

ACKNOWLEDGEMENTS

This document was developed with invaluable input and support from the following individuals:

Andrew Battin	US EPA
Chris Clark	US EPA OIC
Connie Dwyer	US EPA OIC
Dennis Burling	Nebraska DEQ
Dennis Murphy	Delaware
Glen Carr	Oregon DEQ
Imegwu Akachi	US EPA
Jeffrey Wells	US EPA
Jennifer Gumert	Pennsylvania DEP
Ken Blumberg	EPA Region 1
Maryane Tremaine	US EPA
Michael Beaulac	Michigan DEQ
Mike MacDougall	US EPA
Miles Neale	Washington DOE
Mitch West	Oregon DEQ
Molly O'Neill	ECOS
Pat Garvey	US EPA
Terry Forrest	US EPA

REVISION HISTORY

Revision Date	Ver,	Remarks
8/22/2005	1.0	Original Release
12/30/2005	1.1	Updated Appendix A Roadmap, Added detail to "Testing the Data Exchange", updated "Best Practices for Schema Design"
8/3/2006	1,2	Revised guidance on Data Service naming conventions

Prepared By



4000 Kruse Way Place
Building 2, Suite 285
Lake Oswego, OR 97035
(503) 675-7833

<http://www.windsorsolutions.com/>

Table of Contents

INTRODUCTION	4
<i>Background</i>	4
<i>How to Use this Document</i>	4
DATA EXCHANGE DESIGN GUIDANCE	5
<i>Overview of the Exchange Design Process</i>	5
<i>Organization and Planning</i>	6
Form a Project Team.....	6
Determine and Agree upon Exchange Goals	6
Determine Project Activities and Document Deliverables	9
<i>Designing a Data Exchange</i>	12
Determine Data Exchange Architecture	12
Choose Appropriate Transaction Granularity	13
Determine Appropriate Data Services	13
Plan the Data Exchange Frequency	14
Avoid Flat File Conversion to/from XML.....	14
Implement Error Handling and Failure Recovery	14
TESTING THE DATA EXCHANGE.....	16
Testing and the Data Exchange Design Process.....	16
Common Problems Discovered During Testing.....	16
<i>Publishing the Data Exchange</i>	18
Ensuring Schema Design Conformance	18
Proper Versioning of Exchange Deliverables.....	18
DATA EXCHANGE DESIGN BEST PRACTICES	19
<i>Best Practices for Schema Design</i>	19
<i>Best Practices for Handling Large Transactions</i>	21
<i>Best Practices for Managing State</i>	21
<i>Best Practices for Data Services</i>	23
<i>Best Practices for Data Validation</i>	25
APPENDIX A – DATA EXCHANGE DEVELOPMENT ROADMAP	26
APPENDIX B – DATA EXCHANGE MODELS.....	27
The Data Synchronization Model	27
The Data Publishing Model	28
Using a Combined Approach.....	29
APPENDIX C – DATA EXCHANGE DESIGN PATTERNS.....	30
Patterns in Authentication and Authorization.....	30
Patterns in Requesting Data	32
Patterns in Submitting Data	35
Other Data Exchange Scenarios.....	36

Introduction

Background

The National Environmental Information Exchange Network (Network) allows states, EPA and tribes to exchange data using an agreed set of protocols and standards. Each type of information exchanged over the Network is a **data exchange**. Data exchanges are often called “flows” or “data flows” in Exchange Network discussions. A data exchange is the sharing of a specific type of data between two or more partners. Exchanges are often centered on a particular environmental interest (such as surface water quality data), EPA program (such as the Toxic Release Inventory) or EPA database (such as RCRAInfo).

As of this writing, Exchange Network partners have created several fully operational data flows on the Network. The Network community can now reflect on the process of designing data flows and their commonalities or design patterns. While each exchange differs in content, the process of developing, testing and documenting data flows is similar for each. The conclusions of the Network Operations, Technology and Security Team (OTS) in this document are for the benefit of next generation exchange designers.

In addition to providing guidance on the data exchange design process, this document aims to address the common challenges faced by those designing data exchanges along with potential solutions. Over the short history of the Network, recurring problems have emerged across different flows. Data exchange implementers have tackled each problem differently and with varying degrees of success. This document offers the Exchange Network’s approach to these common issues.

How to Use this Document

This document is divided into two major sections; Exchange Design Guidance and Data Exchange Design Best Practices. The first section describes the process of designing a data exchange from inception to completion and is oriented toward both technical and business experts. This section provides information to help project members understand the data exchange design process and avoid common pitfalls. The second section, Data Exchange Design Best Practices, addresses technical topics which will be of most use to technical architects, analysts, programmers and exchange implementers. In addition, the appendices provide additional background information on common exchange models, scenarios and design patterns.

This document is not intended to be the definitive guide on exchange design for the Exchange Network. Such a comprehensive document does not exist at this time. Formalized data exchange design methodology is a work in progress for the Exchange Network and will be supported by its governance structures. OTS intends that data exchange designers use this guide to learn from the experiences, problems and solutions of the early data exchange implementations.

Data Exchange Design Guidance

Overview of the Exchange Design Process

Although each data exchange has different characteristics, the process for developing, testing and publishing the exchange is similar. The exchange development process follows these major steps:

1. Organization and Planning Phase

- a. Establish the project team
- b. Register the project with the Exchange Network governance

2. Requirements Gathering Phase

- a. Determine exchange objectives and requirements
- b. Determine the exchange components and associated services
- c. Determine XML schema requirements (review Network guidance)
- d. Determine security requirements (review Network guidance)
- e. Review applicable Network guidance documents
- f. Determine role of project team members for testing phase

3. Design and Build Phase

- a. Develop draft XML schema (conforming with Network standards)
- b. Develop draft data exchange template (DET)
- c. Develop sample XML instance files
- d. Develop draft flow configuration document (FCD)
- e. Develop data exchange services and supporting software
- f. Develop test scripts and tools to support testing of exchange

4. Testing Phase

- a. Test iteratively, documenting problems and corrections made for each iteration
- b. Refine XML schema and services based on test findings

5. Publishing Phase

- a. Finalize documentation to reflect any changes made during testing
- b. Prepare the schema submission package for the schema review process
- c. Develop Trading Partner Agreements (TPA) for exchange partners
- d. Prepare lessons learned document
- e. Submit XML schema and supporting documents to Exchange Network Web site/registry

A diagram depicting the roadmap for data exchange development can be found in Appendix B.

Organization and Planning

Developing a Data Exchange is similar to any other technology project. While standard project management methodologies apply, there are some specific to the data exchange design process. This section describes coordinating, guiding and controlling this process.

Form a Project Team

The first step in designing a data exchange is to assemble a team of stakeholders. In Exchange Network terms, this group is an Integrated Project Team (IPT). There is no formal process for creating an IPT. Most commonly, IPTs are formed either in developing a joint proposal for or in response to receiving an Exchange Network grant to develop a data exchange. Alternatively, an IPT may form on an ad hoc basis in response to a shared interest in exchanging environmental data.

The IPT should be composed of program area and IT staff from organizations representing interests in the exchange of a particular type of data. This ensures a variety of perspectives, interests and expertise. With the input of these stakeholders, it is likely that the resulting data exchange will serve the needs of a wider audience, therefore encouraging wider participation and adoption.

While each data exchange may differ in purpose and content, the IPT is normally composed of the following groups or individuals:

- Business area experts from two or more organizations, each with a programmatic interest in the data
- Technical experts familiar with the data systems of each of the pilot data exchange partners
- Technical experts familiar with XML Schema design and Exchange Network technologies and standards

It is important that IPT members not only have the interest and will to participate in the data exchange design process, but the time and resources to build, implement, and pilot the exchange of data and commit to its continued maintenance.

Determine and Agree upon Exchange Goals

Common Data Exchange Design Goals

To help determine the goals for a data exchange, it is useful to see the stated goals of other published data flows. The following table describes some of the goals of a few of the current data exchanges.

