1.3 The Protocol and Specification

The Protocol and Specification define the behavior and conversations between Nodes on the Network. The Node 1.0 group expects the Protocol and Specification to have a shelf life of 12-24 months. As a result, the documents are forward-looking. They define and describe certain functionalities that Partners will not immediately utilize but are expected to become paramount as the Network evolves during its initial implementation. For example, the documents discuss and describe Universal Description Discovery and Integration (UDDI) and other Registries as integral parts of the Network. Their use is implicit in the Protocol and Specification and official registries are expected to exist in the next 12-24 months, therefore merit discussion in these documents.

The FCD is identified in the protocol but not fully defined. FCD guidance is a priority of the NSB and should be available in the next 6-8 months. The FCD is identified as the mechanism for explicitly specifying the Flow specific details of exchanges. Once designed and defined, the FCD will contain all the information not governed by the Protocol, Specification, and Security Guidelines that a Partner needs to know to configure and execute a Network service request. The relationship between the FCD and TPA has not yet been determined; the FCD may simply be considered the technical specification in a TPA.

The Protocol and Specification, in their first generation, were designed to support relatively simple state and EPA data flows. They do so by proposing nine primitive Network Web methods which Network Partners group into larger (but still simple) transactions or Network exchange business processes to flow data. The Protocol and Specification describe one optional Network Web Method, Execute.

Table 2 contains a description of the nine required Network Web methods. Table 2 also identifies implementation notes that might be helpful as Node builders begin configuring their Nodes. Lastly, the column entitled, “Information in FCD,” contains the likely information that Partners will have to define in an FCD when using the web method in an exchange. Early Network users should find the information in the column particularly useful as it identifies much of the information that needs to be informally defined before a successful exchange can occur.

Table 3 contains Example Network Exchange Business Processes. The Web Methods, just defined, are combined to create a Network Exchange. Network Partners can combine the Web Methods in any way; the following represent what some of the most common combinations are likely to be.

Table 2: Network Web Methods Description and Implementation Notes

Method	Description	Parameters	Implementation Notes	Information in FCD
Authenticate	The Authenticate method is used to verify users on the Network and is expected to be the first method a requester utilizes to obtain access to the Network. The parameter values are determined by the Partner's security strategy and are described in the Protocol and Specification. For instance, the Node 1.0 group used a centralized security strategy and the central security administrator determined the parameter values. The Node 1.0 security strategy is briefly described in the "Network Security" section below and in greater detail in the Network Security Guidelines and Recommendations v1.0.	• userId: the user ID of the person or system • Credential: The value of credential is the user's credential for accessing the Network services. • Authentication Method: specifies which authentication methods are to be used. The default authenticationMethod, and the only method supported by the V1.0 Specification, is password.	• Every invocation of the Authenticate method must be through a secure transport such as SSL. • To use SSL you will need a certificate and to appropriately configure your server to use SSL. SSL uses port 443 by default. • 128-bit SSL encryption is used for all transactions other than NodePing.	None, method is generic to all flows.
Solicit	The Solicit method is an asynchronous version of Query. Asynchronous means that the Solicit method performs the requested operation in the background or sometimes offline and returns the result when the service provider determines is appropriate (as opposed to Query where the responder is required to immediately service the request). Solicit is designed to be used for queries that may take a long time to service. The Service Provider is responsible for determining when to run the process and to determine if the request is honored using Solicit.	• securityToken: A security ticket issued by the service provider. • returnURL: A Node address where results can be submitted. The service provider must call the Submit method at the specified address if it is not empty. If returnURL is empty, then it is the requester's responsibility to download the result. • request: The operation to be performed. It is usually the name of a predefined database request. • parameters: An array of parameter values for the database request.	• Other then the addition of the returnURL, rowId, and maxRows parameters, the Solicit method is structured exactly like the Query method. • The returnURL is important as it determines the expected behavior of the service provider. If the Solicit request message has a value for the returnURL, once the service provider has finished processing the request, the result set is sent to the URL specified in this parameter. • If the returnURL parameter is blank, the service provider should wait for the service requester to use the download method to retrieve the information.	• returnURL. • allowed Requests (e.g. GetMontlyReport(01-01-2003) • allowed parameters and their associated specification values for respective requests .
Query	The Query method allows a Partner to send an information request to another Partner's Node and receive the data in XML formatted as per its respective Schema.	• securityToken: A security ticket issued by the service provider. • request: The database logic to be processed. It should be the name of a stored procedure in general. • rowId: The starting row for the result	• Given the inherent flexibility of this method, partners will need to establish "approved queries" and document them in the FCD for that flow. Many of these queries will be standardized across the Network (e.g. GetFacilityByID) while others	• Request. • parameters and their associated specification. • RowID (user

Method	Description	Parameters	Implementation Notes	Information in FCD
		set, it is a zero based index to the current result set. The value of rowId must be either null or 0 if positionedfetch is not requested. • maxRow: The maximum number of rows to be returned. The service provider uses a default value if maxRow is 0 or negative. • parameters: An array of parameter values for the stored procedure.	will be unique to sets of trading partners. Nodes typically implement Query (and Solicit) by establishing an internal stored procedure which produces the required information and is invoked with its corresponding Query request is received.	defined /default and null values) • maxRow (user defined/default and null values) •
GetStatus	GetStatus is the method a requester uses to query the status of a previous transaction. The requester sends a securityToken and a transaction ID obtained during a previous transaction from a service provider (Node).	• securityToken: A security ticket issued by the service provider. • transactionId: is a transaction identification returned by the Submit or Notify method.	• None	• Indicate flow specific GetStatus messages (if any)
Submit	A Network Partner invokes the Submit method to push information to another Network Partner.	• securityToken: an opaque string returned by the Authenticate method. • transactionId: A transaction ID for the submission if the operation is a result of an asynchronous operation. It should be empty if the Submit operation is independent. • dataflow: The name of target dataflow. • documents: An array of documents of type nodeDocument. Each. nodeDocument structure describes a single attachment or payload.	The permissible values for dataflow will be established by trading partners and documented in the FCD. In many cases these values will be standardized for national systems-bound flows with EPA, in others they may be developed by smaller trading partner groups for unique flows.	• dataflow value • Attached document parameters, naming, and organizational convention. • Description of attached document (i.e., content, governing schema)
Notify	Notify has three intended uses: 1. document notification 2. status notification 3. event notification	• securityToken: An authentication ticket issued by the service provider. • nodeAddress: For document notification, the parameter contains a Network Node address where the document can be downloaded. It should contain the initiator's Node address, or empty if not applicable,	For document notification, Notify is used in conjunction with other Web methods to complete an entire Network exchange business process. For instance, a service provider uses Notify to inform a service requestor that a document is ready for download. All the information needed to initiate a download, i.e., the location of the documents, is contained in the Notify	• Permissible values for every Notification type.

Method	Description	Parameters	Implementation Notes	Information in FCD
		for event and status notifications. • dataflow: The target dataflow. It identifies an event or status. • documents: An array of related documents.	message. Event and Status notification will either stand alone or be used in conjunction with other web methods. The v1.0 Protocol and Specification do not explicitly define the use of Event and Status notification.	
NodePing	The NodePing method is designed for checking the availability of a Network Node. A positive response from the Node indicates that it is alive and well. A Network error (no response) or SOAP Fault (not ready) means that the service is not available at this time. NodePing is the only operation that does not require authentication.	• The method has one argument that may contain arbitrary text, preferably short or even null.		None, method is generic to all flows.
Download	The Download method is the mechanism used on the Network for document retrieval (document pulling).	• securityToken: A security ticket issued by the service provider. • transactionId: A transaction ID for the submission. It should be the same transaction ID issued by the Node. • documents: An array of nodeDocument structures. It should contain the same set of documents given by the Notify method.	The two expected Network exchange business processes are: a) document retrieval after a Notify, or b) pre-scheduled download. Because the Download method gives access to documents to the requester, Nodes are responsible for implementing any needed authorization. A simple approach used by the Node 1.0 group established a secured directory for each user and placed a document in that folder and "authorized" that user to download it.	• Downloaded document parameters. • Attached document parameters, naming, and organizational convention. • Description of attached document (i.e., content, governing schema)
GetServices	Ultimately UDDI will be used to discover the services Nodes provide. Until then, GetServices provides a simple way for requesters to query services provided by a Network Node	• ServiceType: A complete list of all service types that can be used as the value of the element. • Interfaces: The web service interfaces supported by the Node. • Query: Database queries supported by the Node. A string value "SQL" in the response message means that the Node supports ad hoc queries.	The ServiceType parameter allows a requestor, by leaving it blank, to receive a list of all the Web methods supported by the requested Node. A requester can then send a subsequent GetServices request with the value of the ServiceType parameter a Web method. The response would be all of the services the Node can provide	None, method is generic to all flows.

Method	Description	Parameters	Implementation Notes	Information in FCD
		• Execute: SQL DML (Data Manipulation Language) procedures provided by the Node. A string value "SQL" in the response message means that the Node supports ad hoc SQL statements.	using that Web method. This would be particularly useful, for instance, in determining what queries a Node services using the Query Web method versus the Solicit Web method.	

Table 3: Example Network Exchange Business Processes

Network Exchange Business Process	Associated Network Web Methods		Example
Simple Document Submission	• Authenticate • Submit • GetStatus (optional)		State Node transmits monthly report to EPA CDX.
Requested Download	• Authenticate • Notify • Download		State Node notifies EPA/CDX of the availability of a monthly report for download.
Sending Network Events	• Notify (Event)		Node notifies a trading partner that it is going down.
Broadcasting Network Events	• Notify (Broadcast– under development pending creation of a UDDI Registry)		Node notifies multiple trading partners that it is going down.
Retrieving Information with Query	• Authenticate • Query		Client application queries a Node for "drill down" information on one monitoring location.
Performing Asynchronous Operations	Pure Client Interaction	Network Node Interaction	One partner routinely requests a large or complex query from a partner Node, which the partner services as resources permit.
	• Solicit • GetStatus • Download	• Solicit • Submit	

1.4 Network Security

Each Network Partner is ultimately responsible for security of their Node. The Protocol and Specification present a simple but extensible security approach for immediate Network implementation. The Authenticate (described above) method is the only security mechanism described in the Protocol and Specification. In the current security model, once authenticated, a Partner using legitimate parameters can access all services provided by the authenticating Node. Note that any Node or client can use the Authenticate service to obtain authentication. As a result, any and all potential data exchanges can use Authenticate.

The EPA/CDX has created the Network Authorization and Authentication Services (NAAS). Any Network Partner, subject to approval by CDX, can use this service to authenticate and authorize their users. (Note: all Node 1.0 participants used the NAAS for authentication. Authorization was not available for the Node 1.0 project). An additional Web service, Validate, is required, to use the NAAS. The use of the NAAS and Validate is described in a separate document available from CDX.

1.5 Node 1.0 Implementation of the Web Services Stack

Table 4 contains the information about how the v1.0 Network implements the Web services stack and what the likely future Network use is. The table also contains a description of each component and a narrative describing the likely future Network use of each Web service technology.

Table 4: Network Implementation of the Web Services Stack

	Description	Node 1.0 Implementation	Likely future direction
Universal Description, Discovery, and Integration (UDDI)	This layer is responsible for centralizing services into a common registry and providing publishing/finding functionality. The Network will create and operate one private UDDI registry shared by all Network Nodes.	The Protocol and Specification identifies UDDI as a component in the overall Network architecture but does not define the use because the Network does not currently have a UDDI registry or infrastructure. Currently the GetStatus web method is the only mechanism available for service discovery. Network Partners who plan a Node for Web services outside of the Network may consider a UDDI implementation independent of the Network. A Node 1.0 participant, Maine, is currently planning a State level UDDI implementation.	The NSB, working with EPA/CDX is investigating mounting a UDDI server. A prototype may be available by summer for preliminary testing. Many free test UDDI services are also available now.
Web Services Description Language (WSDL)	WSDL is one of the core technology components of the Network. All Network exchanges are governed by a WSDL. The Protocol and Specification are accompanied by a generic Network WSDL file.	The Protocol and Specification and the Network WSDL file contain the same information. The Network WSDL file is a terse description of the Protocol and Specification. If discrepancies exist between the WSDL file and the Protocol and Specification documents, assume the WSDL file is always correct. Currently, the Network supports only WSDL v1.1	As SOAP and WSDL standards evolve, the Network is expected to migrate to WSDL 1.2 if compatible with future versions of SOAP. WSDL 1.2 is expected to be backwards compatible with earlier versions. The Network anticipates more extensive use of WSDL to document the information now contained in the FCD, especially Query/Solicit methods. Users should be able to generate fully functional clients/stubs from the next version of WSDL files.
Simple Object Access Protocol (SOAP)	SOAP is the messaging component of the Network. As an XML messaging component SOAP is designed to bind to any transport protocol.	The Network only supports SOAP v1.1 and must be used as it is the only version of SOAP that is compatible with WSDL 1.1. All SOAP fault messages must confirm to SOAP v1.1 and use the predefined SOAP fault codes. All Network messages must be in conformance to the W3C SOAP schemas defined at: http://schemas.xmlsoap.org/soap/envelope/ (SOAP Envelope) http://schemas.xmlsoap.org/soap/encoding/ (SOAP Encoding)	As SOAP and WSDL standards evolve, the Network is expected to migrate to SOAP 1.2, if and when compatible with future versions of WSDL.
Transport Protocol	The Protocol and the Specification identify HTTP as the only transport protocol supported by the Network. The XML messaging component, SOAP, can hypothetically bind to any transport protocol but the SOAP specification currently only defines SOAP over HTTP	HTTP is an extremely mature, stable, and widely accepted standard; as a result, a standard Network HTTP implementation will not be defined.	

SECTION 2

2.1 Node Building

Node deployment is a software development and implementation project. Like any such project, it requires planning and resources. Node deployment can probably best be compared to a small systems integration/web-server project.

The Partner's Node implementation team must have some understanding of Network technologies such as XML, SOAP, Web services, XML Schemas, and WSDL. Additionally, it is imperative that part of the Node implementation team be intimately familiar with the existing information systems to which the Node will be attached. Most of the Node 1.0 participants relied on contractor support for the technical details of their Node implementations, but the knowledge of the existing information systems was a key component in troubleshooting and in preparing to use the Node to flow information.

2.2 Acquiring Contractor Support

Many Partners will need to use outside technical support during their Node implementations. The Node 1.0 group was no different. The states and EPA each had different levels of support ranging from two days of training on the middleware to full blown Node implementation and ongoing support.

Also, it is important to note that before acquiring a contractor each of the Node 1.0 states had already chosen their middleware, established their Node architecture (The Node within the context of their agency architecture), received training on the Network technologies, defined their immediate Node usage, and in the case of four participants, had been involved in previous Node building efforts (Alpha Phase and Beta Phase pilots). Many Partners may opt to use contractor support to do one or all of these functions. Further, each of the Node 1.0 state members is creating a Demonstrated Node Configuration (DNC). It is possible that the DNCs will allow Partners to mitigate the level of contractor assistance needed during the Node implementation.

The Network website contains copies of each of the states' RFPs for Node 1.0 contractor support. Due to the limited project budget, each state only had $10,000 to spend on contractor support. This is reflected in each of the RFPs and state participation in the Node 1.0 pilot project was contingent on the creation of v0.8, v0.9, and v1.0 Nodes.

The RFPs contain a list of discrete tasks each of the Node 1.0 states had to perform during the pilot project. While this list is not applicable to all Node implementations, the Node 1.0 group believes that it accurately captures the number of steps during an implementation and what likely tasks the contractors

are used for. Furthermore, the "Strategies for Node Building" section of this document outlines a recommended sequence of web method/Node implementation that might be useful when soliciting contractor support.

2.3 Tool Selection

The Node 1.0 team had to create a Protocol and Specification that provided a robust and interoperable Network framework that could be built using a still young and often incompatible set of new products. The rule of thumb was to not incorporate technologies into the Protocol which were not (or would not shortly) be supported by the major middleware products. This rule motivated the decision to NOT fully utilize WSDL and instead rely on a human readable FCD for the Solicit and Query methods. The tools simply did not support this functionality consistently.

An early step in a Node project is to select the software or middleware to 'run' your Node. The middleware provides an Integrated Web Service Environment and a SOAP toolkit. Web Services and SOAP are leading technologies currently under development; the same is true for related products. Selecting the appropriate products for your implementation may not be an easy task. This section outlines items to consider when selecting a toolkit, what features to look for, and where to find more information.

Note that the recommendations in this section are based on technical features required by the Network Nodes. Costs, technical resources, and other policy related issues are not addressed. Each Node 1.0 state has provided the rationale for their tool selection in Section 3 (Table 5).

2.3.1 Middleware

Server-Side Infrastructure

A Node tool should provide server-side infrastructure for deploying, managing, and running SOAP-enabled services. The following is a list of features to look for:

- Create or Generate WSDL: A middleware should be capable of creating a WSDL file from an existing component (JAVA, EJB, or COM).
- Generate Web Service Code: A Node tool should be capable of generating web service code given a WSDL file, including proxy code for client and stubs for server.
- Easy Deployment: A Node tool should allow seamless deployment of web services from the development environment to production.
- Debugging: An integrated debugging tool is highly desirable.
- Logging and Testing: A Node tool should provide extensive logging for problem tracking. It should also provide tools for unit tests. Many

potential Node tools generate simple HTML pages where web services can be accessed using a browser.
- Web Service Management: A good Node tool should allow easy service registration and be able to enable/disable web services at runtime.

Integration

Web services are only part of an information infrastructure. The ability to build web services based on existing components and integrate web services into other systems is a very important feature of a Node tool. Accessing external components, e.g., ODBC, EJB, CORBA, or COM, from within a web service is essential.

Back-end Support

A Node tool should provide easy access to database systems and legacy systems. Database access is a required feature in the Node functional specification. It will be especially helpful if your web service development environment provides database-to-XML mapping and XML-to-database mapping.

SOAP Services

Interoperability

Interoperability is the most important requirement for SOAP services. This is because your web services will be accessed by other Nodes using different toolkits, and perhaps written in another programming language.