Data Exchange Name	Initial Exchange Goal	Goals for Future Phases
Facility Identification (v1.0)	“...to improve the data [quality] in EPA’s [Facility Registry System]” “...to improve states’ facility data” “...to provide facility data to state	“...[provide] a more complete and robust set of services” “Incorporating fully standardized code lists”

Data Exchange Name	Initial Exchange Goal	Goals for Future Phases
	or EPA partners for value-added integration into their own applications and services...”	“Identifying a sophisticated error tracking and data checking service...”
National Emissions Inventory (NEI v1.0)	<p>-To provide an automated mechanism for uploading data to EPA’s NEI database (as opposed to the current manual process)</p> <p>-To provide an XML format which matches the data content of the existing NIF submission file format</p>	-upgrade the XML schema to match the NIF 4.0 format when published in 2005
RCRA Exchange (v1.0)	-To provide an automated mechanism for states to upload data to EPA’s RCRAInfo database using XML and the Exchange Network	<p>-Allowing EPA to request RCRA data from states, as opposed to the states initiating the submission of RCRA data to EPA</p> <p>-Exchanging RCRA data between states</p>
Toxic Release Inventory (v1.0)	<p>-Provide near real-time TRI reports to states from EPA as soon as they are received from the TRI reporter.</p> <p>-Provide an XML format which fully describes data collected on TRI reporting forms</p>	-Allow states to request TRI data from EPA at will.
Pacific Northwest Water Quality Data Exchange (PNWWQDX v1.0)	<p>“...aggregate and access a comprehensive source of high-quality water data in the Pacific Northwest.”</p> <p>“[Provide] simple query tools to retrieve and consolidate information from each data provider.”</p>	

Some data exchanges have more ambitious goals than others. Some supplant existing manual, flat file data submissions with automated XML submissions without altering the transaction frequency or data content. However, data exchanges that improve the sharing and usability of data are most desirable. While considering a new data exchange, one must examine how the data exchange will improve the sharing and use of data.

To aide this effort, the Network Planning Action Team (NPAT) assembled a list of design questions¹ which all data exchange designers must consider when planning a new data exchange. These questions are:

1. For an existing data exchange, can the use of the Network provide benefit by leveraging common infrastructure without a diminution of service?
2. For an existing data exchange, can the Network:
 - Allow the exchange to happen more frequently, thereby decreasing the lag between partner systems?
 - Make the exchange more efficient, and reduce or eliminate manual intervention, such as scheduling, resubmissions, or security?
 - Provide higher quality data due to additional or more efficient error checking and/or earlier detection of errors/discrepancies?
 - Use data standards and common formats to provide additional definition, structure, and integration opportunities?
 - Improve the exchange by leveraging shared infrastructure services (e.g., a node and/or CDX)?
3. Can the new Network data exchange expand upon the existing (or proposed) exchange to include additional data by:
 - Encouraging partners who did not previously provide data to do so?
 - Providing data on a broader universe of entities (e.g., new classes of facilities, new areas, or new contaminants)?
 - Allowing for more detailed data about the current entities (either in a system-to-system flow or interactively) than the original flow?
4. Does (or will) the flow leverage the use of Web services, within the larger framework of the Network's shared strategy to provide new functionality to the implementer and their flow partners?
5. How are the flow partners or their customers planning or aspiring to use these new capabilities and/or data to make improved environmental decisions?

Consider New Possibilities

The possibilities offered by Web Services are enticing. The Exchange Network lets partners exchange data in near real time. This allows them to consider alternatives to the traditional infrequent exchange of large volumes of data.

While the Exchange Network can support traditional data exchanges, the Network's capabilities should encourage data exchange designers to provide more timely and focused access to data. Even if the initial data exchange pilot replicates traditional data processing, it may be possible to design XML schema and data exchange services which allow for more focused querying and more detailed transactions in the future.

¹ The design decision question list is included in section 4.2 of the NPAT Exchange Network Business Plan available at http://www.epa.gov/oeiinter/imwg/pdf/business_plan.pdf. The Business Plan contains valuable information about data exchange design and Exchange Network governance.

Furthermore, it is useful to consider how the data transport container format (XML schema) may be unbound from a specific source or target system. By doing this, the schemas can become useful in a broader array of applications. For example, a generically designed schema may be suitable for exchanging data between any two partners, not only the partners which initially designed the schema. Methods for designing schema with this goal in mind are covered in the *Schema Design* section of this document.

Often, exchange partners identify exciting new opportunities for data sharing and consumption during the project. Data exchange developers are encouraged to undertake innovative data exchange scenarios, for it is these activities which ultimately evolve the Network.

While these ideas are enticing, it may not be feasible to include them in the initial data exchange because of project resource and scope limits. In these cases, partners can capture these ideas in the data exchange documentation for consideration by the next generation data exchange designers. It is important that these ideas are not lost.

Determine Project Activities and Document Deliverables

Project Activities

Data exchange development involves more than simply constructing a series of documents. The project team needs plan, build, test and refine a data exchange between two or more partners. Ideally, three or more agencies should implement the data exchange. The advantages are many when multiple agencies are involved with building and testing an exchange:

- The exchange is testing using multiple software/hardware platforms leading to the discovery of platform-specific, or cross-platform implementation issues
- more software tools are produced which can be shared with future implementers
- variations in each partners' database and node architecture lead to the discovery of new issues
- by involving more agencies, a larger resource base is available for future implementers to query

For example, four states participated in the pilot Toxic Release Inventory data exchange. Each agency discovered new problems and as a result, these problems were corrected and the quality of the exchange improved with each iteration.

Document Deliverables

Exchange partners will design and produce several documents to support the data exchange. Each of these products is described below, along with the purpose and role of each:

Deliverable Name	What is It?	Why Do I Need It?
Data Exchange Template (DET)	A spreadsheet outlining each data element within the Schema along with definitions, validation rules, and example content. This is a more human readable	Partners wishing to participate in the data exchange will need to map their databases to the schema so data can be translated and moved between the two. This document provides technical staff

	version of the XML Schema. The DET also defines minimal data content and data quality considerations.	with all the information they need to do the task
Example XML Instance document	A sample XML file containing valid data using the schema developed for the exchange	An example XML document can greatly help developers and user understand what the intended 'output' of the Node should be when servicing a request. For example, for a Facility Identification exchange, one could get an XML file from an existing partner that includes data for a few of their Facilities.
Flow Configuration Document (FCD)	The principle document which captures the detailed data exchange processing rules governing the data exchange using narrative text, diagrams and examples.	This document is the centerpiece to publishing a data exchange, since this is the first thing a prospective partner will read when considering how to participate in the data exchange.
XML Schema	The formal definition of the structure and format of the data.	Without the schema, there would be no formalized, computer-readable package in which to deliver data from one partner to another.
XML Schema User's Guide	A document explaining the usage of the XML schema and any special usage conditions which are not implicit within the schema	(Optional) Most schema have special usage considerations which are not evident by simply looking at the schema. This document provides a place to describe some of the big picture schema design features as well as any intricacies that exist.
Data Services XML File	This file formally specifies the data services described in the FCD	(Optional). This may be used during testing of the exchange using a Node Client so the data requestor can provide a more intuitive user interface to your Node's data services.
Schema Conformance Report	This document describes the findings resulting from a comparison of the final draft schema against the guidelines for schema	This is one of the resources needed by the Network schema review team to determine if the data standards, guidelines and resources (CRM/SSC, DRC, etc.)

	design as proposed by the Exchange Network.	were adequately considered and utilized during schema development.
--	---	--

Designing a Data Exchange

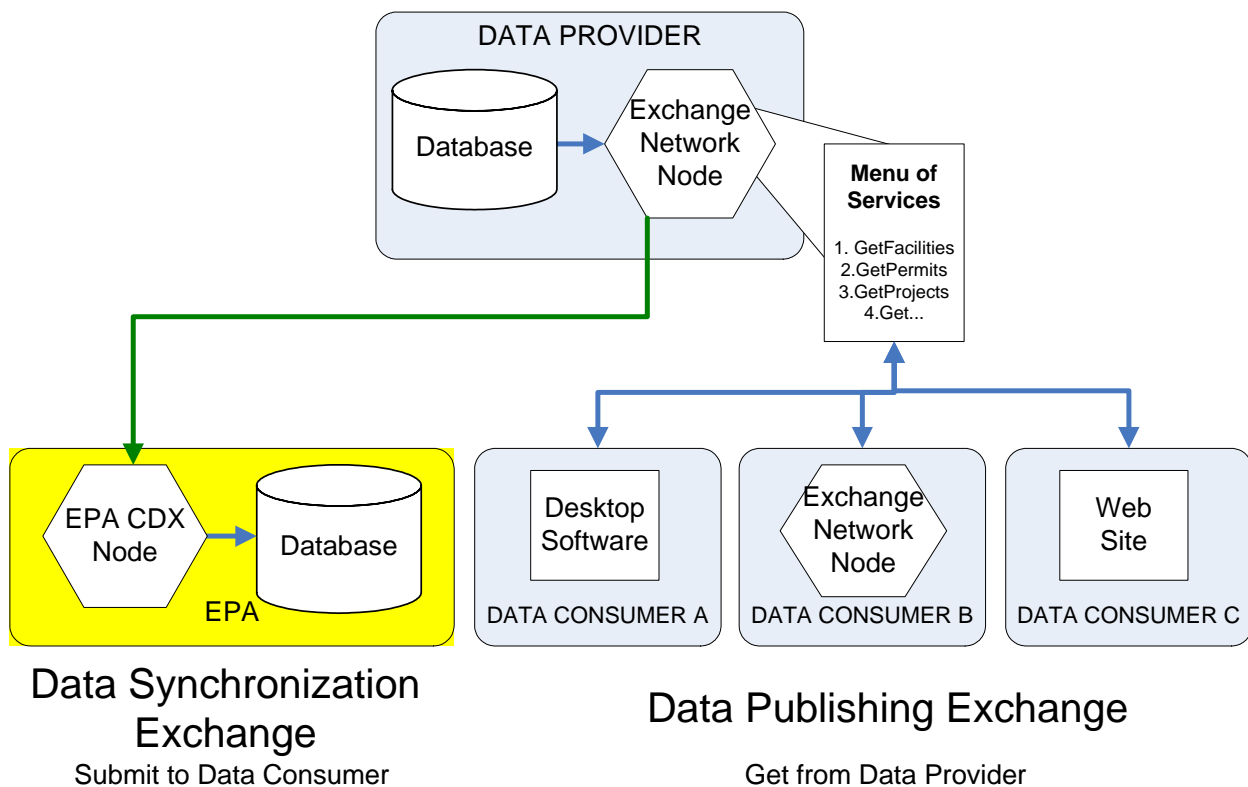
Determine Data Exchange Architecture

Two major exchange design types have emerged on the Network; **data synchronization exchanges** that update or synchronize with another partner's database, and **data publishing exchanges** that broaden the availability of data to one or more external audiences. Some data exchanges may have characteristics of both.

A data synchronization exchange is akin to traditional regulatory exchanges where one or more data providers regularly submit a bulk set of data to a data consumer such as the EPA. Examples include annual National Emissions Inventory (NEI) reporting, annual beach monitoring and notification data reporting, and submission of NPDES data to the EPA Permit Compliance System (PCS). While such reporting is still required by many programs within EPA, this type of exchange is discouraged in favor of the data publishing approach.

The data publishing design model makes services and data available to data consumers via the Internet. These data services are typically always available, thus allowing a requestor to "come and get" the data on demand. The data publishing model can allow a partner's data system to be a logical extension of another partner's system. Often these exchanges are more transactional with more frequent exchanges of smaller amounts of data than data synchronization exchanges.

The following diagram illustrates how a data provider can implement functionality from both models.



Appendix B – Data Exchange Models and *Appendix C – Data Exchange Design Patterns* contain more detailed information about the synchronization and publishing exchange models.

NOTE: Each data exchange must be given a name (i.e. FacilityID, AQS, NEI, RCRA). The flow name is either an acronym or shortened description of the exchange. The flow name can not exceed ten characters in length. The data exchange name is critical for several reasons; it is a required parameter for all Submit transactions, it is a required prefix for all Data Service names (see *Use appropriate data service names* section on page 23), it is required to register a flow on the Exchange Network Discovery Service (ENDS) and it provides a simple, common term to describe the exchange.

Choose Appropriate Transaction Granularity

Partners must decide on the detail level of their exchanges. When they select the finest level of granularity, they transmit field-level data. At the coarsest level of granularity, for every transaction, they transmit the entire source system dataset (a full replace).

Both extremes of granularity can be difficult to manage. Of the two, fine-granularity transactions are the more difficult. They typically involve tracking, transmitting, decoding and consuming a large amount of data for each transaction.

The best solution is to decide on an appropriate level of granularity somewhere between the two extremes. For example, a coarse granularity exchange for Facility Identification would include everything in the submitters' system if any element in any facility changed. At the other extreme, a fine granularity exchange would include only the data elements that changed for any facility. Facility Identification exchange partners use an intermediate level of granularity. If any element, such as facility name or zip code changes for any facility, the next data submission would include that facility record.

GRANULARITY IN FACILITY IDENTIFICATION EXCHANGE		
Data Granularity	Data Changes	Data included in Payload
Coarse	Any changed data element (e.g., name, zip code) in facility database	All data in facility database – “full replace”
Fine	Any changed data element (e.g., name, zip code) in facility database	Only changed elements
Intermediate (Used in Facility Identification Exchange)	Any changed data element (e.g., name, zip code) in facility database	All data elements for any facility record with a change in at least one element

For data synchronization exchanges, it is typical for the XML schema to include a transaction code at the level in the data hierarchy which delineates a transaction unit. A transaction codes specify whether a record and its related data is to be added, deleted or changed in the receiving system.

Determine Appropriate Data Services

When designing a data publishing exchange, it is necessary to define the data services which will be supported. Depending on the requirements of the exchange, it may be ideal to include both

fixed and dynamic-parameter data services. Data providers should make a small set of use-case driven queries available which utilize the fixed-parameter model. These services are analogous to standardized reports in a traditional database application. In addition, providing a small set of dynamic-parameter queries will enable data requestors to construct queries flexibly. These queries are akin to ad hoc query tools which are available in many traditional database applications.

Plan the Data Exchange Frequency

Data exchange designers should consider how the near real-time availability of data from network partners can reduce the need for large queries. The Network makes it possible for a record received by one partner, to be forwarded immediately to another partner using web services. For example as soon as a state receives a DMR from an NPDES facility, the state node could forward that record to EPA's PCS system.

This type of small but frequent data exchange is ideal for Web services and promotes timely transfer of data between partners, but it can be difficult to implement since it may require tighter coupling of the web services with the source database. For example, partners may need to devise a method to spawn the data exchange process via Web services upon a change in database data.

Avoid Flat File Conversion to/from XML

Avoid, when possible, designing a data exchange which involves chaining an XML converter into an existing flat-file data exchange. If at all possible, XML should be composed directly from either a source relational database or a staging relational database constructed for the exchange on the sender side and parsed directly into a relational database on the receiver side.

When XML-to-flat file converters (or vice versa) are integrated, one must examine if it is possible to replace (or at least completely decouple) the flat-file processing with the XML processing. Each converter which transposes one format to another is another potential breaking point in the processing chain. In addition, creating dependencies on legacy file formats works against the directive that XML be the sole standard transport format for the Exchange Network.

Implement Error Handling and Failure Recovery

Why error handling is important

Error handling and recovery is consistently underdeveloped in current production data flows. Very few of the published data exchanges have a robust system of handling and resolving fault conditions when they occur. Given the regulatory requirements in place for many of the data flows in production, it may seem surprising that more robust error handling and fault recovery processes are not in place.

Why are fault-recovery procedures not well developed? It may be that once data exchange testing begins, developers often assume the design process is complete. It is important to start the error recovery discussion and to involve the business experts early in the process.

Because many flows are completely automated, Exchange Network data which is lost while being processed is especially dangerous. This is because it is possible that no person will notice a fault occurred or that data is missing. For this reason, it is especially important to provide for fault conditions when constructing data exchanges for the Network.

Common fault conditions

The following fault conditions are commonly experienced during the exchange testing process.

1. Target node does not respond.
2. Composition of XML exceeds the server's available resources, causing a system failure
3. XML Payload parsing fails, but CDX GetStatus may indicate the payload was processed (thus making the assumption that the parsing successfully completed)
4. XML Payload is parsed, but causes database errors when written to a database (i.e. key violations).
5. Asynchronous follow-up activities never occur. (i.e. a solicit is never processed)

Testing the Data Exchange

Testing and the Data Exchange Design Process

Testing is an integral and essential part of the data exchange design process. Through testing, many issues will surface that will lead to altered data exchange design. For this reason, it is essential that data exchange partners not release the schema, Flow Configuration Documents (FCD), and other supporting documentation as final until after all testing is complete.