Pioneers of SOAP development have long recognized the danger of incompatibility, and they started to work on a set of standard interoperability tests in early January 2001. The third round of interoperability testing is underway, and great success has been made so far. It is highly recommended that you select a toolkit that has participated in this industry-wide effort, and demonstrated wide interoperability.

There are four major groups of interoperability tests that have been done so far:

- Base: Tests simple data types, arrays and structures in SOAP/RPC style messages.
- Group B: Focus on complex data structures, two-dimensional arrays and nested structures.
- Group C: Tests SOAP header capabilities and compliance to SOAP 1.1 specification.
- Group D: Tests WSDL parsers and document/literal style messages. Importing WSDL documents or schemas from a remote location is part of the tests.

Among these interoperability tests, Base and Group D are the most important ones for Network Nodes. It is recommended that you ask the toolkit vendor to provide a table of test results.

SOAP 1.1 Compliance

SOAP 1.1 is the protocol for Network Nodes. SOAP 1.1 defines a list of features. The following are important for Node operation:

- Support simple and complex data types
- Support one-dimension SOAP-ENC:Array types, including array of Structs (note: this support is essential--- the Node 1.0 uses it to carry parameter information for the query and solicit methods)
- Support SOAP-ENC:base64, xsd:base64Binary and xsd:hexBinary types,
- Support multi-ref (id/href) in both request and response messages.

Attachment Support

It has long been recognized that SOAP is inefficient for sending large documents. The solution is to use attachments. There are two specifications available for attachments: SOAP with Attachments and DIME. However, the Network, as defined in the Protocol and Specification, only supports DIME attachments. DIME attachments are considered part of a SOAP stack. This means that, if you select a complete package, you will not need to develop anything for attachments.

Just as it would not be wise to buy an email reader that would not support attachments, the same is true for a SOAP toolkit. Network Nodes are for data exchanges; data flows are document oriented. It will be extremely hard to do the job without attachment support.

WSDL Support

WSDL files are specifications (instructions) for computers. Thanks to wide support from vendors, many middleware products now understand WSDL. They are able to automatically generate stubs and client-side code, which greatly simplifies web service development. In some instances, on the client-side, you can execute web services without a single line of code (a process called dynamic binding), given the WSDL file. The first generation of the Network, however, does not support dynamic binding, but it is expected that as the toolsets and the web service standards continue to mature and the Network increases in complexity, that dynamic binding becomes an integral part of the information exchange process.

Not all SOAP toolkits support WSDL, so it is important that you choose one that does.

Encoding Styles

Only SOAP/RPC (Section 5/Section 7) encoding is required by the Network. However in the near term it is likely that the Network will also incorporate Document/Literal encoding. Your toolkit should be able to serialize/deserialize messages in these styles.

SOAP Header Processing

The SOAP header is an essential part of SOAP. Many application level extensions, such as routing and signature, are carried on SOAP message headers. Your selected SOAP toolkit must support headers and dispatch to your components for custom processing as required.

Encryption and Signature

Encryption and Signature are integral parts of web service security. XML encryption and XML signature are mature standards (both are W3C recommendations) for securing SOAP messages. It would be ideal if the toolkit you choose could encrypt and sign SOAP messages. If it does not, make sure there is space for you to plug in an external or third party security package in the future. It is expected that as the Network security model matures, that some type of encryption and signature will be used.

SSL Transport and Proxy Support

The primary transport defined in the Network Exchange Protocol is HTTPS (SSL). If you are going to build client side applications, you will also need to make sure that secure proxy support is provided.

Custom/Flexible Serialization and Deserialization

In most situations, SOAP toolkits should automatically deserialize SOAP messages.You will need to build your own components to deal with custom structures and XML documents. Your toolkits should allow you to plug in the components easily (pluggable serializers).

Routing

SOAP routing allows you to send a message through several intermediaries; each provides additional service such as logging, auditing and authenticating. It is not a must-have, but certainly a useful service to have.

2.3.2 Other Resources

Here is a list of SOAP resources where you can find further information:

- SOAP Toolkit List: A comprehensive list of all SOAP Toolkits. You can also find a list of first grade SOAP implementations in the open directory. http://www.soapclient.com/rss/rss.sri?requestname=getOPML&uri=http://services.soaplite.com/toolkits.opml
- SOAP 1.1 References http:/www.zvon.org/xxl/soapReference/Output/index.html

2.4 Node-Building using the Network WSDL

The Network WSDL file 1.0 is machine-readable and is the canonical description of the Protocol and Specification. Node implementers should use the WSDL file as the starting point for their Node and client development. Each Node will have to customize the generic WSDL file for their Node. The ability to generate code is an essential feature of most SOAP toolkits and could potentially simplify Node and Client creation.

2.5 Strategies for Node Building

2.5.1 Suggested Activities for Getting Started

There are many activities Partners can begin before the actual implementation of their Node. For example, any activities in preparing your data for exchange. Partners can begin data cleaning, data mapping, and formulating your TPA agreement with EPA and other Partners. These activities should be completed before a Partner can begin using their Node and can take much longer, and cost much more, than expected. They can lead to policy, data collection, and storage changes, none of which occur rapidly.

Additionally, Partners must start thinking about components of their local Node supporting infrastructure. The Protocol and Specifications identify the necessary minimum technical Node requirements, but additional hours and dollars will be needed to build the local supporting structure for your state node. This could include policy and system decisions, as well as doing systems setup work and additional application code. Local Node enhancements could include, custom Node user interfaces, local security mechanisms, and workflow management. The level of local effort required will vary by Partner experience, toolset, and intended Node use, and for some Partners it may represent a substantial investment.

2.5.2 Essential Steps For Building a Node

In the Beta Phase pilot, the participants learned that, like most projects, incremental Node implementation helped with efficiency and troubleshooting. The Node 1.0 participants attempted to develop a sequence for Node implementation. This sequence assumes that a Partner has already identified what information they want to exchange and have agreed to a format with their Trading Partners. All Node 1.0 participants, more or less, followed this order during Node Implementation:

Step 1: Decide to build a Node
While obvious, this step must come first.

Step 2: Establish and identify the necessary resources to build a Node

This step involves the identification of available and missing human resources, available and necessary hardware and software, identification of other resource constraints, mitigating administrative challenges, and other administrative tasks. This is likely to be one of the most challenging steps in the entire Node Implementation. As mentioned in the Tool selection section, Node implementers will have to identify their Node hardware and software based on multiple factors.

Fortunately, there are many resources available during this step, including at least eight experienced Node builders (the Node 1.0 team). Other resources available include the Network Help Desk, Network Readiness Grants, Demonstrated Node Configurations, Network Guidance Documents, and other Network resources.

Step 3: Install Node Hardware and Software

Step 4: Configure Node to comply with the Protocol and Specification

To participate on the Network, Partner's must build Nodes that comply with the Protocol and Specification. Compliance with the Protocol and Specification is compulsory to assure interoperability and proper Network operation. To assist Node-building, Node builders should consider using the following sequence when building Nodes:

Step 4a: Establish NodePing Method

NodePing is the most basic service and it allows others to have an initial interaction with your Node. A successful NodePing message exchange is a major milestone. It indicates that the web service framework and the development/deployment environment have been set up successfully.

Step 4b: Establish Authenticate Method (Authenticate and Validate Method if using NAAS)

All methods other then NodePing require authentication. If you will be authenticating users, consider using a simple user name and password authenticate scheme at beginning. The method must return an encrypted security token when successful. It is also good time to add an internal function to verify security tokens. If you are planning on using the NAAS, you must contact CDX to receive specific instructions on NAAS usage policies and procedures.

Step 4c: Establish Notify and Download (retrieve interface)

The retrieve interface is for delivering documents in the response message. This is the key interface for the CDX Node to interact with state Nodes. This will also test how good the selected SOAP toolkit is in terms of handling custom data structure (the cdxDocument structure).

Step 4d: Establish Submit and GetStatus

This step is a good time to create a transaction table that will store all transactions retrieved. The table is required by the GetStatus method. The Submit method will also test the DIME implementation of the SOAP toolkit.

Step 4e: Establish GetServices

The method allows other Nodes to query the extent of a Node implementation. It is relative easy to implement because it does not involve database operation and attachments. The GetServices method paves the way for implementation of Query and Solicit.

Step 4f: Establish Query

Step 4f can coincide with steps 5 and 6. You will not be able to honor a Query method until steps 5 and 6 are complete.

Step 5: Establish Data Validation procedures

There is a relatively easy way to check Schema conformance of submissions: passing the XML file into a validating XML parser along with the XML schema definitions. Most of the XML parsers can do pretty good job in verifying Schema conformance. There is no standard way for validating flat files.

Step 6: Establish Database to XML Mapping

Pulling data out of database tables and formatting it into XML format could be a tedious task unless there is a mapping tool available or the native DBMS supports XML format directly. Many DBMS, such as ORACLE and SQL Server, provide capabilities to convert data into XML format using extended SQL commands.

Step 7: Create Stored Procedures

This task is closely related to the Data-XML mapping. Note that "each" or "a" data field retrieved using the procedure should have one-to-one mapping to the predefined XML schema. Once created, the stored procedure name should be listed in the GetServices Method (with Query as the serviceType).

Step 8: Establish Solicit

Solicit is an extension of the Query method. Partners must decide when business rules for when to honor a Query vs. Solicit apply.

2.6 Test Tools

Currently, EPA is providing two test tools for Node implementers. Use them early and often! To use the test tools, Node implementers must have their Nodes externally available (accessible from an external client), a valid WSDL file accessible from an external client, and at least one Web method functioning.

The first test tool dynamically builds a client, in a Web browser, using the Node's WSDL file. With this client, you will be able to see both request and response messages, and interact with Node services without a single line of programming. Hence it is an ideal tool for testing and validating Node implementations. This test tool is available at http://epacdxnode.csc.com . To generate the test client follow the directions on this page.

There is also an automated test tool and a description of the test scenarios it uses at http://epacdxnode.csc.com/testcase.html. This test tool runs automatic tests to see if your Node implementation complies with the Protocol and Specification. This tool invokes each of the Web Methods and indicates whether your Node has passed or failed. Passed indicates that it has successfully serviced the Web method compliant to the Protocol and Specification. This tool also provides performance measures– the time each Node took to service each Web Method.

Passing all of the tests provided by the testing tools indicate your development is on the right track, but it does not guarantee your Node will interoperate. The real

test of your Node comes during a real interaction with several other state nodes and CDX. For example, during the Node 1.0 project, two Nodes that had independently passed the testing tools were not able to communicate because of a spelling error in a line of code. The testing tools are maturing with Nodes and new testing tools will be developed as Network use increases.

2.7 Node 1.0 Lessons Learned

The lessons learned are divided by Network Technologies; Node-building and Node usage; and security.

2.7.1 Lessons Learned about the Network Technologies

Web Service Technologies are important but not yet fully utilizable on the Network

- UDDI is important to the Network but is not yet available for Network Partners
- WSDL is imperative to the Network, but cannot be fully utilized because of immaturity of Node tools
- Immaturity in Node tools forces the use of RPC encoded SOAP messages (SOAP Toolkits could not handle mixed encoded messages)
- Migration to subsequent Web Standards will be necessary but not immediate as tools cannot yet handle the increased functionality (SOAP 1.2, WSDL 1.1)

WSDL is a core Network Technology

- WSDL files must be made available to all potential Network Partners
- Using the Network WSDL file to build a Node is mandatory

The Node 1.0 project reconfirmed that SOAP 1.1 is an appropriate XML messaging protocol.

2.7.2 Lessons Learned about Node-Building and Node Usage

- DIME implementations differ by platform and pose challenges for toolkits without native support.
- AXIS toolkit successfully implemented and shared between Node Partners
- Node builders MUST perform a comprehensive analysis of their SOAP toolkit. Not all SOAP toolkits are built equal.
- Node builders should be able to use Node Code from similar Nodes. The Node 1.0 team is creating Demonstrated Node Configurations.
- Client generation is not automated but is important for Network Partners who intend to use the Node to initiate service requests.
- Mapping the ‘back-end’ systems to the Schema is one of the most challenging and time intensive tasks in preparing a Node to exchange information.

- Availability of 'expert' consultation at crucial Node decisions is important and imperative to assure interoperability.
- The Protocol and Specification does not define how Nodes should be used to Flow information.
- Creation of the Flow Configuration Document and establishing guidelines for Trading Partner Agreements are imperative for flowing information.
- Node usage is not automated and no standard Node user interface is defined by the v1.0 Protocol and Specification. Partners will have to enhance Nodes or create a separate user interface if they intend to initiate service requests.

2.7.3 Lessons Learned about Security

- While the current Protocol and Specification ultimately places the burden of security on each Network Partner, using the NAAS (centralized security provided by EPA) is efficient and cost effective.
- Authorization is needed as soon as possible.
- Authorization and Authentication is sufficient for the first generations of Network Exchanges but will be insufficient as the Network proliferates.

SECTION 3

3.1 Demonstrated Node Configurations

As part of the Node 1.0 project knowledge transfer, participants created Demonstrated Node Configurations (DNCs). DNCs are a combination of a detailed, narrative description of each Node 1.0 Node and any common 'code' that might be useable by other Node builders with similar systems. Through the use of these DNCs, states wishing to deploy a Node on the Network can choose to leverage work that has already been done by implementing one of the Demonstrated Node Configurations, reducing the cost and time of implementation. Node 1.0 participants believe that DNCs serve as legitimate starting points for Node implementers. The DNCs are available for download at the Network Website.

The states that have developed local Node applications/enhancements will also make these available with the DNC. Even if states do not to make use of local Node applications and enhancements, these can serve as a list of areas to address when moving towards full control and automation operation of a Node. Maine will be providing their basic Node code in the DNC, and will additionally be providing information and code for their:

- Local Authentication and Authorization Service;
- Local UDDI registry;
- Efforts to implement Dynamic Invocation Interface (DII);
- Scheduler development efforts; and
- Logging functionality

3.2 Node 1.0 Participant Implementations

The Node 1.0 participant implementations are described in detail in the Demonstrated Node Configurations available on the Network Website. The purpose of this section is to give readers a snapshot of each implementation, provide a short rationale for middleware selection, and an idea of each Partner's intended Node usage.

Table 5: Node 1.0 Participant Implementation Description

Node	Middleware	Rationale for Selection	Anticipated Node Usage
Delaware	.Net 1.0 / Win 2000 Advanced Server / IIS Database Hardware: Dell PowerEdge Dual XEON 700 MHz RAID 5 3x9GB Drives 1 GB Memory	• Consistent with IT Strategy: Production data in SQL Server and already in the process of migrating to .Net for their data entry and display. So far .Net is providing all the tools needed to develop a web service.	Exchange FRS and NEI data as part of the 1.0 phase of the project. In the future, they plan to use their Node to exchange Beach Monitoring Data with Earth 911, NPDES, Air emissions and RCRAinfo data.
Maine	Oracle 9i Application Server (9iAS) 9.0.2.1.0	• Industry Standard: A comparative review in the 9/2/02 eWeek magazine rated Oracle 9iAS Release 2 middleware number one when compared with its most direct competitors, IBM's WebSphere and BEA's WebLogic. • Automated Coding: Oracle 9iAS R2, a.k.a 9.0.2, is a bundle of software functionality sold as a labor-saving, cost effective unit. It includes several wizards that relieve users from doing most/all of the coding in SOAP and WSDL. The past experience of Florida DEP with Oracle was that the most time-intensive part of setting up their beta Node using 9iAS R1 dealt with SOAP and WSDL code which they had to write manually. Oracle included wizards and other features in Release 2 to simplify these and other tasks. • Consistent with IT Strategy: Maine's Bureau of Information Services, offering in-house operations, network, and development services to State agencies, is an Oracle database and development shop. Deciding to use Oracle middleware is consistent with its IT strategy and is consistent with their decision to limit diversity in the infrastructure supported due to small staff size and a limited budget. The considerable benefits of a single-vendor solution are attractive, for e.g., less software administration, faster throughput, no integration issues with various software packages and the database, lower cost as Maine DEP already owns the software. • Vendor Experience: Oracle has directly relevant experience through their work with the Florida DEP beta Node and related interactions with EPA. They have demonstrated that they are highly motivated to be a leading player in this arena. They have relatively local staff that is capable of developing the Node within the Node 1.0 timeframe and beyond the Node 1.0 requirements.	Exchange FRS and NEI data as part of the 1.0 phase of the project. In the future, their Node will serve as an Enterprise XML server for other State Agencies. Maine DOT and Federal DOT plan to use it to track Hazardous Waste movement in real time. The Maine Revenue Service could be another potential user in collaboration with the IRS.
Mississippi	.Net	• Consistent with IT Strategy: Mississippi DEQ's network infrastructure consists of Microsoft Windows 2000 and Microsoft Internet Information Server (IIS). Recent in-house development projects have used ASP and hence .NET is the logical progression of their efforts and is consistent with their long-range goals.	Exchange FRS and NEI data as part of the 1.0 phase of the project. In the future, they plan to use their Node to exchange PCS data through IDEF, RCRAinfo and Beach Monitoring data.

Nebraska	XAware XA-Suite using a Tomcat 4.1.12 server	• Past Experience with middleware: Xaware was the company and product used in the Alpha Node project by all of the state participants and in the Beta Node project by two participants. The product has worked well for Nebraska DEQ in the first two projects. • Consistent with IT Strategy: Their Integrated Information System is housed on an AS/400 using DB2 and Xaware has demonstrated a successful solution to Node implementation.	Exchange FRS and NEI data as part of the 1.0 phase of the project. In the future, they plan to use their Node for all EPA Type I flows and will explore the possibility of using their Node for Facility to State submissions. Nebraska DEQ is also developing preliminary plans to use the Node for exchanges with Nebraska Emergency Management for Homeland Security.
New Hampshire	BizTalk Server 2000, .NET Programs	• Industry Standard: BizTalk is a Microsoft product. NH DES selected an industry standard that will lead to consistency as they go forward with future data flows. In addition because of the rapid progression of the XML market, NH DES feels that finding personnel to continue the development of additional flows will be easier when using BizTalk rather then other products. • Application integrator: BizTalk is an application integrator and as such offers advantages over other products such as a queuing system that enables flows to be de-hydrated, taken offline then re-hydrated or put back on-line. • Orchestration: Orchestration is a tool that will allow NH to build a complete flow. Not only will the XML request be fulfilled but also based off the status of that process other processes will be executed. A good example is Water Supply transfers data to EPA. When that process is completed successfully. Water Supply will go run additional reports that will update NH database. With BizTalk this entire process can be automated and track with built in scheduling. • Scalability: Scalability is an important consideration. BizTalk offers a clustering of its servers where additional servers can be brought on-line and integrated into the IIS site. Mapping of ports to individual flows, meaning if the port is paused it won't disrupt other data flows.	Exchange FRS and NEI data as part of the 1.0 phase of the project. In the future, they plan to use their Node to exchange Air Toxics, Beach Monitoring and Beach Water Quality data.
New Mexico	IBM WebSphere v4.05, AXIS toolkit, using XAware data integration components	• Industry standard: The New Mexico Environment Department (NMED) evaluated web application server software in the spring of 2002; WebSphere earned the highest ranking in the evaluation. WebSphere Studio Application Developer (the WebSphere development environment) was rated the best of all competing development environments in their evaluation. WebSphere fully supports XML and provides many tools to integrate XML with enterprise data and features a powerful administrative console as well as performance and tuning analytics. • Consistent with IT Strategy: WebSphere combines a highly rated web services platform with support for Java 2 Enterprise Edition (J2EE) applications the agency also needs.	Exchange FRS and NEI data as part of the 1.0 phase of the project. In the future, they plan to use their Node to provide Data access to public and industry and share Research studies on Air Pollution with other States.

		NMED has set up the network connections from WebSphere to our enterprise database for dynamic access; NMED can leverage these connections for Node purposes. The State of New Mexico web portal will run on WebSphere. NMED will have opportunities to leverage expertise and resources supporting this application. • Competitive Pricing: Cost of ownership shows that WebSphere is competitive with other web application servers.	
Utah	Sybase EASserver - Advanced model 4.2.2	• Consistent with IT Strategy: The use of Sybase EAServer conforms to state and departmental standards established for middleware. Sybase EAServer is currently being used for web enabling existing Sybase PowerBuilder applications. The adoption of Sybase EAServer for the Utah Node allows UDEQ to leverage existing hardware, software, and staff technical skills. Only one platform must be supported for the Node and for N-tier applications. Because EAServer has been adopted by a number of other state agencies, training costs and technical expertise can be shared with other agencies.	Exchange FRS, NEI, and RCRAInfo data. Future use of the Node will include participation in the CDC Environmental Public Health tracking project as well as state projects currently being planned.
CDX	BEA WebLogic	• Industry Standard: Given the unique EPA requirements of CDX in terms of scale and diversity of transactions, EPA has elected to use BEA WebLogic as its middleware. WebLogic is very stable, has excellent support and proven scalability. This platform, which has full open standards compliance with J2EE support, is consistently one of the fastest to support the latest specifications. With extensions for web service development, security, and enterprise application integration, BEA provides industry leading performance benchmarks.	Exchange FRS and NEI data as part of the 1.0 phase of the project. Future use will include support for additional data flows [TBD].

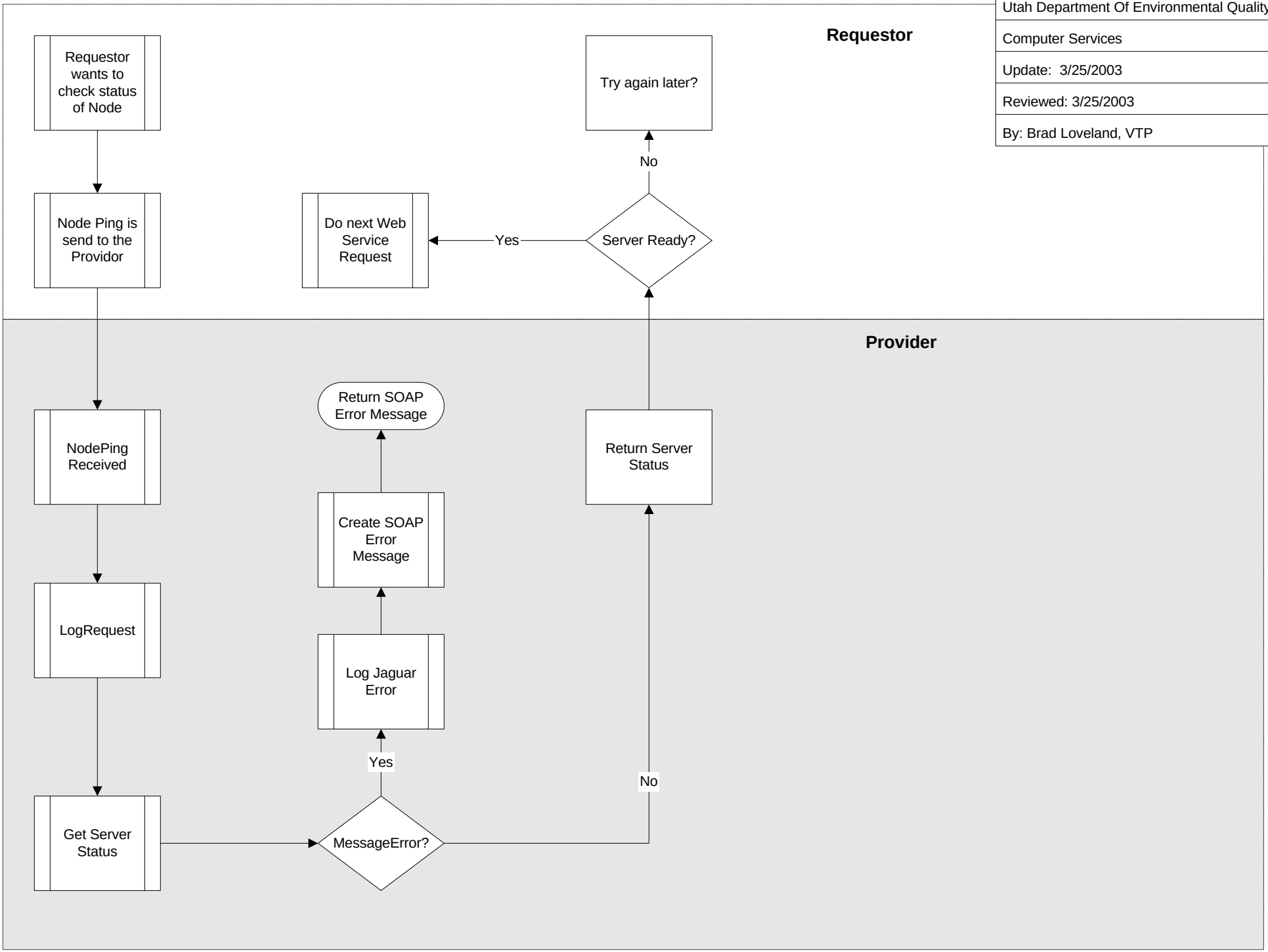
Exchange Network Node Implementation Guide v1.0

Appendix: Process Flow Diagrams

04/25/03

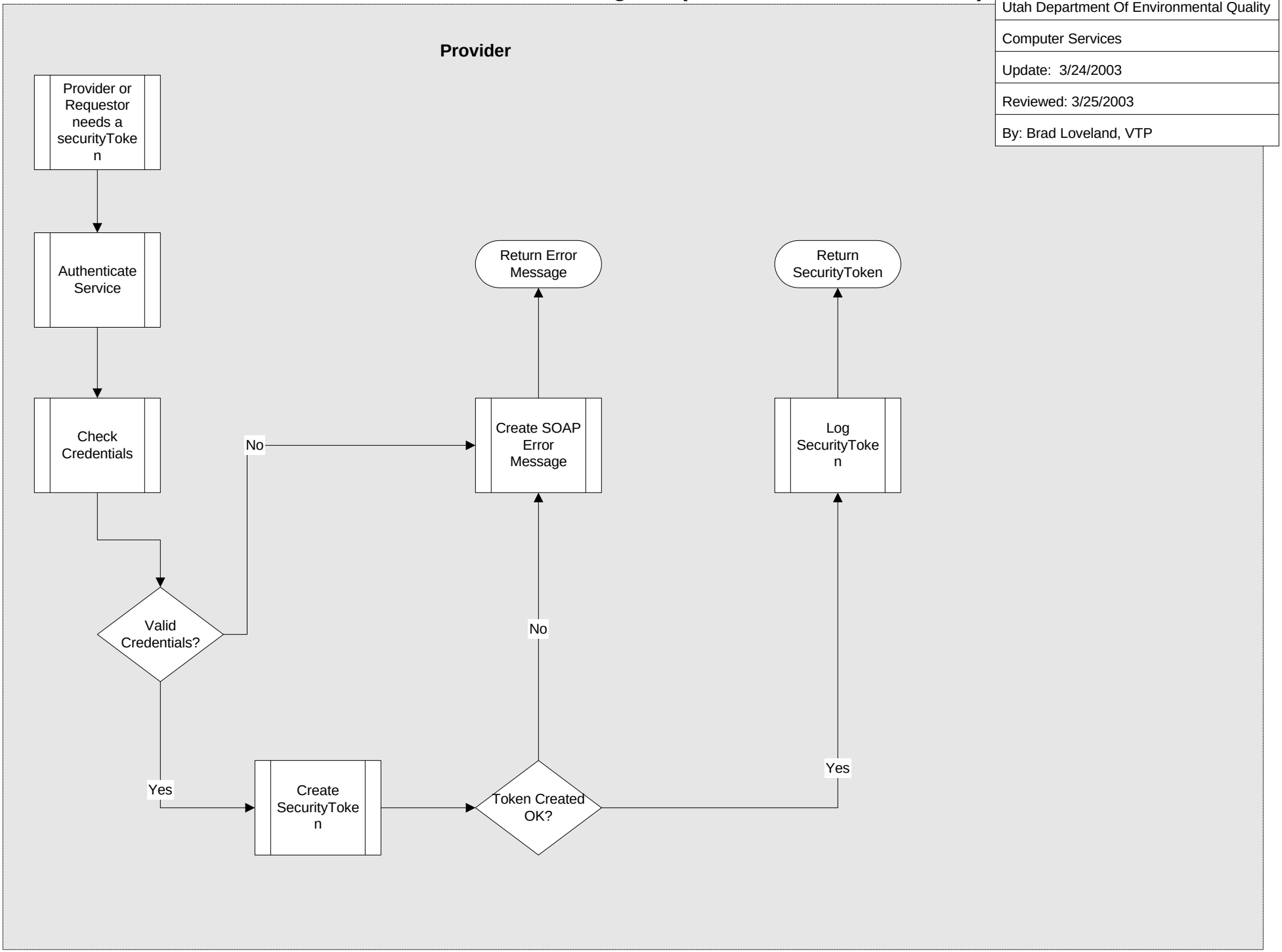
High Level NodePing Process Flow Diagram

Utah Department Of Environmental Quality
Computer Services
Update: 3/25/2003
Reviewed: 3/25/2003
By: Brad Loveland, VTP



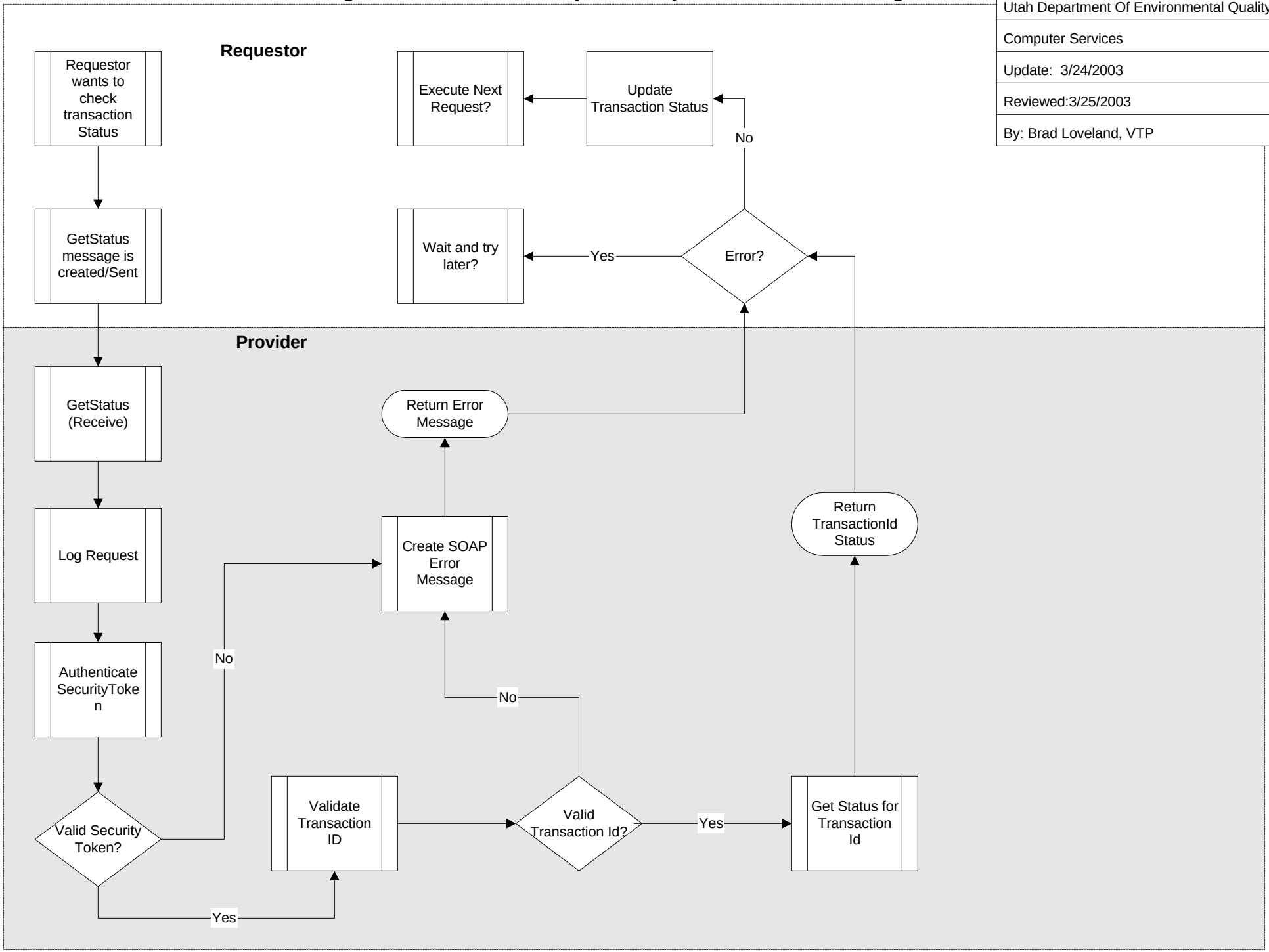
Authenticate Process Flow Diagram (For Local Authentication)

Utah Department Of Environmental Quality
Computer Services
Update: 3/24/2003
Reviewed: 3/25/2003
By: Brad Loveland, VTP



High Level GetStatus (Receive) Process Flow Diagram

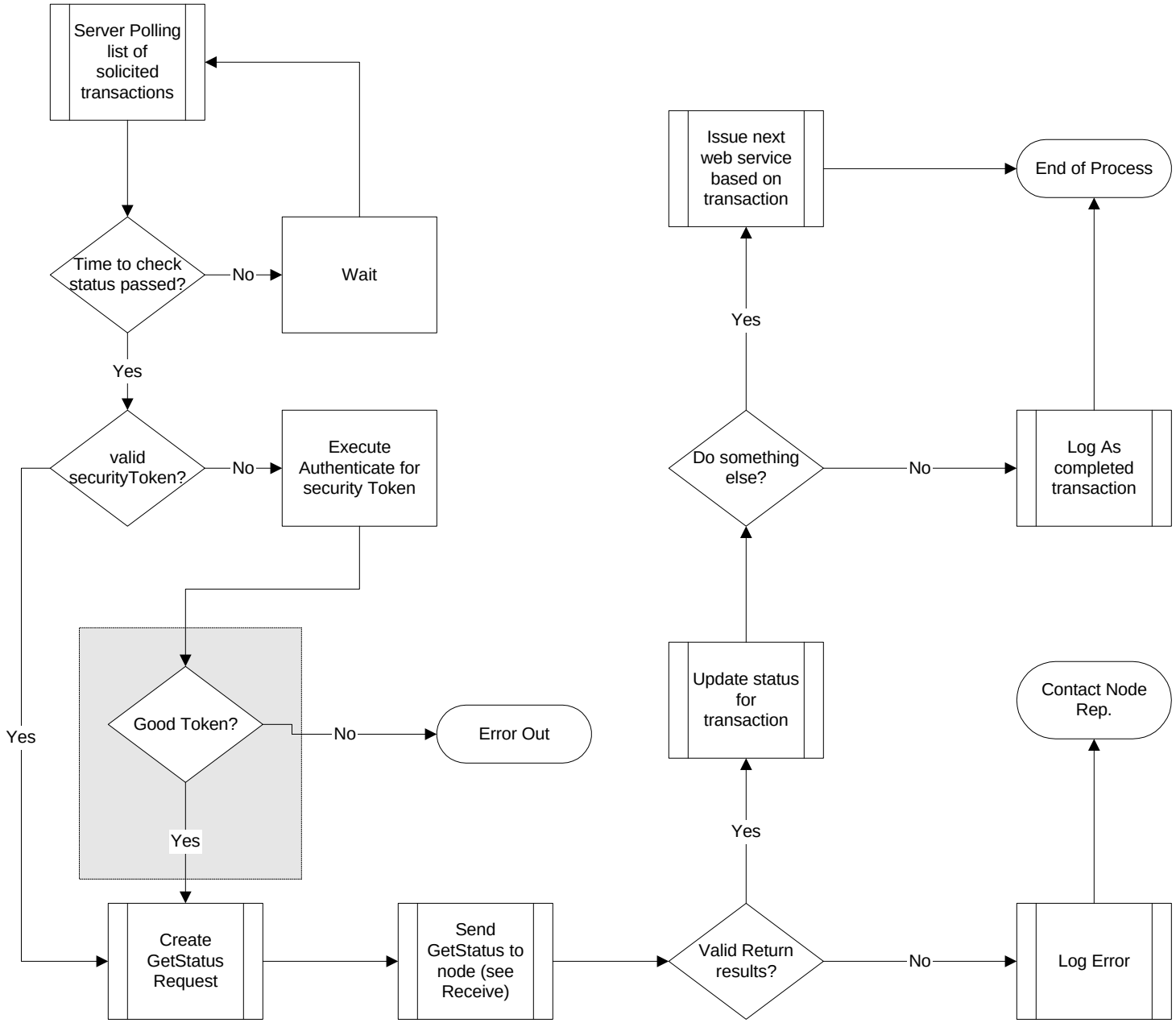
Utah Department Of Environmental Quality
Computer Services
Update: 3/24/2003
Reviewed:3/25/2003
By: Brad Loveland, VTP