At least three partners (i.e. two states and EPA) should pilot any data exchange before the release of final versions of the documentation, schema and supporting materials. The first implementers typically identify and correct 70% of the problems that emerge during testing. The second implementation will identify and correct an additional 20%. The implementers will either manage or correct the remaining 10% through subsequent implementations.

Testing is an iterative process. Publishers and consumers of data must perform dozens of test transactions. They must examine each payload down to the data element level to ensure proper mapping and conversion of data elements. The process is tedious, but necessary to ensure the exchange is accurate and complete. Testing should be performed for all transaction types (i.e. each root schema and many different query parameters, if applicable) and should be performed by multiple entities. Several different instance documents of each transaction type should be used to help identify mapping problems or other anomalies which can easily escape notice if only a single instance file is used.

A successful end-to-end test should not only consist of a successful XML transmission from sender to receiver, it should entail successfully automated payload composition at the sender side (if applicable), successful transmission from sender to receiver node, and successful parsing and storage of the XML payload at the receiver side (if applicable). The IPT should determine the criteria for a successful test prior to beginning the testing process.

Exchange partners should also perform stress tests. These tests will help partners to discover the limitations of the data exchange (such as payload size) and may prompt the data exchange developers to add constraints to the data exchange, such as limiting payload size. Finally, program staff familiar with the data should follow a test plan developed by the IPT or developers, which validates that the data and modifications made in the source system (e.g. the state system) match the target database, prior to implementing a production exchange.

Even after exchange partners go through exhaustive testing, it is likely that they will encounter some issues that they cannot resolve. They should document these issues in the FCD so that future implementers of the data exchange will be aware of them and possibly address them. The architects of the next generation data exchange will need to be aware of these issues as well so they can attempt to resolve them.

Common Problems Discovered During Testing

As the testing process unfolds, problems will undoubtedly appear. Some known problems include but are not limited to:

- **Suspect data quality from one or more data providers.** For example, data is incomplete, invalid or inconsistent.

-
- **The data provider's infrastructure prevents certain data from being accessible to the Internet.** For example, the EPA does not provide Internet access to its Toxic Release Inventory (TRI) database. This prevents Exchange Network partners from querying the TRI database directly.
 - **Each partner's definitions of terms are inconsistent, making the data ambiguous.** For example, the definition of a closed facility in the Facility Identification data exchange may mean something different to each partner.
 - **Exchange partners do not use the same set of code lists.** Please see the *Crosswalk Codes* section of this document
 - **Node operations cannot handle too large sets of data, causing long delays or system crashes.** Please see the *Handling data-intensive transactions* section of this document
 - **Inadequate partner resources and scheduling conflicts.** Early in the project, exchange partners must allocate technical resources to ensure adequate staffing for building and testing. Furthermore, resources must be allocated to support the exchange once it is fully operational.
 - **Business rules may have been overlooked in developing the schema and or the conversion process, which are not uncovered until the parsing process.**

Each problem may force the Integrated Project Team (IPT) to reevaluate their approach to exchanging data. Every problem solved may require changes to the XML schema and will certainly require changes to the FCD. For this reason, it is important that data exchange designers to develop or change the schema and the FCD together.

Publishing the Data Exchange

As this Guidance is being written, there is no formalized the process of publishing a data exchange to the Network. Exchange Network governance is working to create standards and procedures for publishing a data exchange.

Ensuring Schema Design Conformance

Designers should formally test their XML schema for conformance with Exchange Network standards. The schema should be presented the final schema to the Exchange Network governance for conformance review. Upon completion of the conformance review process the schema, FCD and supporting exchange documents should be published to the Exchange Network Registry at [http://oaspub.epa.gov/emg/xmlsearch\\$.startup](http://oaspub.epa.gov/emg/xmlsearch$.startup).

Proper Versioning of Exchange Deliverables

It is important to version and timestamp each of the documents which are developed for the data exchange. One of the most significant and frustrating versioning problems occurs when multiple schema sets are released with the same version number. Even if only a single element in a schema changes, the minor version number should be incremented for all schema files in the release. Furthermore, revision numbers should not be used (i.e. version 1.3 revision 4). Corrections and updates to schema and supporting documentation are almost inevitable. Much confusion can be mitigated through careful and consistent usage of versions and timestamps on all deliverables. Specific guidance for versioning of Exchange Network deliverables is currently under development.

Data Exchange Design Best Practices

Best Practices for Schema Design

Schema should be designed in coordination with the data exchange design process. This will ensure the schema (or schemas) can accommodate the needs of the data exchange².

The quality of schema design can have a tremendous impact on the flexibility and extensibility of a data exchange. Several methods can help prolong the longevity and usefulness of a schema for the Exchange Network. The following schema design principles will help extend the useful life of the schema and ensure that development efforts are not being replicated.

Check the XML Schema Registry and bulletin boards

Schemas may already exist in the National Registry, or development efforts may be underway for a schema not yet registered. Prior to initiating development efforts on creating a new schema, research the National Registry or bulletin boards on the Exchange Network Web site.

Follow Exchange Network schema design standards

The Exchange Network has developed robust standards for schema design and approval. The prescribed conventions ensure maximum compatibility with other schema promulgated by the Exchange Network. The schema design standards are published on the Exchange Network web site and are in the process of being revised.

Exchange Network schema design guidance documents include but are not limited to:

1. EDSC Data Standards – (<http://www.envdatastandards.net/>)
2. Exchange Network Design Rules and Conventions (DRCs)
3. Exchange Network Namespace Guidance
4. Exchange Network Schema Review Process
5. Core Reference Model (CRM) Shared Schema Components (SSC) and Usage Guidelines

Schema developers should check with the Exchange Network Web site to ensure the latest schema design guidance is being used. A full discussion of these standards is outside the scope of this document.

Divide logical data groups into separate schema files

Schema components are often separated into schema modules which are stored in separate schema files. Each schema module may reference other schema modules (using schema *import* and *include* statements). This enables schema modules to be independent or combined in the future with other schema modules to allow data to be returned in a different format.

² For comprehensive Exchange Network schema design guidance, please see the documentation available at the Exchange Network Web site (<http://www.exchangenetwork.net/schema/index.htm>).

Use schema constraints effectively

XML Schema constraints (such as required fields, field lengths, enumerations, etc.) provide a powerful way to force XML instance files to comply with a strict set of rules, however developers should use extreme caution before adding restrictive rules to a schema.

The use of schema constraints should be driven in part by the type of data exchange being designed. Generally, schema designed for data publishing exchanges should use fewer element constraints. This will ensure that exchange partners which publish data using the schema will have more implementation flexibility. For example, by leaving most fields optional, data publishers will not be required to populate data element in the schema which they do not store in the source database.

For data exchanges which synchronize data with a defined target system (such as RCRAInfo), more rigorous use of schema constraints may be warranted to help ensure compatibility with the target database. Even with known target databases, systems can change, leaving schema constraints (such as embedded lookup lists) out of date. One must determine whether constraints limit the use of the schema by some other organization which may not have the same restrictions as the system to which the data exchange is originally targeting.

The use of constraints in schema may seem necessary to help enforce the validity of an XML instance file. However, schema constraints alone will not be able to provide all the validation required to ensure an XML instance file can be successfully processed by a data recipient. Tools such as Schematron³ can perform robust validation of XML instance documents. Schematron validation is also dynamically configurable, allowing business rule changes to be added to the Schematron processor at any time without the need to alter the XML schema.

Limit the use of required fields

Unless end uses absolutely require a field, consider making it an optional data element in the schema. Each required field places a demand on the data provider which they may or may not be able to satisfy. For example, the schema may make a Hydrologic Unit Code (HUC) required to describe the location of a feature. If the data provider does not have the data, they will most likely choose to fill the field with junk data (i.e. all zeros).

Schema are more flexible when child elements of a complex type are optional. This, in theory, can allow a large, complex schema to act as a “list” schema with no record detail. This increases the flexibility of the schema to become useful in other situations.

Use Count, List and Detailed Result Sets

Data providers can allow queries to return three different result sets; count, list and detail.