High Level Get Status (Send)

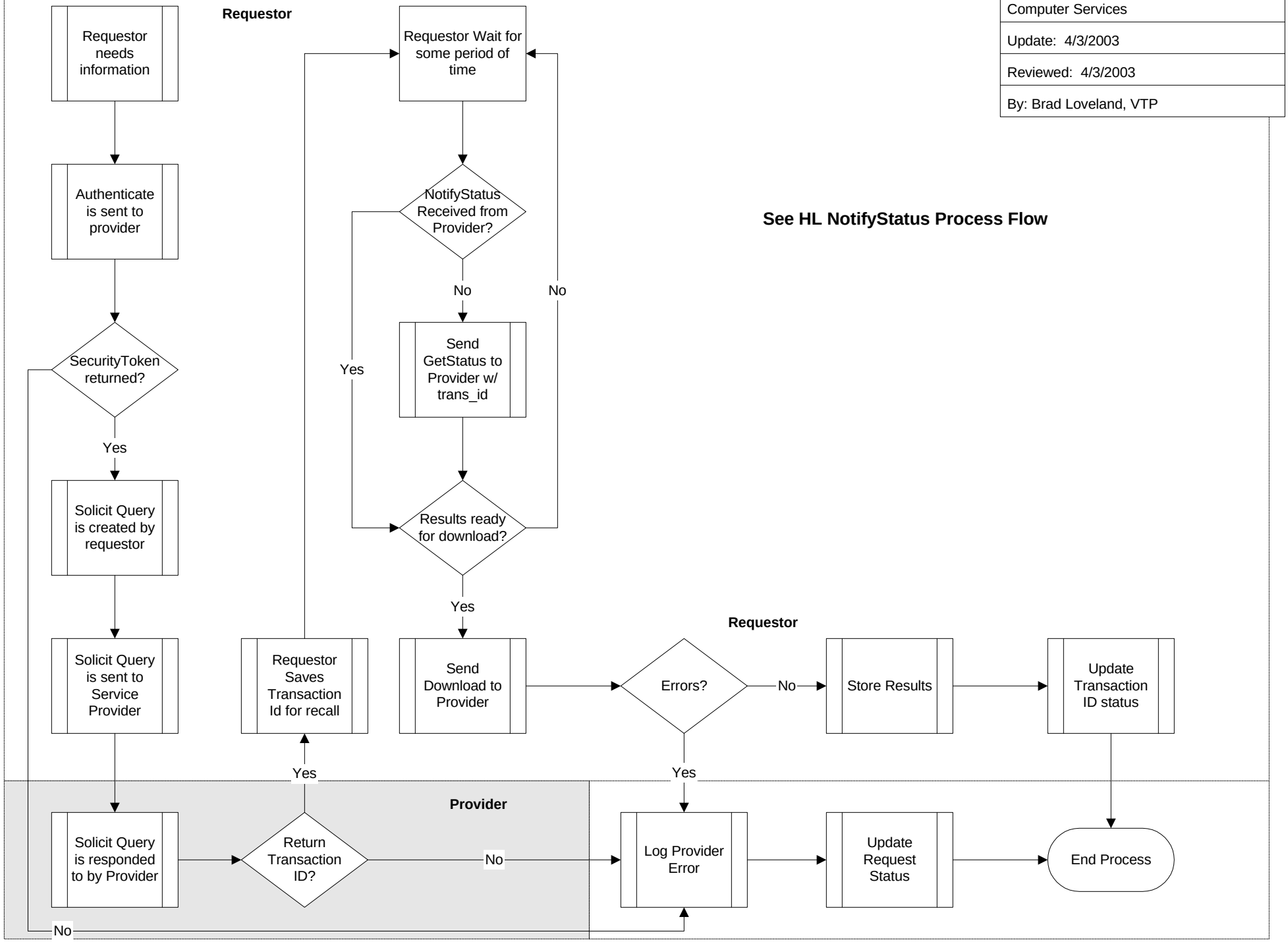
Utah Department Of Environmental Quality
Computer Services
Update: 4/11/2003
Reviewed: 4/11/2003
By: Brad Loveland, VTP

Requestor



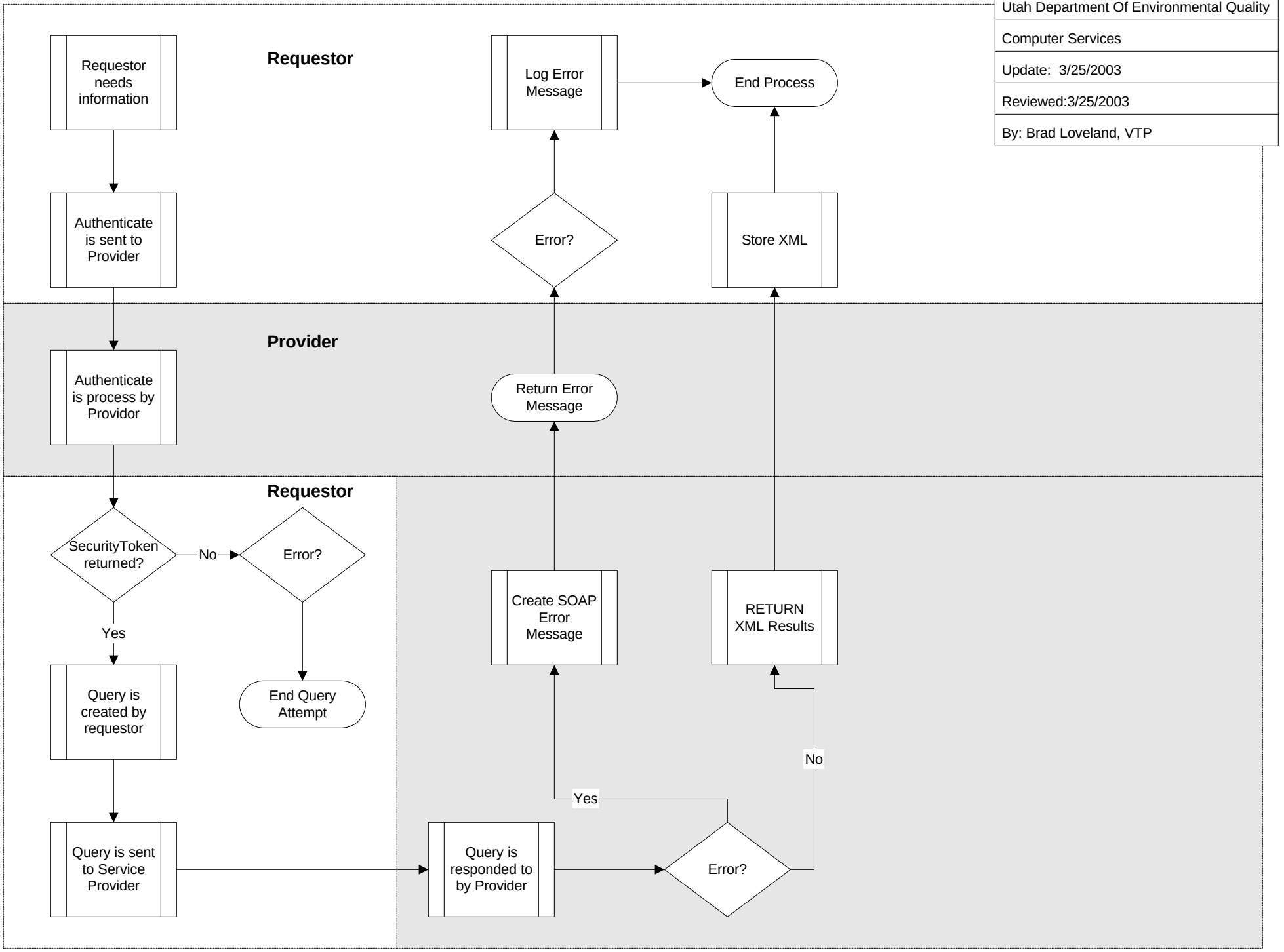
High Level Solicit Process Flow Diagram

Utah Department Of Environmental Quality
Computer Services
Update: 4/3/2003
Reviewed: 4/3/2003
By: Brad Loveland, VTP



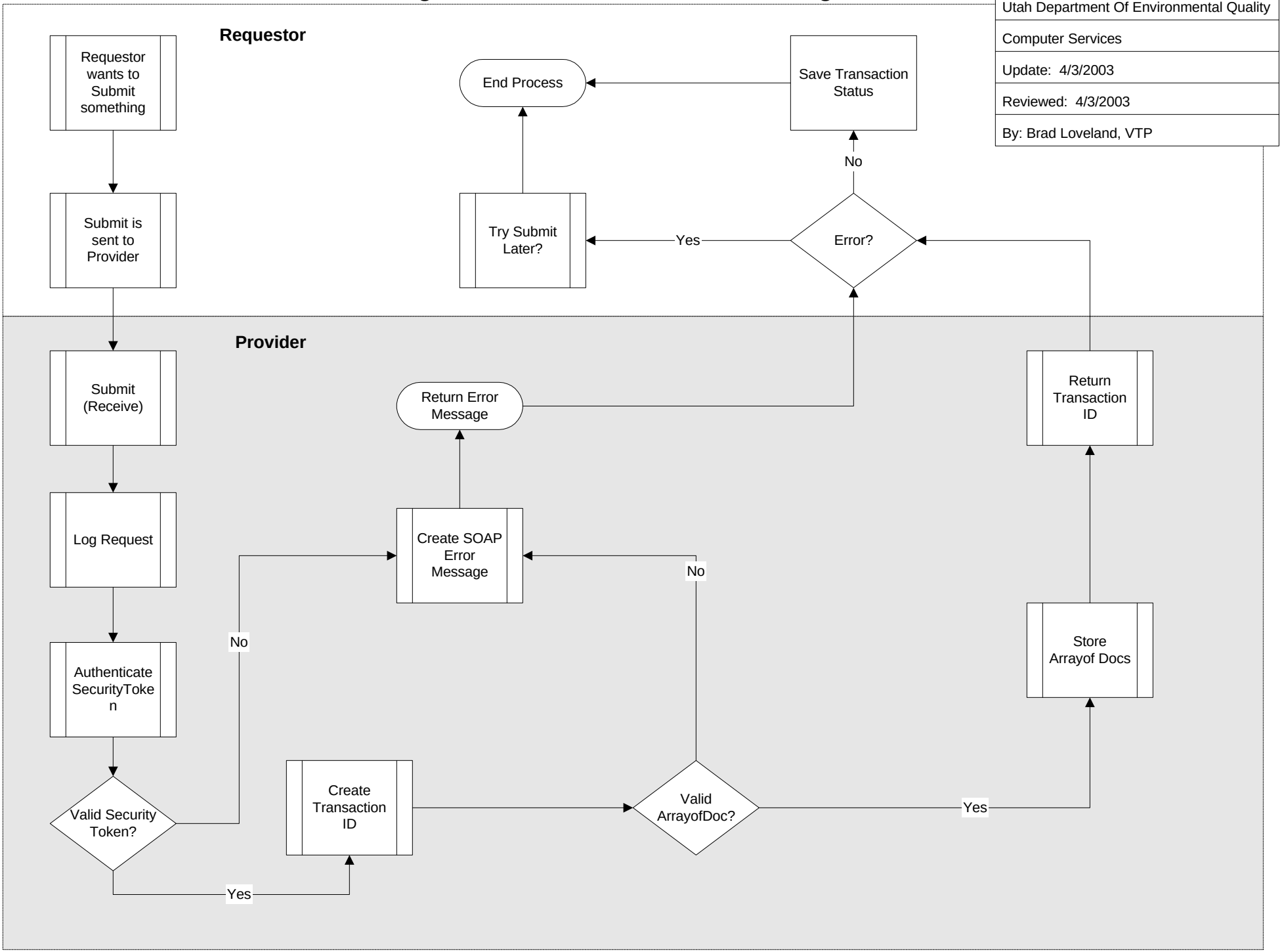
High Level Query Process Flow Diagram

Utah Department Of Environmental Quality
Computer Services
Update: 3/25/2003
Reviewed:3/25/2003
By: Brad Loveland, VTP



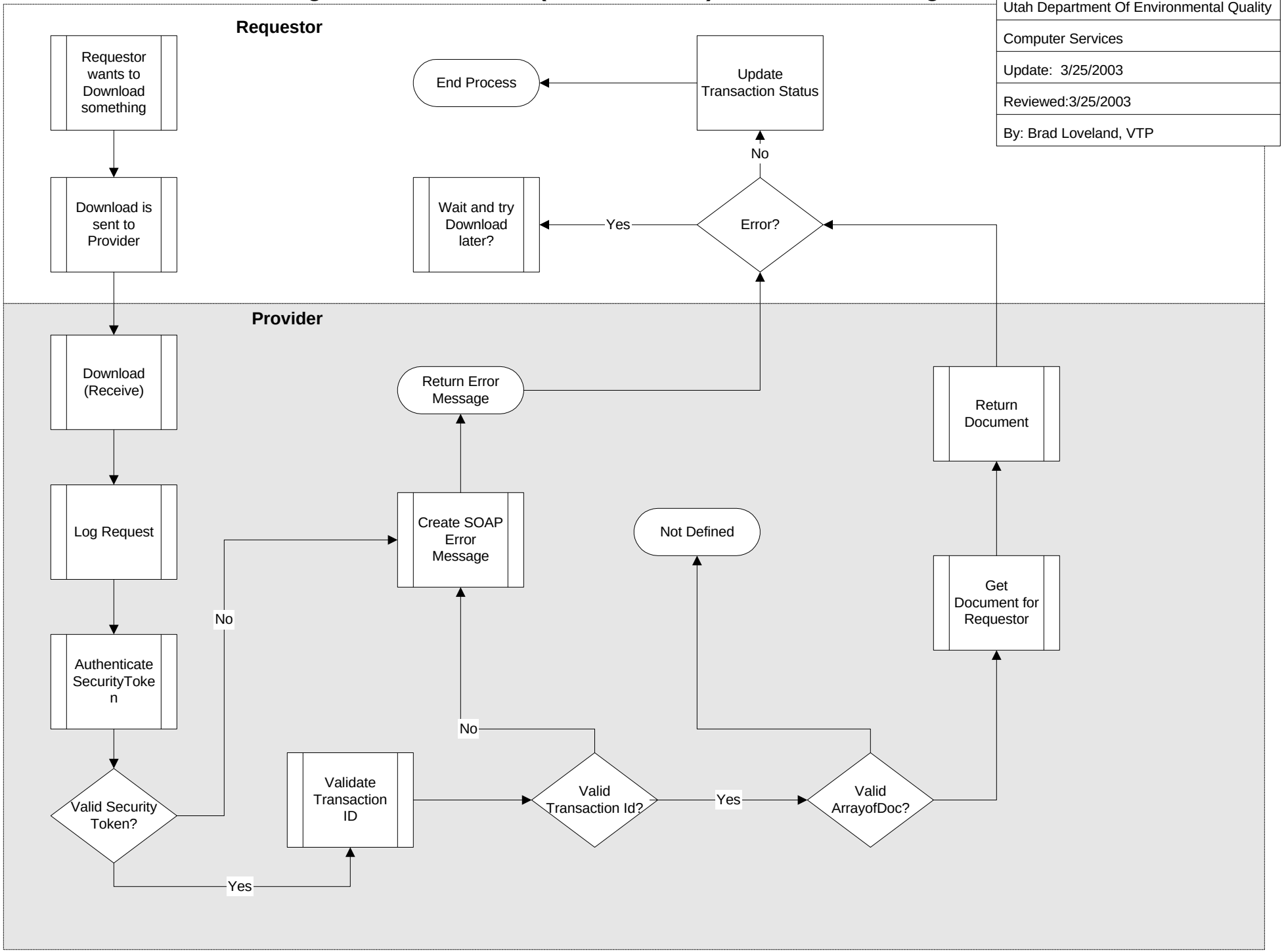
High Level Submit Process Flow Diagram

Utah Department Of Environmental Quality
Computer Services
Update: 4/3/2003
Reviewed: 4/3/2003
By: Brad Loveland, VTP



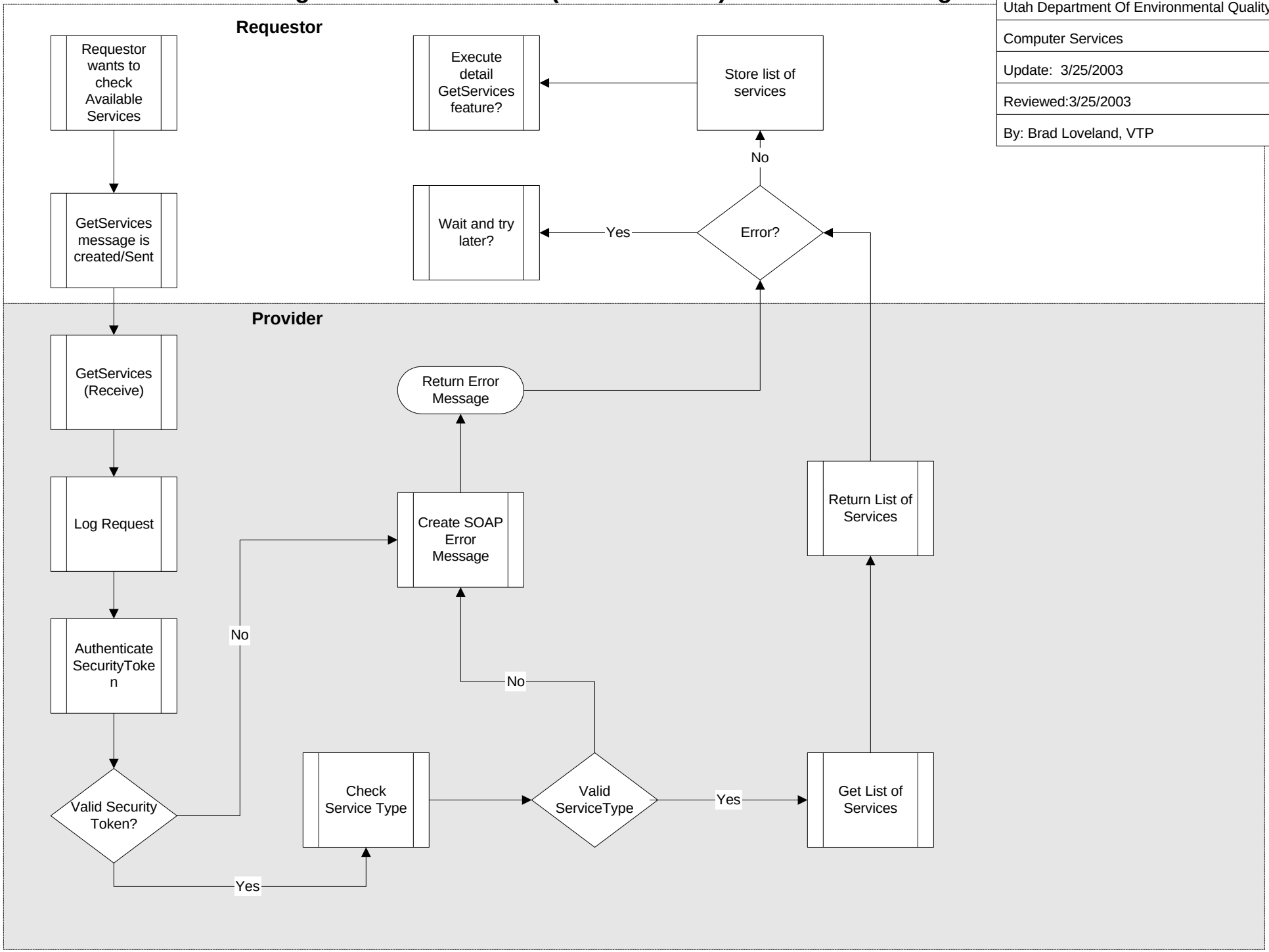
High Level Download (Send/Receive) Process Flow Diagram

Utah Department Of Environmental Quality
Computer Services
Update: 3/25/2003
Reviewed:3/25/2003
By: Brad Loveland, VTP



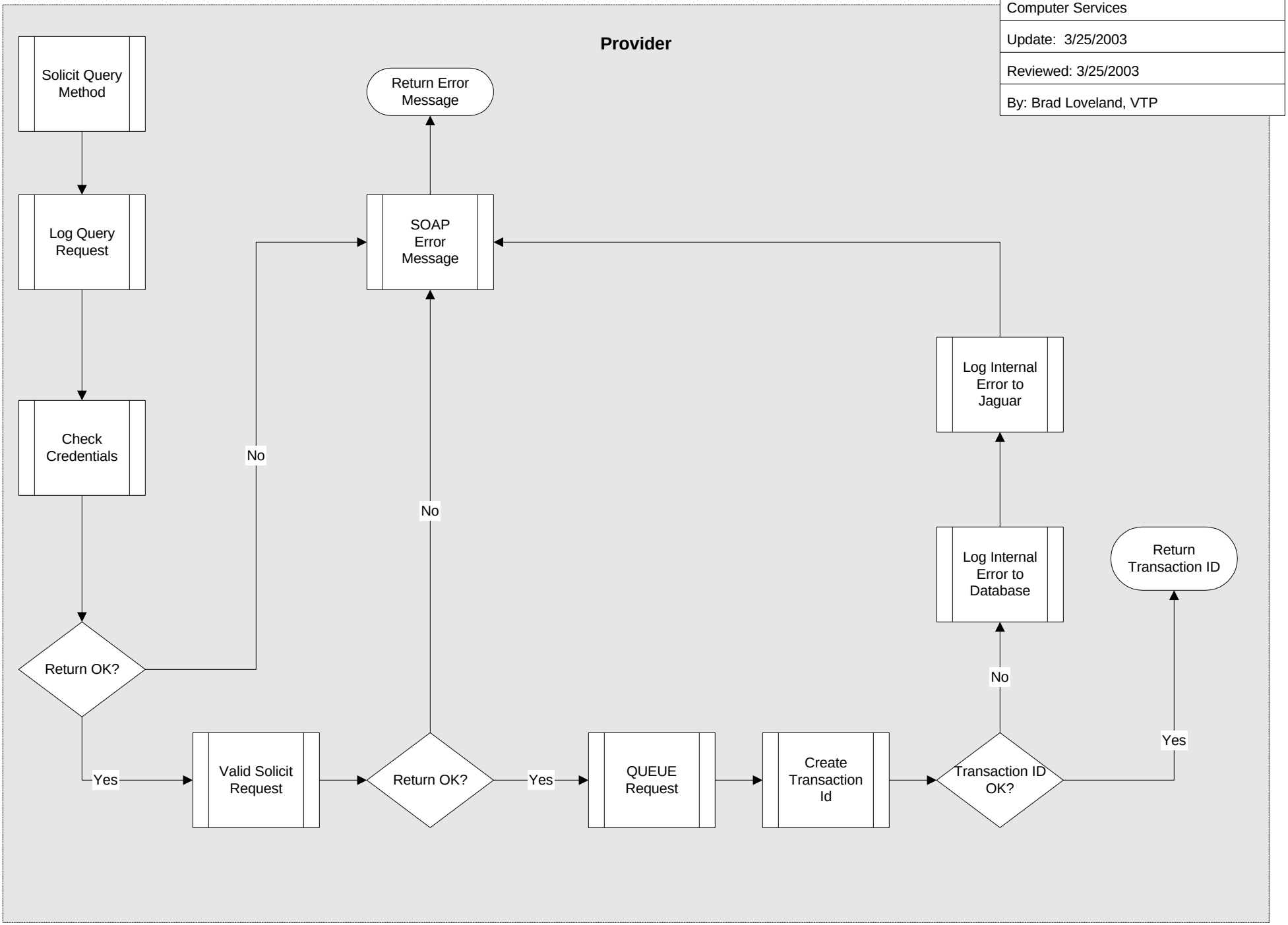
High Level GetServices (Send/Receive) Process Flow Diagram

Utah Department Of Environmental Quality
Computer Services
Update: 3/25/2003
Reviewed:3/25/2003
By: Brad Loveland, VTP



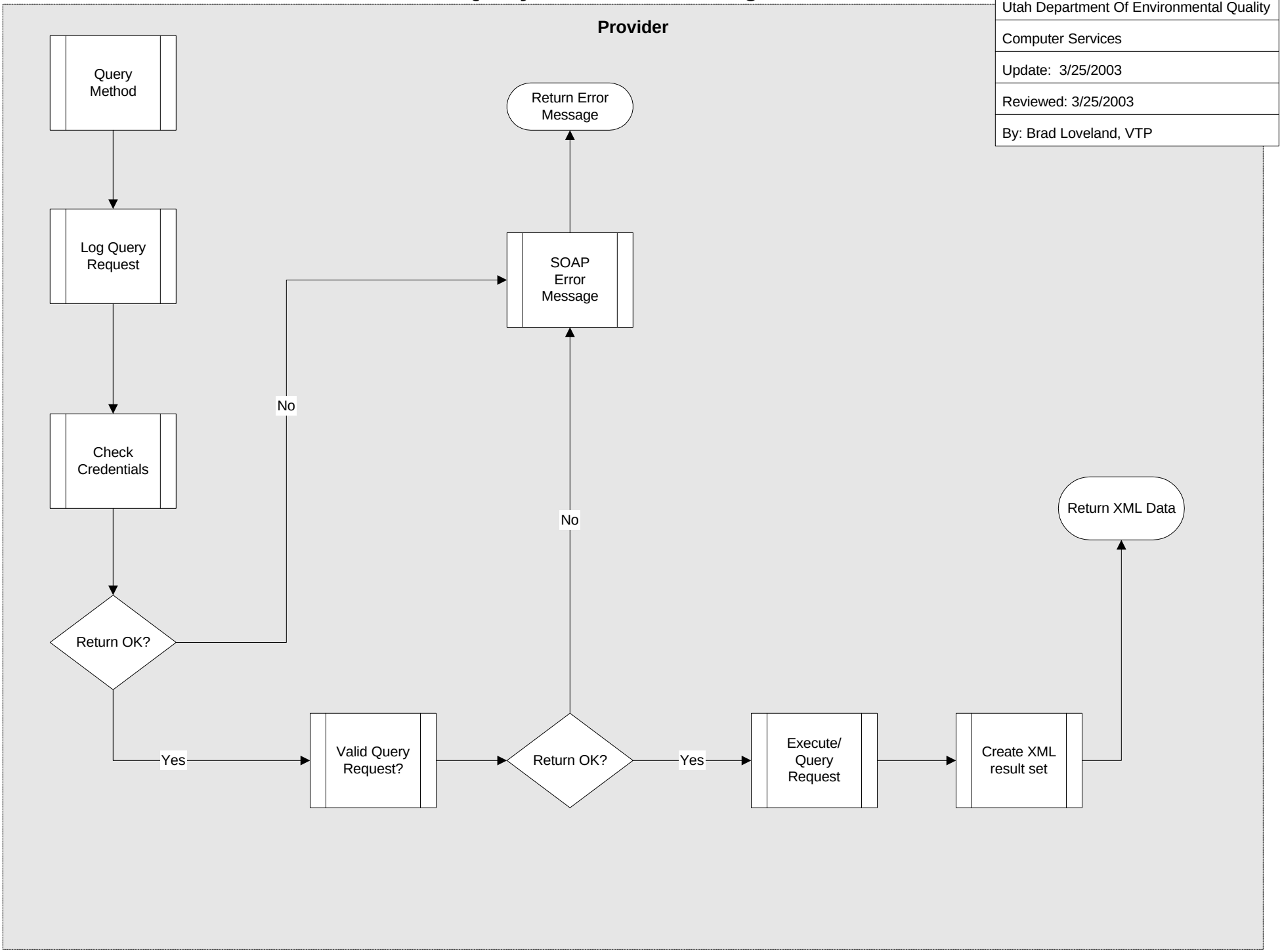
Solicit Process Flow Diagram
** Lightly documented future web service

Utah Department Of Environmental Quality
Computer Services
Update: 3/25/2003
Reviewed: 3/25/2003
By: Brad Loveland, VTP



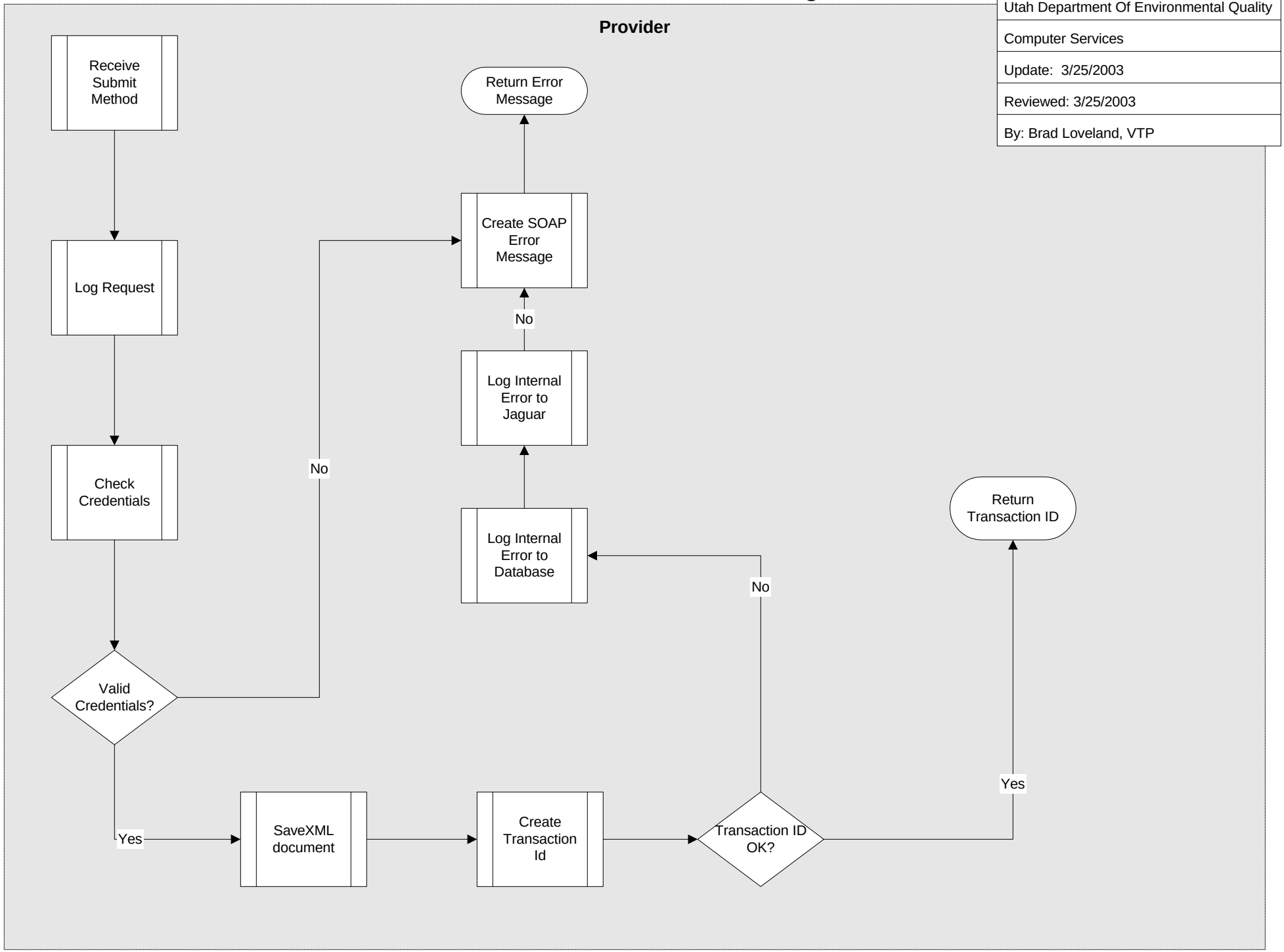
Query Process Flow Diagram

Utah Department Of Environmental Quality
Computer Services
Update: 3/25/2003
Reviewed: 3/25/2003
By: Brad Loveland, VTP



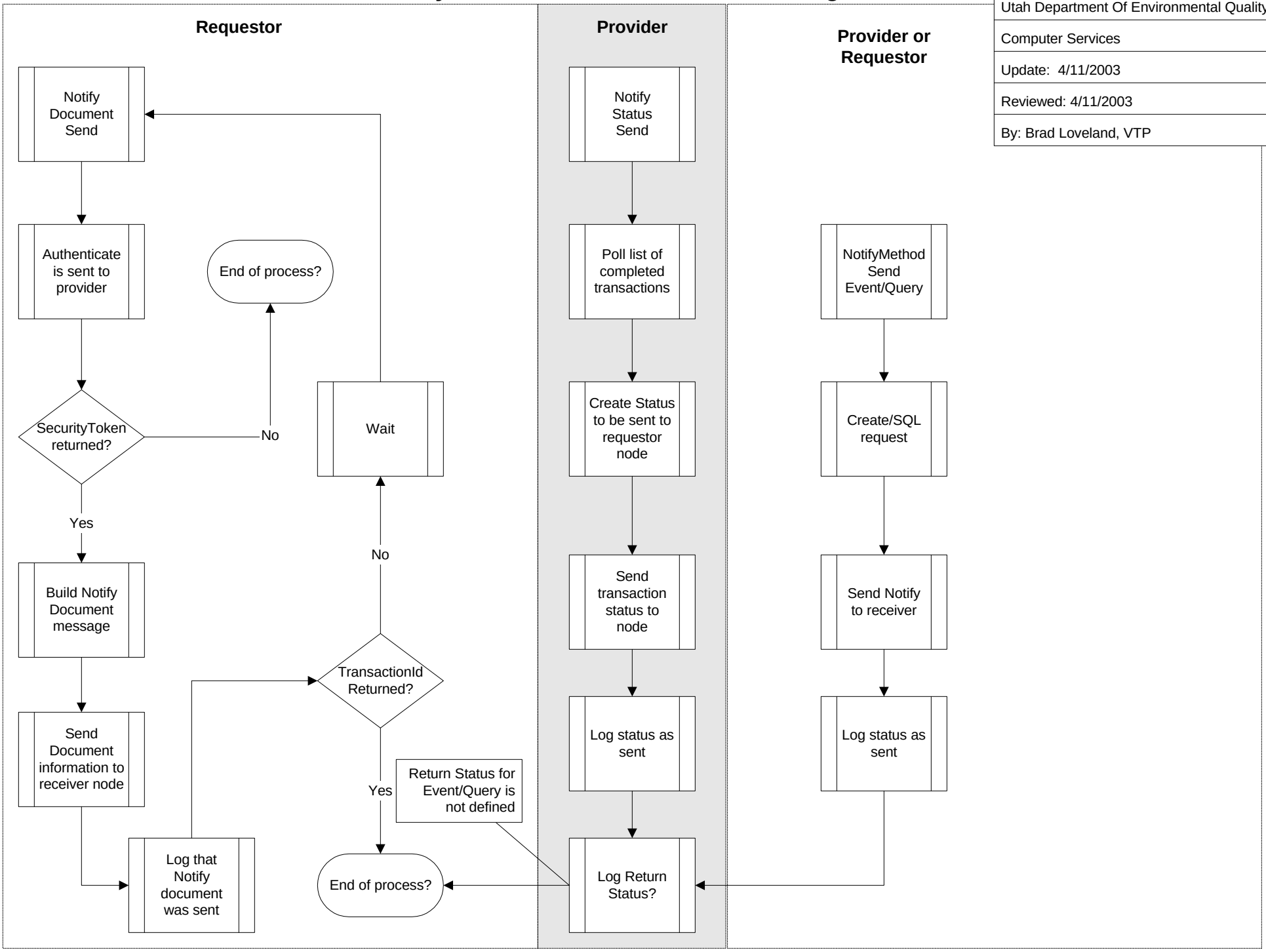
Submit Receive Process Flow Diagram

Utah Department Of Environmental Quality
Computer Services
Update: 3/25/2003
Reviewed: 3/25/2003
By: Brad Loveland, VTP