The Count result set returns the number of records that matched the criteria provided. This allows data requestors to decide whether they wish to continue with the query. For example, if the count query returned a very large number of records, the requester may decide to limit his or her query before asking for detailed result data.

The List result set returns a simple list of the items that the query produced (such as a list of facilities). This also limits the size of the result set and gives the requestor more detail than simply a count of records which matched their query.

³ Schematron is discussed more fully in the section labeled *Pre-validate Data before Submitting*

The detailed information results set returns all of the detail included in child records. This fully detailed set would be appropriate for partners performing data synchronization.

Count and List-type result sets are commonly needed for exchanges which allow ad-hoc querying of data from one or more nodes. If it is determined that an exchange will need to support count and list queries, the schema will need to be developed in such a manner that the result of these queries can be carried in the XML payload.

Best Practices for Handling Large Transactions

While it is ideal to design a data exchange which avoids returning large datasets, it is not always possible in practice. There are several ways to successfully manage unavoidable large data processing tasks:

1. **Only allow large transactions at certain times.** An Exchange partner may configure a node to only allow large queries when there is less node traffic. Typically, this would be during non-business hours.
2. **Allow only “Solicit” for queries which may yield large datasets.** Since “solicit” is an asynchronous method, this allows the data provider to process the request at the time of its choosing.
3. **Pre-process requests to estimate their impact on the node.** Exchange partners can use techniques to “cost” queries. Query costing is a relatively new Network concept and as such, no documentation yet exists on query costing methods.
4. **Pre-prepare data to simplify the conversion of relational data to XML.** Data can be staged in a relational database whose model is similar to that of the XML schema, which simplifies the data mapping to XML.
5. **Use flexible schema design.** Schema design can affect the options for returning data. For example, simplified “list schema” can serve as a lightweight data container for long lists while excluding bulky data detail. See the *Flexible Schema Design* topic below.
6. **Limit query size using “rowId” and “maxRow” parameters of the Query method.** A data exchange can designate a ceiling for the number of rows returned by each query.
7. **Limit query parameter choices.** An Exchange Partner can avoid large datasets by tailoring the allowable query parameters. For example, a query named “GetData” which required one or more query parameters as criteria would be less likely overburden the node server than a “GetAllData” service with no parameters.
8. **Consider more frequent exchanges of smaller datasets.** See the next topic for more detail
9. **Use file compression.** The Network Submit method allows one or more documents to be included in the transmittal. Consider sending the document as a compressed file in ZIP format.

Best Practices for Managing State

Perhaps one of the biggest challenges in a synchronization data exchange is keeping track of what partners do or do not submit. Partners may have further difficulties when a portion of submitted data is rejected, forcing the submitter to evaluate, fix and resubmit portions of data. There is no best way to solve this problem due to the varying architecture and interface of each

system participating in the exchange. Submitters may choose among several methods listed below to solve this problem.

Flagging data

Using this method, the submitter creates a series of exchange tables within the source system to store the submission status of each record. Often, there will be a record in the exchange table for each master record transmitted to the exchange partner. Maintaining a coarse level of detail (granularity) for a transaction will reduce the number of tables and records required by the exchange tables. Transaction granularity is covered in a later section.

Partners can populate an exchange table in either of two ways. One uses database triggers on the actual source system table. The other detects changes periodically through a batch process. The trigger method works well if the detection mechanism can track changes in child data reliably at the parent level. This allows the data exchange detection routine to log the modification in the exchange table at the appropriate level of data granularity. The batch process method checks data in the source system for changes. When the process detects a change, it adds a record to the exchange table flagging the changed data to be queued for transmission to the exchange partner.

Using data “differencing” to identify changed data

Some source systems make it quite difficult for a synchronization method to identify which data has changed. This may be the case if a given source system database does not store a timestamp for when records were created or last updated. Partners working with these systems can store a snapshot of the data upon the last submission to their exchange partners. The differencing process compares the snapshot of the previous transmission with the current data set. The comparison allows the process to identify and insert changed data into an XML file. This XML file will contain a record all new, updated and deleted data in the source system since the snapshot was taken. This is an especially good option when transmission of deleted data is important, since most databases do not store records of deleted data.

Incremental and Full-refresh services

The Facility Identification data exchange introduced options for both incremental and full refresh services. Submitters using this model send regularly-timed incremental changes to an exchange partner. Incremental refreshes are small and efficient. The data provider may also send a full refresh of data to their exchange partner on a less frequent interval. Full refreshes ensure that incremental exchanges do not inadvertently lose data. For example, incremental refreshes may occur weekly. Quarterly or annually, a full refresh of data will be dispatched to the exchange partner.

The challenge of tracking deleted data

Most databases do not archive deleted data. This can make it difficult to transmit a record to the exchange partner notifying them of the delete. The Full Refresh service solves this problem, by replacing an exchange recipient’s data with the current snapshot from the data provider. The data differencing technique also can solve this problem, since it detects missing (deleted) records by comparing the current dataset with the previous snapshot.

Use Crosswalk Code Lists

Network partners must insure that code lists used by the data provider are intelligible to and accepted by the exchange partner. Usually the data provider creates and maintains crosswalk

tables to map source system codes to destination system codes. The Exchange Network governance is working to address the management of shared code lists. Where possible, code lists when a common authoritative standard exists such as with CAS Numbers or ITIS codes, those standards should be used.

Best Practices for Data Services

For background information on data services, please see *Appendix B – Data Exchange Models*.

Use Appropriate Data Service Names

Data Service names should support the following goals:

- **Simplicity:** Keep the naming convention as simple as possible while ensuring uniqueness
- **Flexibility:** Allow the naming convention to be flexible so that new Data Services can easily be added to flows
- **Network-wide consistency:** Data Services will be named using similar patterns
- Query and Solicit transactions must utilize the same Data Service name

Data Service names:

- Must be unique across Exchange Network
- Must be less than 50 characters
- Must contain major and minor version number of the schema used by Data Service

The naming convention is as follows:

{FlowName}.{Method}**{Object}[By{ParameterName(s)}]_v{Version}**

Data Service names must use UpperCamelCase to distinguish words within the data service names. White space is no allowed.

The following table describes each service name component:

Part Name	Description	Example
FlowName	The data exchange name, system name, responsible party or environmental interest.	NEI, RCRA, TRI
Method	The verb describing the type of action to perform. Get will be used for all Query and Solicit transactions.	Get, Submit, Other
Object	The principle entity or module affected by the operation	Facility, User, Policy
ByParameter(s)	Parameter(s) describe the constraint by which the Service is executed. It is <u>optional</u> and can be used at the discretion of the Data Service creator. It is envisioned that this will primarily be used when there is only one main parameter as a constraint	Fixed parameter services: ByName, BySIC, Dynamic-parameter services: ByParameters

	on the Data Service. If a Data Service accepts two or more primary parameters, the Data Service name can use “GetDataByParameters” to indicate it accepts multiple parameters.	
Version	The Version number component of a Data Service name indicates the version of the schema used in the service (i.e., for queries and solicits, the version of the schema that is used to generate the response). Since this ties Data Service names to version of schema used, incrementing the schema version requires a change to Version component of the Service name. However, if an existing Data Service is changed (by, for example, adding a parameter), a new Data Service is created that must be given a new name – even if the schema (and version of schema) used in generating the return remains the same.	v1.0

Fully Document Each Data Service

Each data service must be fully documented in the FCD for the data exchange. The following information should be included for each data service⁴:

- Data Service name
- Whether the service is supported by Query, Solicit, or both
- Parameters
 - Parameter Name
 - Index (order)
 - Required/Optional
 - Minimum/Maximum allowed values
 - Data type (string, integer, Boolean, Date...)
 - Whether multiple values can be supplied to the parameter
 - Whether wildcard searches are supported and default wildcard behavior
 - Special formatting considerations

⁴ This list is an elaboration of the data service descriptors provided in the Flow Configuration Document template, which is not available on the Exchange Network Web site at the time of this writing.

-
- Access/Security settings
 - Return schema
 - Special fault conditions

Non-published standards for Data Services

Default behavior for Data Services has not been formally defined, but conventions have been established. These conventions are subject to revision if and when formal guidance is developed.

Delimiters The de facto standard is to separate multiple values in a single parameter with a pipe character (|). For example, if a data service has a parameter for Zip Code, the user could supply a list of five ZIP codes to the query separated by the pipe character.