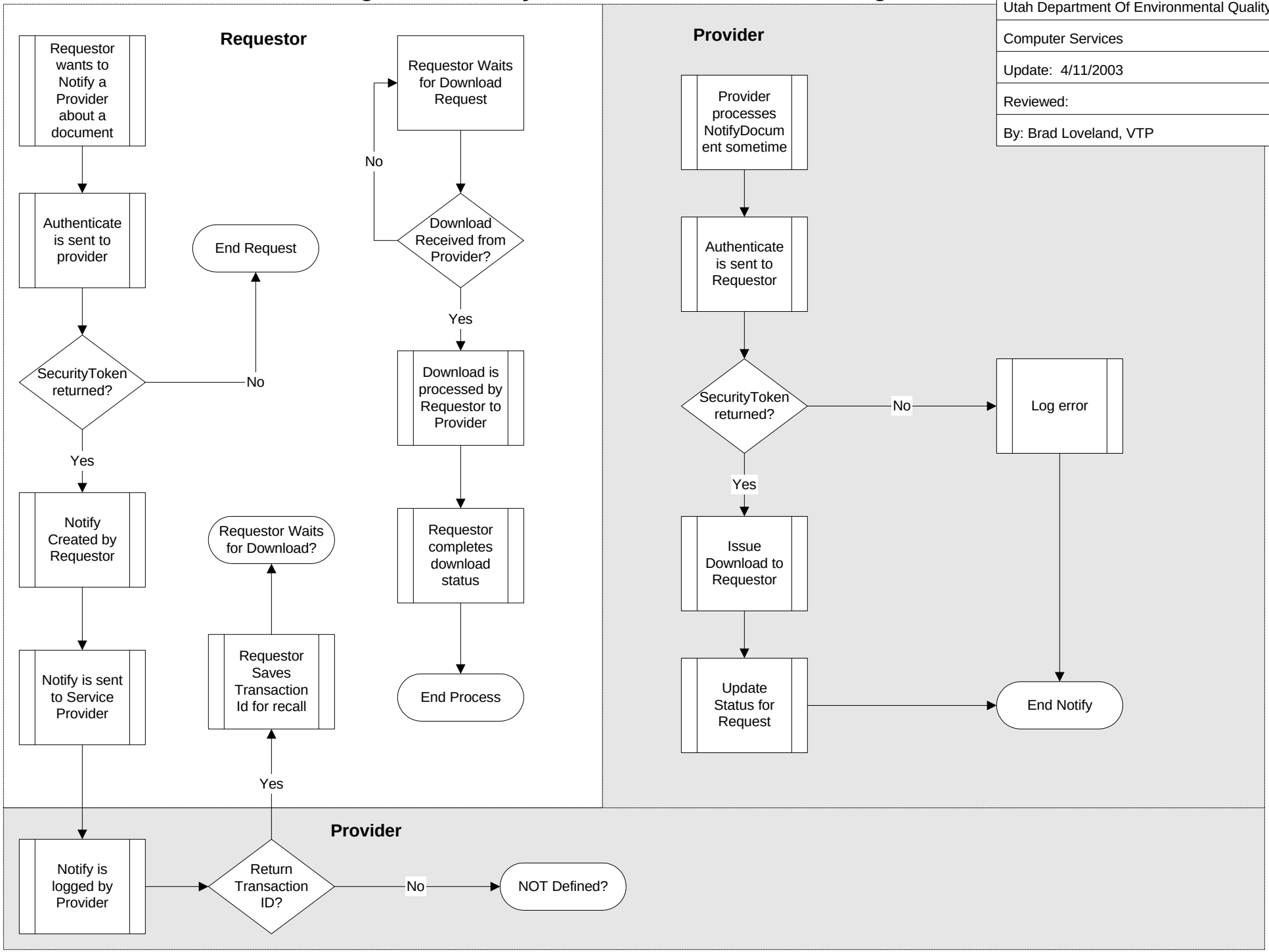
Notify Send/Receive Process Flow Diagram

Utah Department Of Environmental Quality
Computer Services
Update: 4/11/2003
Reviewed: 4/11/2003
By: Brad Loveland, VTP



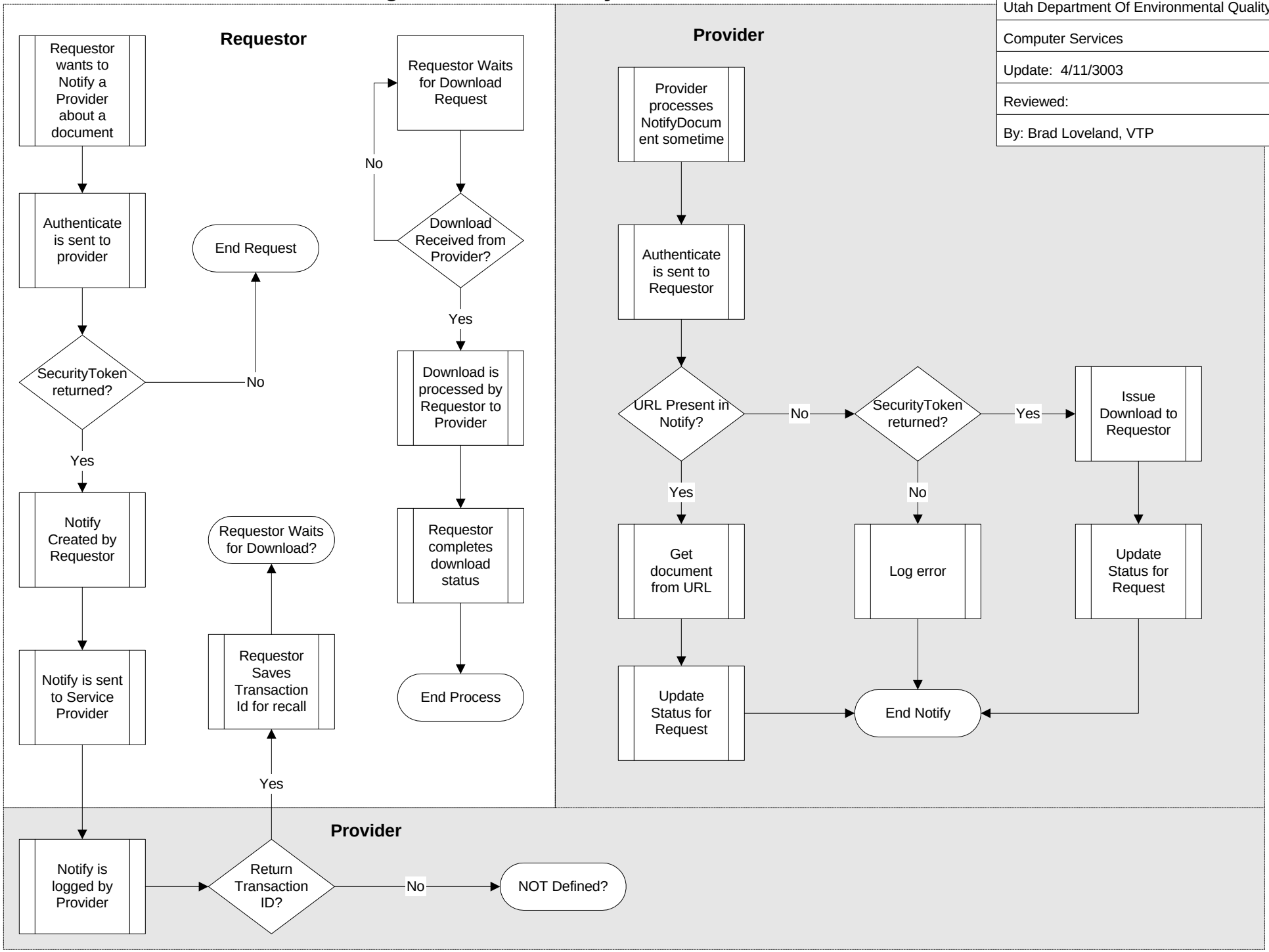
High Level Notify Document Process Flow Diagram

Utah Department Of Environmental Quality
Computer Services
Update: 4/11/2003
Reviewed:
By: Brad Loveland, VTP



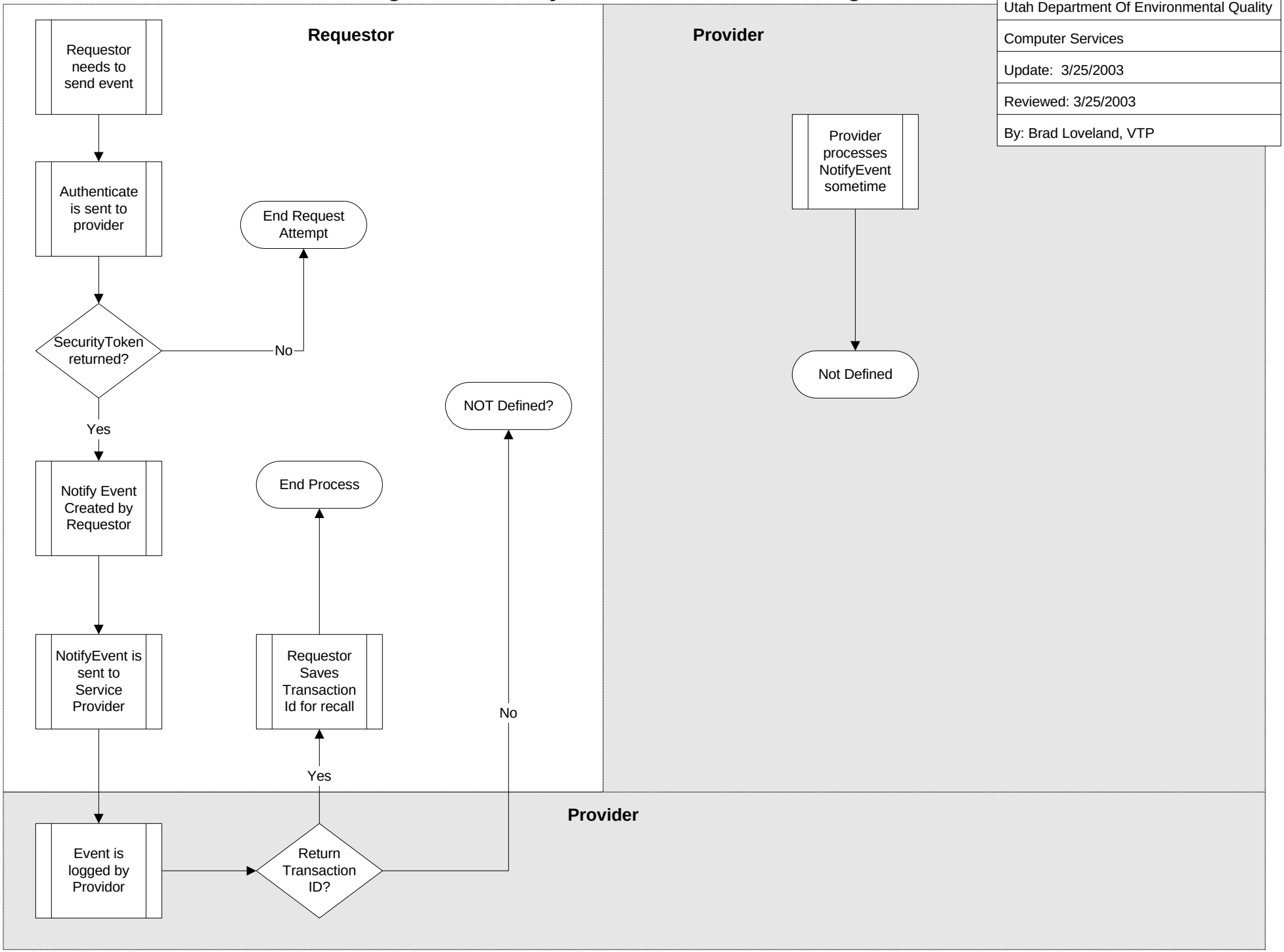
High Level URL Notify Document Process Flow

Utah Department Of Environmental Quality
Computer Services
Update: 4/11/3003
Reviewed:
By: Brad Loveland, VTP



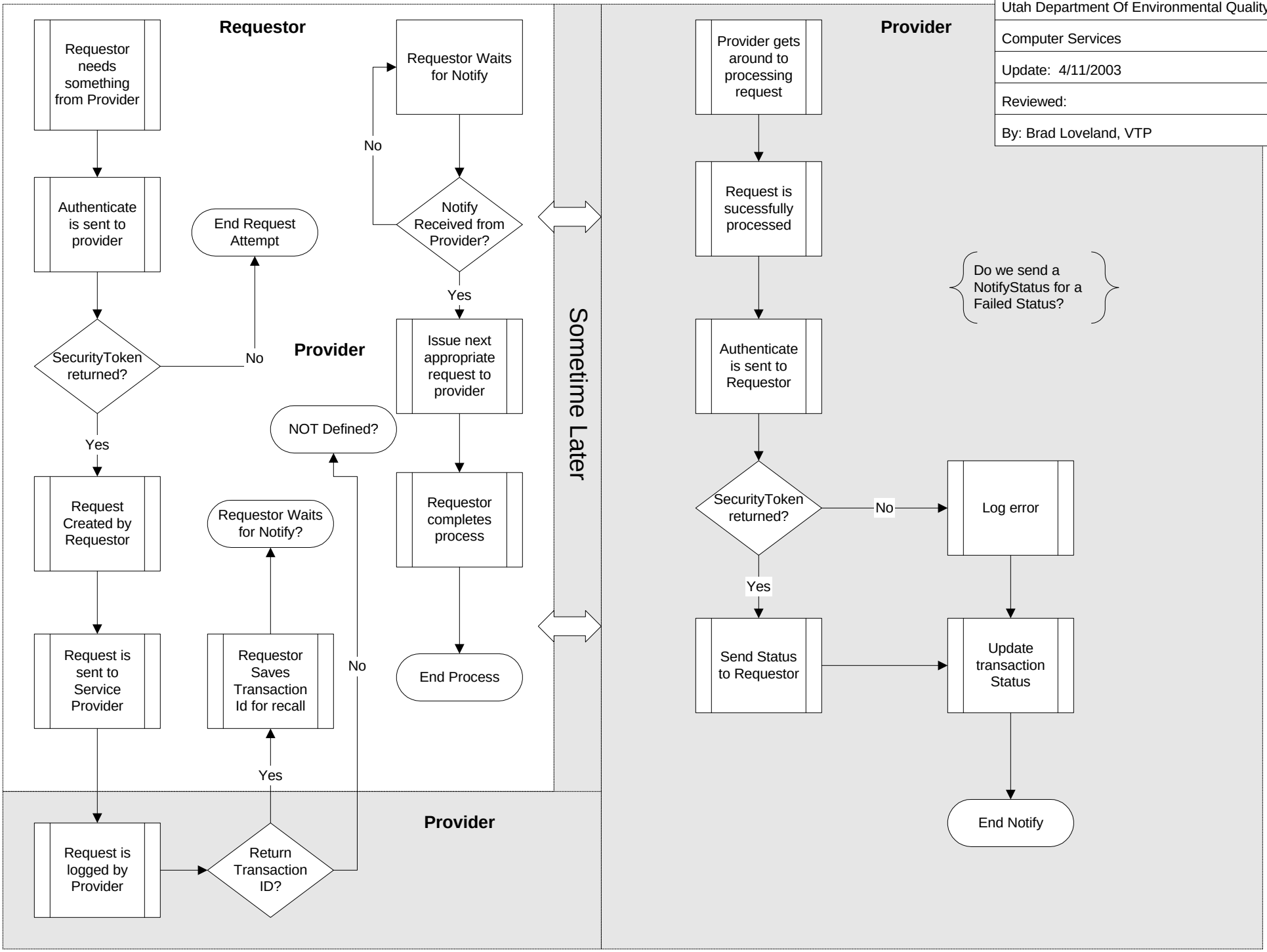
High Level Notify Event Process Flow Diagram

Utah Department Of Environmental Quality
Computer Services
Update: 3/25/2003
Reviewed: 3/25/2003
By: Brad Loveland, VTP