Wildcards The percent character (%) is the most common choice although no formal guidance has set this as a standard for the Exchange Network.

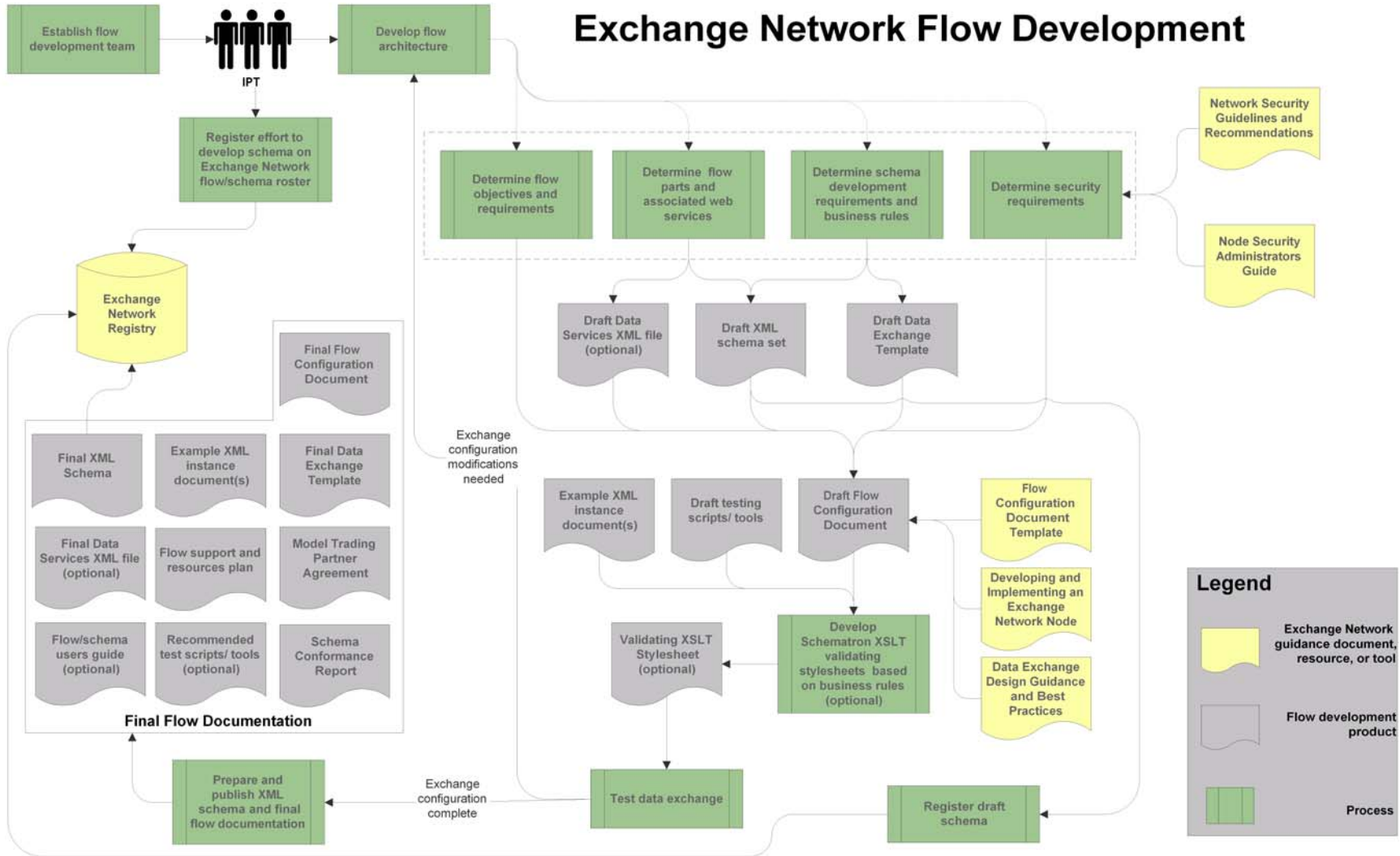
Criteria If two or more parameters are defined for a query service, it is expected that the criteria will be additive, otherwise known as an AND condition. For example, if SIC Code and CountyName are provided to a Facility Identification Data Service, the service would return facilities which exist in the county **and** have a matching zip code.

Best Practices for Data Validation

CDX is beginning to offer a series of services which allow submitters to pre-validate XML instance files before submitting to CDX. Using Schematron technology, the CDX document validation service provides much more robust validation than can be provided by schema alone⁵. Schematron can enforce data relationships between fields, a task which can not be performed by XML schema alone. For example, a Schematron rule may state that an XML instance file must either contain data for a First Name and Last Name *or* a Full Name for an individual.

⁵ The CDX document validation service is available at <http://tools.epacdxnode.net/>

Appendix A – Data Exchange Development Roadmap



Appendix B – Data Exchange Models

The Data Synchronization Model

Data synchronization flows are to make comparable data the same in two databases. Partners often use this design for traditional data submissions from state regulatory agencies to EPA. Examples include data for the National Emissions Inventory (NEI) or Permit Compliance System (PCS). Often, these exchanges are extensions of or replacements for data flows which existed before the Exchange Network. Typically, a state agency uses a data synchronization exchange for routine updates to EPA systems.

This exchange scenario represents the prevalent philosophy among the first data exchanges on the Network. More recently, this model has been avoided in place of the data publishing model which is described in the following section. While this model is still supported on the Exchange Network, data exchange designers are encouraged to examine the data publishing model and evaluate its appropriateness for any new data exchange.

Advantages

- Using the Exchange Network can at least partially automate what has been traditionally a manual process
- Since the Exchange Network requires that data be formatted in XML, tighter data validation can be performed, potentially reducing the number of errors on the receiver side
- Exchange Network infrastructure can be reused to perform the transfer of data between partners, potentially reducing the number of paths, vehicles and security mechanisms used to move data between agencies
- Each partner's network node serves as a central portal for information exchange. By aggregating data send/receive services, administrative overhead may be reduced

Disadvantages

- Maintaining records of what data has been sent forces the submitter to determine what data is already in the receiver's system and whether it is up to date.
- Finding out why some data is rejected by the receiver can be very difficult.
- Error feedback processing is not fully automated for many data exchanges. This requires tedious human auditing, correction and resubmission of rejected data.
- Replacing existing mechanisms for submitting data may not provide any advantage over traditional data submission formats or mechanisms, other than 'upgrading' or adhering to new standards.
- Data synchronization often involves constructing and transporting very large payloads which are not ideal for Web services. Constructing large datasets (XML serialization) can be very server resource intensive.

-
- XML is very verbose, greatly inflating the size of already large datasets to be transported across the Internet⁶.

The Data Publishing Model

With the data publishing model, the requestor does not attempt to store all of the service provider's data, since the publisher can supply any chunk of data to the requestor on demand. The data publishing data exchange design model allows for much more frequent and timely exchanges of data.

Data services are the defining feature of a data publishing exchange. A data service is a piece of functionality (or service) exposed to the Internet by an Exchange Network node. Typically, this service is available at all times for an authorized data requestor to query.

Most often, a data service will simply return requested data in the form of an XML document. For example, the Facility Identification data exchange offers a service named **GetFacilityByName**. The data requestor passes a name such as "ABC Chemical Company" and the data service returns data for all facilities which contain the text provided in the facility name. In this instance, the data service merely returned data based on criteria provided.

A data service may also allow an authorized external party to update data stored on the service provider's system. For example, the NAAS offers a service named **AddUser** that allows an authorized entity to create a user account on the Exchange Network security service.

As mentioned, Data Services require the requestor to provide criteria or parameters. Depending on how the Data Service is constructed, the service can fall into two major categories, fixed-parameter and dynamic-parameter data services. These two service types are described in the following section.

Advantages

- The data provider needs no knowledge of the data requestor's system in order to fulfill a request for data. For this reason, no complex data structures are needed to maintain what the requestor already does or does not know or have.
- "On demand" availability of data. Since services are typically always available, the requestor may retrieve data at any time.
- Scalability. The number of data requestors can expand greatly without any need to change the data provider's system.
- XML messages are typically small and focused, allowing for quick turn around between request and response

Disadvantages

- Not ideal for traditional state-to-EPA data exchanges which require that the data provider "feed" the federal system

⁶ File size can be greatly reduced by compressing or "zipping" XML before sending. Data is then decompressed before being parsed by the receiver.

Fixed-Parameter Data Services

Each data service takes one or more criteria. Using the Facility Identification example above, the **GetFacilityByName** takes two criteria, a state abbreviation (“MI”) and a search term (“ABC Chemical Company”). This is a fixed-parameter data service since each parameter is required and the parameter list is typically short.

There are two shortcomings to this approach. Firstly, a large number of data services are required to give the user flexibility to query on different criteria. For example the Facility Identification exchange defines eight fixed parameter queries such as **GetFacilityByName**, **GetFacilityByCountyName**, **GetFacilityByLocalityName**, **GetFacilityBySICCode** and so on⁷. Maintaining a large number of services adds clutter to the service list and may add maintenance overhead. Secondly, these services do not allow the requestor to mix and match criteria. For example, what if a requestor wanted to view all facilities in a given county which belong to a particular industrial classification (SIC)? Dynamic-parameter data services offer the flexibility to solve this problem.

Dynamic-Parameter Data Services

A data exchange may also define a single service that can take one or more of several optional parameters. Data requestors can then pass in a variety of criteria to construct the specific query of their choosing. Appendix C of the Facility Identification FCD defines 27 parameters into a single, consolidated data service.

Using a Combined Approach

Not all data exchanges are wholly one type or another. For example, a node may act as an intermediary between two or more partners. One production example of this is the TRI data exchange processor at EPA’s Central Data Exchange (CDX) node. When a TRI report is received by EPA from a reporting facility, the EPA TRI processor automatically forwards a copy of the report to the appropriate state node.

In this example, the TRI data exchange does not clearly fall within the definition of either exchange design described above. Since it does not offer “come and get it” services to state partners, it is not a data publishing model. While the TRI data exchange does contain some aspects of a data synchronization exchange, it does not fully synchronize with the state database. Furthermore, the data exchange frequency is dynamic in nature and contains small, focused data sets, thus deviating from the broad definition of a data synchronization design model.

Although no such data exchange exists today, it is possible that one would contain characteristics of both the data publishing and data synchronization models. For example, the next generation TRI data exchange envisions that states will be able to query TRI data from EPA on demand. This data publishing service would supplement the current “push” of TRI data from EPA to state partners.

⁷ A full list of Facility Identification Data Services can be found in the Facility Identification FCD available at http://www.exchangenetwork.net/exchanges/cross/frs_fcd_v1_061804.doc

Appendix C – Data Exchange Design Patterns

Data exchange design patterns are common, recurring methods used for transacting data over the Exchange Network. These themes are independent of the data being exchanged.

This section addresses patterns in two areas: network security and the regular exchange of data over the network. When designing a data exchange, one or more of these patterns are likely to be part of the final design. Following patterns make the data exchange easier to implement and more easily understood.

Patterns in Authentication and Authorization

NOTE: The Exchange Network Protocol 1.1⁸ outlines many of these same data exchange scenarios more fully.

There are three major methods in which partners can negotiate credentials before initiating a data exchange. The OTS considers neither solution superior. The choices are either the **Direct Authentication** model or the **Delegated Authentication** model. A third option, **Local Authentication**, is also available. Partners should choose and document one of these models for the data exchange.

Exchange Network security primer

The Network Authentication and Authorization Service (NAAS) is a centralized service which maintains information about all valid users on the Exchange Network and the privileges associated with each user's account. Each node must interact with NAAS to ensure that participants in the exchange have sufficient privileges to engage in the transaction of data.

NAAS issues *tokens* to valid users upon authentication. This token is then passed between exchange partners with each invocation, ensuring the authenticity and authority of the partner over the course of the exchange. Tokens expire after a set time period (10 minutes at the time of this writing), further increasing the security of the data exchange.

Direct authentication model

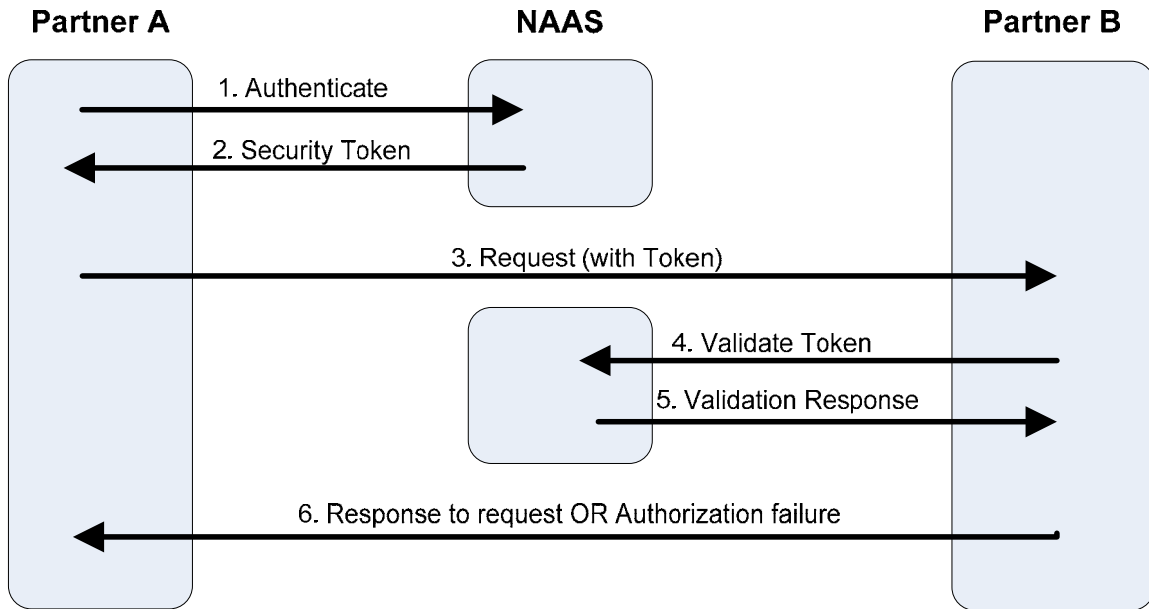
In the direct authentication model, the initiating partner obtains a security token directly from NAAS before contacting the intended recipient. Exchange partners using the direct authentication model take the following steps:

1. Partner A sends a username and password to NAAS
2. NAAS ensures the user is valid and returns a security token to Partner A.
3. Partner A then sends a request to Partner B, along with the security token
4. Partner B then checks that the token is valid with NAAS
5. NAAS responds to Partner B indicating that the token is valid

⁸ The Exchange Network Protocol and Functional Specification can be found at http://www.exchangenetwork.net/node/dev_toolbox/network_exchange_protocol_v1.1.doc

-
- Partner A and Partner B initiate the data exchange, passing the token between each partner with each request.

The following diagram illustrates this arrangement.



Direct Authentication Model

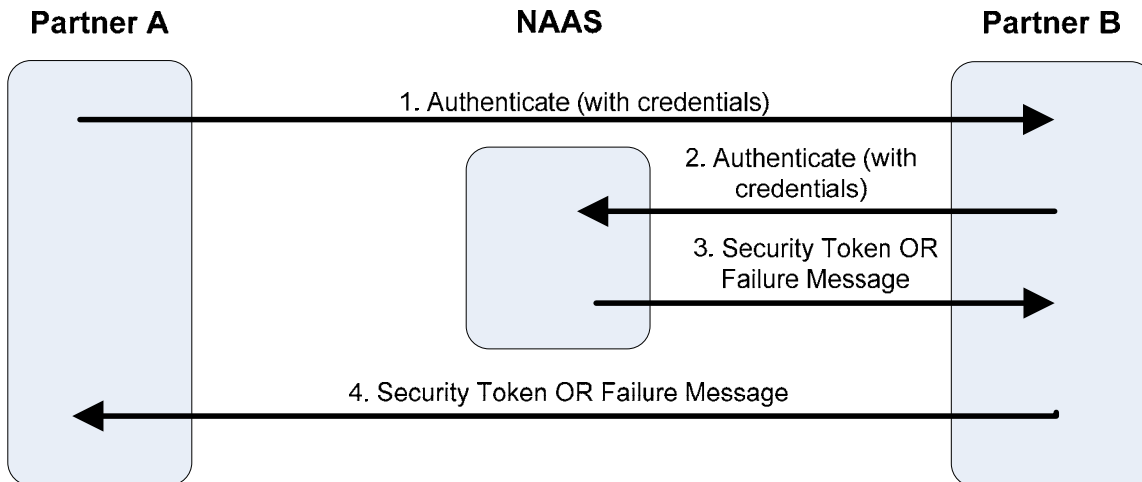
This type of validation is ideal for data exchanges which do not involve CDX since Partner A's credentials are only visible to NAAS. Partner B only sees a security token and is not privileged to Partner A's user name and password, which is the case in the delegated authentication Model.