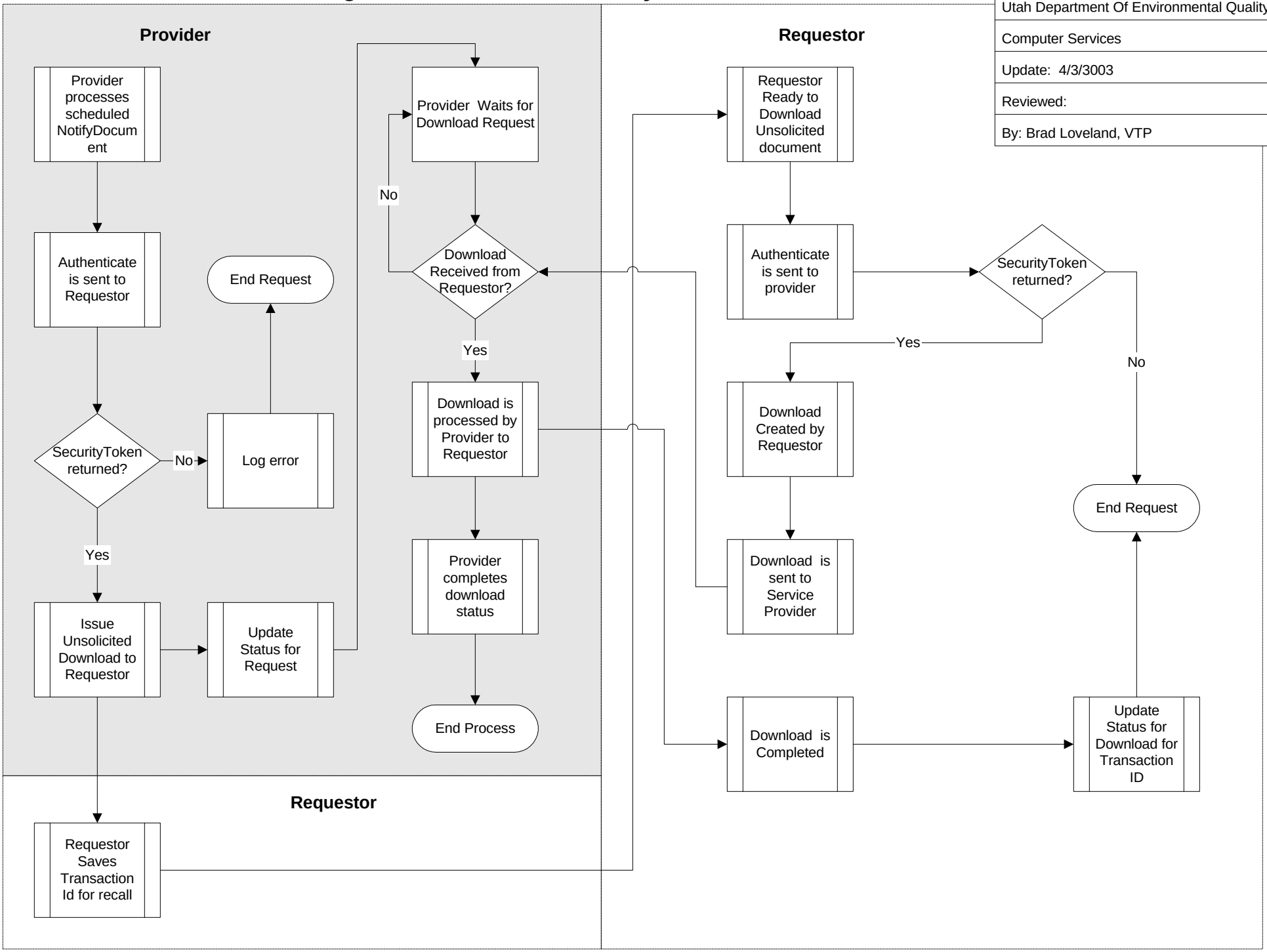
High Level Notify Status Process Flow Diagram

Utah Department Of Environmental Quality
Computer Services
Update: 4/11/2003
Reviewed:
By: Brad Loveland, VTP



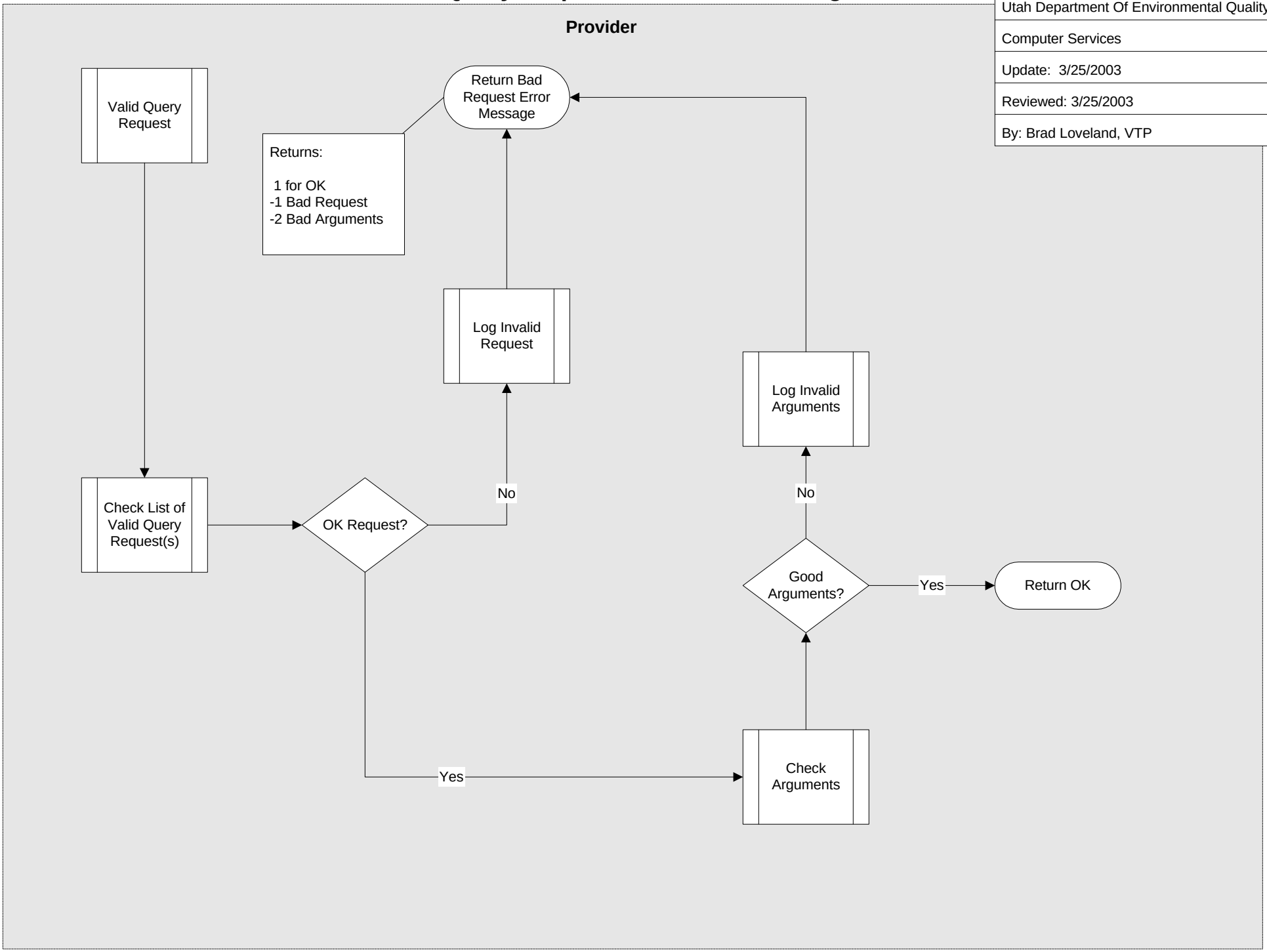
High Level Unsolicited Notify Document Process Flow

Utah Department Of Environmental Quality
Computer Services
Update: 4/3/3003
Reviewed:
By: Brad Loveland, VTP



Valid Query Request Process Flow Diagram

Utah Department Of Environmental Quality
Computer Services
Update: 3/25/2003
Reviewed: 3/25/2003
By: Brad Loveland, VTP



Exchange Network Node Implementation Guide v1.0

Glossary

04/25/03

Glossary of Terms

Address
Parts of a WSDL file describing information for locating the specified service.

Authentication and Authorization
All operations, except NodePing, in the Network Protocol are restricted to registered users only. The restriction requires a user to be authenticated successfully before any other operations can be conducted. Authentication is a process of establishing trust, i.e., who the remote party is and what kind of privilege it has. Authorization relies on a good authentication scheme to protect Network resources. Authentication is also necessary for establishing security policies based on users or user groups. It is also important for creating trusted relationships among Network Nodes (trusted peer relationship) so that highly confidential message exchanges, such as intrusion notifications, are possible between peers.

Bindings
Defines the physical representation of messages on the wire and how they are transferred on the transport layer. In other words, the binding specifies how messages are serialized or deserialized.

Central and Federated Authentications
SecurityTokens may, ideally, be issued through a central authentication server. The tokens are recognized and honored by all participating Network Nodes in a trusted relationship. A central authentication server facilitates single sign-on. Users need only register or login once in order to access services provided by all Network Nodes. In a federated authentication scheme, however, each Node owns and manages a set of user identities locally and each Node is authorized to issue securityTokens. The securityTokens are recognized and honored by other Nodes in a trusted group. A federated authentication scheme is a distributed authentication system where Network Nodes are autonomous in that they have authoritative control over user identities registered at their site.

Client
Any user having access to the information from the Network, typically a Node, utilizing a browser based tool for the retrieval and display of a Service Provider/Partner's information.

Data Type Definition (DTD)
Defines the legal building blocks of an XML document. It defines the document structure with a list of legal elements, i.e., where each tag is allowed, and which tags can appear within other tags.

Data Type
Part of a WSDL file describing information for all message requests and message responses.

Date Exchange Template (DET)
Identifies types of information required or allowable for a particular type of data set according to predefined standards. DETs are empty and contain no data. They simply define the format data must take prior to exchange.

DET Authority
This is the Technical Resources Group (TRG). It has responsibility for defining the DET and administering its deployment for other applications to use.

DET Registry
This is a logically centralized directory of Data Exchange Templates (DETs). DETs are the XML Schemas that describe the various payloads (data files) that may be exchanged across the Network. The DET registry provides a central place where the DET Authority, the TRG, can publish new DETs for subsequent discovery.

Digital Signature
Information requires non-repudiation and integrity protection in addition to privacy and authentication. Digital signature may be required by some data flows. When required, it is strongly recommended that XML-Signature be used for digitally signing the documents, and the signature be inserted into the SOAP header part of the message under such situations.

Document Notification
The Notify method is different from Submit in that there are no document contents, or attachments in the request message. The message simply informs a Network Node that some documents are ready to be retrieved; the service provider can, at its own convenience, download them at anytime. This operation is also known as Solicit.

Document Validation
These are data flow specific and Node specific. DETs identify types of information (data element and data groups) required or allowable for a particular type of data. DETs can take two forms DTD or XML Schema. Verifying a document based on DTD is a very tedious, error-prone process (involving many string comparisons). This is because a DTD only defines the structure of the expected document, with no definitions about the internal data elements (such as data types, data ranges, max and mins, occurrences etc). Furthermore, because DTDs are not in XML format, it is difficult to use XML tools to automate the data validation process.

Download
This method is a complement of Submit in that it facilitates bi-directional data flows between Nodes. In other words, a Network Node can be a sender at one time, but a receiver at another. With Download and Submit, the Node Network becomes symmetrical from a dataflow point of view. The following data flow diagram shows a symmetrical Network with three participating Nodes. The Download data flows inbound from the requester point of view; the Submit data flows outbound. The Download method is a function in the Retrieve Interface.

Dynamic Binding (DII - Dynamic Invocation Interface)
This enables clients to invoke operations on any registered service without having to first compile the client stubs that were created for that service by the compiler. By using DII,

a client can dynamically build operation requests for any interface that has been stored in the Registry.

Embedded Payloads
All payloads that are RPC-style messages must be base64 encoded. XML payloads of document/literal style messages can be inserted directly into the message body with a default namespace.

Encoding
This is a style which governs how a SOAP message is serialized and deserialized. The SOAP 1.1 specification defines only one encoding style, i.e., SOAP encoding, also known as the Section 5 encoding. An empty encoding style or missing encoding style indicates that no claims are made for the encoding style of contained elements.

Error
A condition that would be returned as a SOAP fault. There are many types of errors that could occur.

Event Notification
The Notify method can also be used for sending event notifications. If the stream element in a request message contains the relevant URI, then an event has occurred in the Network and the Document structure contains the type and description of the event. The event URI can be extended to carry additional information if needed.

Flow Configuration Document
Defines the business rules and parameters that will be in effect between a given service requester and service provider. The FCD registry provides a central place where Network participants can publish new FCDs. FCDs have traditionally been paper documents signed by the parties to the agreement. However, they can also exist in executable form supplying needed information to help automate business transactions that occur within the scope of the agreement.

Flow Configuration Document (FCD) Registry
This is a logically centralized directory of Flow Configuration Documents (FCDs).

FRS – Facility Registry System
This provides Internet access to a single source of comprehensive information on facilities subject to environmental regulations or of particular environmental interest. Currently CDX and FRS are working with 6 state partners engaged in the exchange of FRS data.

GetServices
This method is an administrative function for examining the capability of a Node. It allows requesters to query services provided by a Network Node. The type of services that can be queried includes, but is not limited to Interfaces. The web service interfaces supported by the Node. SQLQuery Database queries supported by the Node. SQLExecute SQL DML (Data Manipulation Language) procedures provided by the Node. A Node may choose to support additional types (meta-data) when needed. To get a complete list of all service types, a requester can pass ServiceType as the value of the ServiceType element. With GetServices, a requester can determine the capability of a Node at runtime and proceed accordingly. On the other hand, it allows the service

provider to extend the services provided, e.g., adding a new database report, without changing the infrastructure. The smart invocation and easy extensibility can greatly enhance the overall usability, stability and capability of the information exchange Network.

GetStatus
This is a method for transaction tracking. Once submitted, a transaction enters into different processing stages. GetStatus offers the client a way of querying the current state of the transaction.

HTTP – Hyper Text Transfer Protocol
An underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.

HTTP Security
This offers some basic authentication services on the transport level. The HTTP Specification (RFC 2616 and RFC 2617) defines an authentication mechanism known as "Basic" authentication. A client is challenged to provide identification information if authentication is required. The client then sends user name and password in the Authentication header. At this time, the user credentials can be passed to the web service for verification. HTTP authentication schemes are considered weak in term of confidentiality. Information exchanges between client and server are clear text, which are subject to attacks. Therefore, HTTP authentication is not recommended for securing Node operations.

HTTPS - Hyper Text Transfer Protocol Secure
This is a secure version of HTTP. Simply it is SSL underneath HTTP.

Interface
Part of a WSDL file of information describing all available functions/methods. It is also recommended that an activity log, a log that contains detailed processing steps, be provided to assist problem finding and debugging.

Literal Encoding
A style which allows arbitrary XML elements to be sent in a SOAP message. It has been a common practice to set the encodingStyle attribute to empty in such a situation.

Logging
All Network Nodes must log received transactions in a persistent storage and provide search capability for tracking transactions either by transaction id or submitter's id. In addition to information about submitted documents, the log record should contain requester's id, time received, transaction status and so on.

Message Confidentiality
Assures that most situations where messages are delivered through HTTPS transport. There are several situations, however, message may be compromised during transaction if not encrypted 1) Use of transports such as SMTP or FTP; and 2) Use of WS-Routing when messages travel over intermediaries. It is strongly recommended that messages be encrypted using XML-Encryption under such application scenarios.

Message Integrity and Non-repudiation
Assures that contents of a document were not tampered with and are protected during transition by using digital signatures. Contrary to the popular belief that digital signature offers more protection than encryption, signature and encryption are actually integral parts of one thing information security. Encryption only hides contents of a document; the contents can still be altered during transition. On the other hand, a digitally signed document without encryption is similar to sending an open letter without sealing it. Another very important aspect of digital signature is Non-repudiation. Some documents may require a digital signature to be considered valid by some data flows from a legal point of view. Digital signature is no longer an optional feature in such situations.

Messages
A logical grouping of parts, each of which is either an element or a complex data type.

Namespaces and Encoding Rules
Messages defined in this specification use either SOAP/RPC encoding (also known as Section 5/Section 7 encoding) or Document/Literal encoding. The SOAP encoding is governed by rules in SOAP section 5 specifications, while messages in Document/Literal encoding must conform to the specified schema.

NEI – National Emissions Inventory
A Clean Air Act (CAA) collection of point, area, and mobile and biogenic emissions data periodically submitted to EPA's Office of Air and Radiation by State and local air programs.

NEIEN
An acronym for National Environmental Information Exchange Network.

Network Exchange Business Processes
Web service, usage scenarios and typical interactions that occur during the course of the lifetime of a web service. For each of these typical situations, the scenario outlines who requests what of whom, and what kind of responses can be expected.

Network Service Interfaces
These protocols are classified into four major interfaces Send Interface A group of methods for submitting documents and other basic Network services. Database Interface A set of methods for database operations. Retrieve Interface A set of methods for event notification and polling, and document retrieval. Administration Interface Methods for Network-wide coordination and management

Network WSDL
The first version of a WSDL file to be used by all Exchange Network partners in building a first generation Node or a Node 1.0 product.

Node Validation
Refers to a process assuring full compliance with the Specification. It is conducted through a series of test messages. Test messages are request messages for verifying Node operations and validating responses. Data exchanged through test messages can be discarded. The service provider, however, must perform operations as requested. Tests are conducted using the same set of methods defined in this Specification, with special identifiers indicating that the messages are only tests.

Node Web Methods
The Network Node Functional Specification describes the behavior and interfaces of the service provider component. One of the design goals of this document is to create a framework of web services such that data exchanges of any type between Nodes can be conducted seamlessly and automatically. The web interface layer of the framework will create fully programmable environments on which clients can build automated tools, in any programming language, to send documents into the Network or to track previous submissions.

NodePing
Refers to a method that is a function in the Administration interface. It is a utility method for determining whether a Node is accessible. A positive response from the Node indicates that it is live and well. A network error (no response) or SOAP Fault (not ready) means that the service is not available at this time.

Notification
Commonly sent from the service provider to the service requester. This will typically be as a follow-on message to an initial service request that had a receipt acknowledgement sent from the service provider to the service requester. These notifications will typically contain large payloads that are being returned by the service provider to the service requester in response to the initial service request. It is possible that some very large payloads may be broken up into pieces for transmission. In this case, multiple notifications may be necessary.