Delegated authentication model

Partner A sends security credentials directly to Partner B. In turn, Partner B validates the credentials against the NAAS service. The steps in this process are outlined below:

- Partner A sends a request to authenticate, with username and password to Partner B
- Partner B forwards A's authentication request to NAAS
- NAAS responds to Partner B with either a security token if the request is successful or an error message when there is a problem with the request such as an incorrect password or username
- Partner B either continues or aborts the exchange based on the response from NAAS

The following diagram illustrates this arrangement.



Receiver-Initiated Security Validation

Using this model, Partner B acts as a proxy to the NAAS service. This arrangement is less complex for the initiator, Partner A, since only Partner B is involved with the security validation exchange (from Partner A’s point of view). The initiating partner, A, makes only one method call.

Delegated authentication is most appropriate for data exchanges with CDX, since CDX can validate against NAAS using inbound credentials. This arrangement is less appropriate for non-CDX exchanges since the receiver has direct access to the sender’s credentials. This arrangement presents a security risk to the initiating partner should the receiving partner mishandle Partner A’s credentials.

Local authentication model

A final method for negotiating security is to use local authentication. This model bypasses the NAAS entirely. Using this model, the requestor must have a user account on the service provider’s system. This model is only appropriate for non-regulatory flows which do not go through CDX. The local authentication model is only appropriate for data exchanges between two partners. This authentication model is not addressed in the Exchange Network specification.

Patterns in Requesting Data

There are four ways for Partners to request data using the Exchange Network. Only the first method, Simple Query, allows for immediate data transactions. The other three methods allow the data provider to respond at a later time.

This guidance does not discuss error conditions in each of the examples below. Partners must determine how to handle flow-specific errors since each data exchange has individual characteristics. Please see the *Implementing Error Handling* section for more information on this topic.

NOTE: Partners must perform security negotiation using one of the three methods discussed in the previous section prior to exchanging data.

Simple Query

The simplest possible communication between two Exchange Network partners is the simple query. The following table illustrates how the data exchange occurs using this configuration:

Event	Conversation	Network Method ⁹
1. Partner A requests information from Partner B	Partner A: Show me all the facilities in your system located in Zip code 12345	Query
2. Partner B responds with the requested information	Partner B: Here is the list (hands over list)	queryResponse

Simple queries are a *synchronous* transaction meaning Partner B processes the request and replies to Partner A immediately. This arrangement is appropriate for small datasets only, since immediate processing of large datasets may tie up Partner B's server resources.

The following three data exchange methods are *asynchronous*, meaning that the response to a request occurs at a later time. These scenarios are ideal for operations which require a long time to process. Asynchronous transactions also allow a node to process intensive requests for data during the time of its choosing, such as during the night when contention for server resources is minimal.

Solicit with Download

This method allows Partner B to process a request for data at a later time. The Solicit with Download method requires the data requestor to check back periodically to see if the data is ready.

The following table illustrates an exchange of data using Solicit with Download:

Event	Conversation	Network Method
1. Partner A requests information from Partner B	Partner A: Show me all the facilities in your system located in Zip code 12345.	Solicit
2. Partner B responds response with a transaction ID, which serves as a tracking number for Partner A to come back later and retrieve the document	Partner B: I will process that request at a later time. You can check on the status of your request by referencing tracking number is 1234.	solicitResponse
...time passes...		
3. At a scheduled time (usually an off hour) Partner B processes the request and composes a response	N/A	N/A
4. Partner A checks back with Partner B to see if the request has been processed	Partner A: Do you have a response to my question. My tracking number is 1234.	GetStatus

⁹ The method and method response (i.e. Solicit and solicitResponse) represent a single unit of work and should not be viewed as two autonomous actions.

5. Partner B responds that the document is either ready or not	Partner B: Yes, you may download it at any time.	getStatusResponse
6. Partner A requests to download the response	Partner A: I would like to download the response. My tracking number is 1234.	Download
7. Partner B sends the document to Partner A	Partner B: Here is the document	downloadResponse

Solicit with Submit

This method allows Partner B to process a request for data at a later time. With Solicit with Submit the data provider automatically submits the data set containing the query response to the data requestor once it becomes available.

The following table illustrates Solicit with Submit:

Event	Conversation	Network Method
1. Partner A requests information from Partner B	Partner A: Show me all the facilities in your system located in Zip code 12345. When you are done getting the data together, send it to this URL: (insert URL here)	Solicit
2. Partner B responds response with a transaction ID, which serves as a tracking number for Partner A to come back later and retrieve the document	Partner B: I will get back to you on that. Your tracking number is 1234.	solicitResponse
...time passes...		
3. At a scheduled time (usually an off hour) Partner B processes the request and composes a response	N/A	N/A
4. Partner B submits the data back to Partner A	Partner B: Here is the payload you were expecting.	Submit
5. Partner A receives the data and sends a reference number back to Partner B	Partner A: Got it. Here is the reference number for your submission: 4567	submitResponse

Solicit with Notify

With Solicit with Notify the data provider alerts the data requestor when the data is ready for retrieval.

This data exchange method is not commonly used since it relies on three separate calls across the Network to complete; Solicit, Notify, and finally Download. Furthermore, each call is executed by an alternating partner. Such an exchange must be heavily orchestrated such that state of the transaction is maintained by both sender and receiver over a possibly lengthy time interval.

The following table illustrates an exchange of data using Solicit with Notify:

Event	Conversation	Network Method
1. Partner A requests information from Partner B	Partner A: Show me all the facilities in your system located in Zip code 12345. When you are done getting the data together, notify me at this URL: (insert URL here)	Solicit
2. Partner B responds response with a transaction ID, which serves as a tracking number for Partner A to come back later and retrieve the document	Partner B: I will get back to you on that. Your tracking number is 1234.	solicitResponse
...time passes...		
3. At a scheduled time (usually an off hour) Partner B processes the request and composes a response	N/A	N/A
4. Partner B notifies Partner A that the response is ready to be picked up	Partner B: The answer to your question is ready.	Notify
5. Partner A attempts to retrieve the document	Partner A: I wish to download the document now.	Download
6. Partner B responds with the document containing the answer to Partner A's original request	Partner B: Here you go (hands over document)	downloadResponse

Patterns in Submitting Data

Submitting data is an operation where a data provider is responsible for sending data to one or more partners. This is appropriate for traditional regulatory exchanges of data, referred to earlier in this document as a data synchronization data exchange.

Simple Submit

The Simple Submit delivers data from one partner to another in a single transaction.

The following table illustrates how the data exchange occurs using this configuration:

Event	Conversation	Network Method
1. Partner A submits data directly to Partner B	Partner A: Here is my annual submission of NEI data (passes report)	Submit
2. Partner B receives the data and sends a reference number back to Partner A	Partner B: Got it. Here is the reference number for your submission: 9876	submitResponse

Notify with Download

With this method, after the submitter gives the recipient notice that the data is ready to be picked up; the recipient can choose the time of its retrieval.

The following table illustrates how the data exchange occurs using this configuration:

Event	Conversation	Network Method
1. Partner A notifies partner B that a document is ready to be picked up	Partner A: I have data ready for you. You can get it at this URL: (insert URL here)	Notify
2. Partner B requests to download the file	Partner B: I would like to download the file you have ready.	Download
3. Partner A sends the file to Partner B	Partner A: Here is the document	

Other Data Exchange Scenarios

The previous examples illustrate the most common arrangements for transacting data between partners. The Exchange Network Protocol and Specification contain definitions for additional data services not included in this discussion. Data exchange designers should investigate other options if the approaches outlined here can not satisfy the requirements of the data exchange.