Notify
This method has three intended uses document notification, event notification, and status notification Document notification A Node or client notifies a service provider about availability of some documents (soliciting). The service provider can retrieve the documents at anytime later. Event notification A Node sends, or possibly broadcasts, an event that is of interest to other parties. Event messages can be security alerts, shutdown notices, and other network management notes. Status notification A service provider sends a message to a requester to provide the current status of a submission or service request. In document notification, locations of the documents are provided in the cdxDocument structure. The service provider will use the same structure to download the available documents, so it is very important for requesters to include sufficient information so that the documents can be easily located. This specification does not define the semantics of events, as they are operation specific. Service providers are free to state the specific meaning of network events.

One-Way
A configuration, a message that is sent from the service requester to the service provider, and no response or fault is allowed back from the service provider.

Operation
A logical grouping of messages that can be defined as either “input” to the web service, “output” from the web service, or a “fault” or error returned by the web service. This is the basic information needed to generate the operational primitives that are the foundation of the Network service interactions.

Operational Primitives
There are four fundamental ways that messages can be configured. These four ways of configuring messages are called operational primitives. These primitives should be considered building blocks for the basic service interactions that will be discussed in the next section.

Operations
A logical grouping of messages that can be defined as either "input" to the web service, "output" from the web service, or a "fault" or error returned by the web service. This is the basic information needed to generate the operational primitives that are the foundation of the Network service interactions.

Partner
One of the two entities in a Trading Partner exchange. The Partner can either be the information provider or the information requester (client/consumer).

Payload Compression
XML payload can be compressed using a number of different techniques. Most compression techniques, when applied to XML, typically achieve very high compression ratios. However, XML compression changes the structure of an XML document, which complicates the process of digital signature (An XML document requiring signature must have a canonical form in order to be processed correctly by both the signer and the verifier). Therefore, compression will not be permitted at this time.

Payload Validation
The Exchange Protocol does not govern payload issues. However, it is expected that all XML payloads will be validated using the XML schema-based Data Exchange Templates (DETs) located in the Network registry that are used to XML encode the payload.

Payloads as Attachments
Network Nodes must support Direct Internet Message Encapsulation (DIME). DIME is a binary protocol with better performance compared to the SOAP with Attachment protocol. It is expected that more SOAP stacks will provide support for both protocols, in such a way that the attachments are decoded at the transport level.

Peer Node
Refers to a Node that has trusted security relationships with a group of other Nodes. State Nodes and the EPA Node are considered peers.

PKI – Public Key Infrastructure
The combination of software, encryption technologies, and services that verify and authenticate the validity of each party. PKIs integrate digital certificates, public-key cryptography, and certificate authorities into a Network security architecture. While PKI in theory provides an effective, robust means of securing electronic communications and transactions, deploying and managing the technology remains a daunting challenge to many organizations, especially in a large-scale deployment.

Port Types
This ties the input and output messages together as a request-response pair corresponding to a method invocation. An operation inside Port Type without output message indicates that it is a one-way operation.

Public Access
Public information that requires no authentication or certification of integrity.

Query Fields
Limitations or restrictions on the values that can be passed as parameters to the methods of the web service (e.g. date restrictions to control the amount of data that would be returned by the query). These would be applied before the web service processing was performed to save the web service from consuming more than a reasonable amount of system resources in attempting to carry out the request, or to keep the web service from consuming any resources if the query value (or various combinations of values, in the case of more complex business rules) was outside of acceptable, agreed-upon limits for the field for whatever reason.

Query
This method allows a Partner to send an information request to another Partner's Node and receives the data in XML formatted per its respective Schema. Given the inherent flexibility of this method, partners will need to establish "approved queries" and document them in the FCD for that flow. Many of these queries will be standardized across the Network (e.g. GetFacilityByID) while others will be unique to sets of trading partners. Nodes typically implement Query (and Solicit) by establishing an internal stored procedure which produces the required information and is invoked with its corresponding Query request is received.

Receipt Acknowledgement
Some requests will be defined to have an immediate response that provides the service requester with an acknowledgement that the request was received by the service provider. The receipt acknowledgement does not contain the data being returned as a result of the request. Instead, this information will be returned in a separate subsequent "notification" message sent from the service provider to the service requester. This type of interaction is necessary in situations where the generation of the regular return value may take considerable time, and the service requester needs to know at least whether the request was successfully received and is being worked on by the service provider. Use of receipt acknowledgement is determined by the specific dataflow and the governing procedures and policies of the flow.

Registry
The title of the repository of schema being utilized for Network exchanges. The registry is the official record of Schema, during different phases of development, for the use of all Network Exchange Partners.

Relational Document
Structured documents with relational constraints imposed on internal data elements. Records from a relational database are considered relational.

Request/Response
A message is sent from the service requester to the service provider, and either a response or a fault is received back from the service provider.

Requester
A Node that initiates SOAP request messages.

Resource consumption
Limitations on other types of resources consumed by the service during the processing of the request (e.g., temporary disk space or intermediate files or database connections.) These would be applied during the processing, essentially setting and starting an alert capability that would be monitored and, once exceeded, would cause an alarm event to be invoked with an appropriate event handler to cause a fault to be returned to the requester. Any partial results that may have been generated as a result of the processing of the request may or may not be specified to be returned with the fault.

Response Types
There are 5 different responses that can be received from a service provider in response to a request. 1) Simple Response; 2) Receipt Acknowledgement; 3) Notification; 4) Solicit Response; and 5) Error.

Return
The response message contains a data flow identifier and a set of documents. Documents transmitted can be either embedded payloads or separate attachments.

Security
This layer insulates the application from unwanted intrusion and unauthorized access. It can employ a number of different security protocols. However, the approach that must be supported by the Network at this time is Secure Sockets Layer (SSL) plus service level user authentication and authorization (user name and password).

Service Consumer
A partner or client using the Network to obtain information from a service provider.

Service Description Repository
This is a logically centralized storage location for the Service Descriptions, also called Web Service Description Language (WSDL) files. The service description repository provides a central place where the parties to a trading partner agreement can store new service descriptions for subsequent retrieval.

Service Description
This layer is responsible for describing the interface to a specific web service. The approach that must be supported by the Network at this time is WSDL 1.1.

Service Discovery
This layer is responsible for centralizing services into a common registry and providing publishing/finding functionality. The current approach for providing this functionality is UDDI (Universal Description, Discovery, and Integration)

Service Provider
This refers to the provider of the web service. The service provider implements the service, publishes its availability, makes it available on the Internet, and processes requests for services.

Service Requester
This refers to any consumer of the web service. The service requester discovers an existing web service, retrieves its description, and then utilizes the web service by

opening a network connection and sending an Extensible Markup Language (XML) request conforming to its interface description.

Services
Element describing a physical web service, i.e., which binding to use and where the service is hosted (known as the endpoint). Note that service is an optional element in the WSDL specification. A WSDL file without a service element defines an abstract interface, which could be implemented by many service providers. In fact, all WSDL files listed in the UDDI registry must not contain any service definition.

Simple Document Submission
An operation where a client sends an array of documents for a specific data flow to a Network Node.

Simple Response
This has the return value encoded in the body of the response SOAP message. The return value will be a single structure. The convention is to name the message response structure with the name of the request with the string, “Response”, appended to it.

SMTP
An additional transport layer protocol that is being considered for moving SOAP messages is Simple Mail Transport Protocol (SMTP). SMTP is used to move messages, and frequently large quantities of information, from a source to a destination. This is accomplished asynchronously in a fire-and-forget fashion. It is very efficient at moving information one way.

SOAP - Simple Object Access Protocol
An XML-based protocol for exchanging information between computers.

SOAP Attachments
In a document exchange process, payloads could be any type of file, including XML files, text files and binary files. There are two standards available for attachments SOAP Message with Attachment (SwA) and Direct Internet Message Encapsulation (DIME). Network Nodes must support DIME. All attachments must be referenced in the SOAP main message body. Unreferenced attachments, which have no meaning to the receiver, will not be processed.

SOAP Body
The body element is used to provide information about the message.

SOAP Envelope
An element which is the root element of the SOAP message. The rest of the SOAP message must be contained within the envelope start and end tags. The envelope element must be prefixed with an indicator of the namespace that defines the SOAP version that is applicable. The version is indicated by the namespace attribute, xmlns, included in the envelope element start tag. The namespace prefix could be any valid XML namespace string, but the convention usually adopted is as follows <SOAP-ENVEnvelopexmlnsSOAP ENV=”http://schemas.xmlsoap.org/soap/envelope/”>. The namespace name SOAP-ENV is really a symbol for http://schema.xmlsoap/soap/envelope. Although it can be any NCName, the URL part

must be exactly as specified. A different URL represents a different version of SOAP and must cause the VersionMismatch fault.

SOAP Fault Codes
A protocol which defines four fault codes that must be used in all SOAP fault messages.

SOAP Fault Detail Codes
Conforms to the SOAP 1.1 specification and uses the predefined SOAP Fault Codes. In addition, all SOAP fault messages must contain a Fault Detail element, with Network exchange specific error codes and error descriptions, when processing of a SOAP request body fails.

SOAP Header
An element used to provide information about the message.

SOAP Message with DIME
A binary protocol proposed by Microsoft and IBM. The advantages of DIME are simplicity and performance. DIME attachments do not need to be encoded, which is often a time and resource consuming process. Each payload, including the main message body, is encapsulated in a DIME record. A DIME message is a set of records with the main SOAP message as the first record.

Solicit Response
Instead of a notification, a solicit response may be returned after an acknowledgement. This might be used in cases where the service requester may have requested a scheduled or otherwise conditional response rather than an immediate response. The service provider may send a solicitation back to the service requester asking whether the conditions are acceptable to the service requester, and expect to get a response from the service requester in reply. If the condition was acceptable to the service requester, then the service provider would subsequently send a notification containing the return data.

SSL - Secure Sockets Layer
A protocol developed by Netscape for transmitting private documents via the Internet. SSL works by using a public key to encrypt data that's transferred over the SSL connection. By convention, URLs that require an SSL connection start with https instead of http.

SSL with Client Authentication
Information that requires some additional level of authentication and a higher level of integrity protection. It is protected through SSL plus application level client authentication (username and PIN). This level of security must be implemented by all Nodes participating in the Network information exchange.

SSL with Dual-authentication
Information requires bi-directional authentication and a higher level of confidentiality. It is often protected using SSL with dual authentication. SSL with dual-authentication will be required depending on the data flow, but is not mandatory for all Network transactions.

Stack
The basic protocol of a web service can be visualized as a stack of several layers of capability with various standards applicable to each layer. Each layer is independent from the layers above and below it. Each has its own job that provides greater flexibility allowing the connection of all forms of disparate systems and Network technologies to support distributed processing over the Internet.

Status notification
This is a Solicit Response operation. A service provider may notify a requester of process status, i.e., file submission status, using the same notification message. Status notification is a complement of the GetStatus operation in that submission (or operation) status information can flow both ways. In some situations when documents have to go through a lengthy process, an impatient submitter may call GetStatus many times with no expected result. With status notification, however, the submitter is notified when the status of the submission changes. Active status notification can, in many situations, reduce network traffic and improve the quality of services.

Structured document
Documents conform to a predefined structure. Documents, in document/literal encoding, carried in the SOAP message header or body are structured documents. External XML documents attached to SOAP messages are also structured.

Submit
A method that allows a client to send documents to the Network service. A document in the request message is formally defined, using XML schema. Document contents are either embedded in the message body as base64-encoded string (of type base64Binary), or a reference to an attachment associated with the request message.

Target Node
The ultimate destination of a data flow, a target Node may or may not implement the Network Exchange Protocol. The first case is more flexible because the requester can construct a SQL statement when needed; the second, however, is more powerful because the provider could construct very complicated business logic, rules, and put them into a procedure.

tModel
Sometimes also referred to as a Technical Model, is used in UDDI to represent unique concepts or constructs. They provide a structure that allows re-use and, thus, standardization within a software framework. Interfaces defined by the Network Exchange Protocol will be registered as tModels in a private UDDI registry.

TPA - Trading Partner Agreements
Written agreements that define the partners, information, stewardship, security, and other items essential for the exchange of information between two or more trading partners on the Network. In short, TPAs establish formal processes for managing the flow of information across the Network. TPAs may apply to exchanges initiated by the sender or those initiated at the request of the receiver. If exchanges are intended to meet mandatory reporting requirement, TPAs are necessary when automated exchanges are to take place without operator intervention.

Trading Partners
Two parties involved in an exchange of information over the Network and will, at some point in time, establish a TPA to formalize their exchange process.

Transport
This layer is responsible for transporting messages between applications. It can also employ a number of different protocols. However, the transport protocol that must be supported by the Network at this time is HTTP/HTTPS 1.1 (Hypertext Transfer Protocol).

UDDI
An acronym for Universal Description, Discovery, and Integration. A Web-based distributed directory that enables Partners to list themselves on the Internet and discover each other, similar to a traditional phone book's yellow and white pages.

UDDI Data Model
A UDDI registry has four major entity types 1) businessEntity Describes a business or an organization that provides web services; 2) businessService Describes a set of services provided by a businessEntity; 3) bindingTemplate Defines how services can be accessed. BindingTemplate provides the technical information needed by applications to bind and interact with the Web service; and 4) tModel Describe a technical model. It often contains an abstract definition of a web service (Web Service Type).

UML- Unified Modeling Language
An industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

Unstructured Document
Documents that do not have a predefined structure fall into this category. Examples include word documents, flat files and binary files.

URI - Uniform Resource Identifier
The generic term for all types of names and addresses that refers to objects on the World Wide Web. A URL is one kind of URI.

Web Services Standards
At each layer of the web services protocol stack there are one or more applicable standards that must be understood and addressed.

WSDL - Web Services Description Language
An XML-based language specification defining how to describe a web service in computer readable form. For a given web service, its WSDL file describes four key pieces of data 1) Interface – information describing all available functions/methods; 2)Data type – information for all message requests and message responses; 3)Binding – information about the transport protocol to be used; and 4) Address – information for locating the specified service. WSDL represents the contract between the service requester and the service provider. Using WSDL, a client can locate a web service and invoke any of its available functions. With WSDL aware tools, you can automate this process. There were originally several other proprietary attempts to create a similar specification (IBM's NASSL and Microsoft's SCL). But WSDL is rapidly becoming the de facto standard for carrying out this functionality.

XML - Extensible Markup Language
A mark-up language designed especially for Web documents. It allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between partners.

XML Messaging
This layer is responsible for encoding messages in a common XML format so that the messages can be understood at either end. The approaches that must be supported by the Network at this time are a) SOAP 1.1 for the encoding of the message structure and b) XML Schema for the encoding of the message payload.

XML Namespace
A collection of names, identified by a URI reference. Namespaces in XML documents provide processing context and prevent name collisions.

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

Exchange Network related Groups

CDX – Central Data Exchange
EPA's Central Data Exchange (CDX) is the point of entry on the National Environmental Information Exchange Network (Exchange Network) for environmental data submissions to the Agency. Through CDX and the Exchange Network, EPA is working with reporting entities, including industry, States and local and Tribal agencies to enable streamlined, electronic submission of data via the Internet. The Central Data Exchange Team is responsible for: developing the technical capabilities to receive and process electronic reports, developing an appropriate policy and legal framework to ensure that electronic submissions are legally acceptable, and working with States, Tribes and reporting entities on facilitating data submissions.

DMWG – Data Management Workgroup
The Data Management Workgroup is the state component of the State/EPA IMWG formed in 1998 to address issues related to information management in the States and to find innovative solutions for common problems. Currently there are 28 official State members, with over 35 States active on various projects.

EDSC – Environmental Data Standards Council
The Environmental Data Standards Council (EDSC) develops environmental data standards to promote the exchange of information among States, Native American Tribes, and EPA. The Council identifies those areas of information for which having standards will render the most value in achieving environmental results, prioritizes the areas, and pursues the development of data standards.

IMWG - Information Management Workgroup
State/EPA Information Management Workgroup (IMWG) is composed of senior leaders from EPA and state environmental agencies. The IMWG was formed in 1998 to work on information management issues of joint concern to States and EPA.

NAAS – Network Authentication and Authorization Services
Network Authentication and Authorization Services (NAAS) provides centralized security services. Security tokens and assertions issued by NAAS are trusted and accepted by all Network Nodes. All operations defined in NAAS must be conducted over a secure SSL channel using 128 bit encryption.

NSB – Network Steering Board
The Network Steering Board is the administrative body which governs the implementation, operation, and ongoing maintenance of the Network. The NSB's primary functions are to oversee and steer implementation of the Network, maintain and operate a Network Registry/Repository and develop guidance and best practice recommendations.

TRG - Technical Resources Group
The TRG's primary responsibility is to provide specific technical advice and assistance to the NSB on issues relating to the implementation and ongoing maintenance of the Network. The TRG is comprised of four subgroups, the Core Reference Model subgroup (CRM), the Data Exchange Template (DET) subgroup, the Network Registry/Repository subgroup, and the Schema Review subgroup.

TRG (CRM) - Technical Resources Group (Core Reference Model)
The CRM subgroup's major responsibility is to create the Network Core Reference Model. The CRM is a high-level description or roadmap of Network data flows that shows the relationships among major data groups, data standards, and data flows. The CRM is a tool used to identify opportunities for data standardization that will improve data flows and data flows that might benefit from data standards & harmonization.

TRG (DET) - Technical Resources Group (Data Exchange Template)
The primary responsibility of the DET subgroup is to provide technical guidance on creation, use, and harmonization of Network DETs. The DET subgroup will or has created several documents which address; Network XML Schema Design Rules, Conventions and Guidance; Handling Code Lists and Enumerations in XML Schema; Managing Nillable Values in XML Schema; and Network Namespace Management.

TRG (Registry) - Technical Resources Group Registry
The Network Registry/Repository subgroup is responsible for scoping, selecting, and building the Network Registry/Repository.

TRG (Schema Review) - Technical Resources Group Schema Review
The Schema Review Workgroup is a new effort intended to further ensure the harmonization and compliance of existing and future Schema. The workgroup will establish a pilot to review 3 existing Schema and to establish the process and procedures for reviewing all other Schema.