

```
XMLSchema" xml version="1.0" encoding="UTF-8"
engenetwork" _ <xsd:schema
rmDefault="unqualified" targetNamespace="http://www.epa.gov/exchangenetwork"
mon_v3_0.xsd" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:nei="http://www.epa.gov/exchangenetwork"
elementFormDefault="qualified"
version="3.0">
<xsd:include schemaLocation="EN_NEI_Common_v3_0.xsd"
- <!--
XML 3.0 Start of Schema Header
-->
<xsd:annotation base="http://www.w3.org/2001/XMLSchema"
angenetwork" </xsd:documentation>
Point</xsd:documentation>
NEI XML 3.0 Point data</xsd:documentation>
Available: http://www.epa.gov/exchangenetwork/
by Environmental Protection Agency
input format</xsd:documentation>
encoding="UTF-8" ?> user</xsd:documentation>
space="http://www.epa.gov/exchangenetwork"
http://www.w3.org/2001/XMLSchema" encoding="UTF-8"
http://www.epa.gov/exchangenetwork"
Default="qualified" attributeFormDefault="unqualified"
>
schemaLocation="EN_NEI_Common_v3_0.xsd"
Default="qualified"
leader
chemaLocation="EN_NEI_Common_v3_0.xsd"
on>
mentation> Schema Name: NEI XML 3.0
sd:documentation>
mentation> Current Version: 1.0
e: http://www.epa.gov/exchangenetwork/
ion>
mentation> Description: The NEI XML 3.0 Point data
rmat</xsd:documentation>
mentation> Application: Varies by user
sd:documentation>
mentation> Developed By: Environmental Protection Agency
ding="UTF-8" ?>
http://www.epa.gov/exchangenetwork/
/www.w3.org/2001/XMLSchema"
/www.epa.gov/exchangenetwork"
t="qualified" attributeFormDefault="unqualified"
aLocation="EN_NEI_Common_v3_0.xsd"
ion> Schema Name: NEI XML 3.0
cumentation>
on> Current Version: 1.0
//www.epa.gov/exchangenetwork</
> Description: The NEI XML 3.0 Point data
f:documentation>
Application: Varies by user
```

# Data Exchange Design Rules and Conventions (EDRC) for the Exchange Network

Version: 1.0

Revision Date: 01/12/2010



THIS PAGE INTENTIONALLY LEFT BLANK

# Table of Contents

<b>1. Introduction</b>	<b>1</b>
1.1. About XML Design Rules and Conventions (DRCs)	1
1.1.1. Audience	1
1.1.2. Background	1
1.2. Semantic Conventions used in this Document	2
1.3. Component Naming Syntax Conventions	3
1.4. Exceptions to Design Rules	3
<b>2. Exchange Design Rules and Conventions</b>	<b>4</b>
2.1. General Exchange Design	4
2.1.1. Exchange Identification	4
2.1.1. Exchange Query and Solicit Services	5
2.2. Exchange Development and Publishing	6
2.2.1. Communication during Exchange Development	6
2.2.2. Publishing an Exchange	8
2.3. Exchange Component Versioning	10
2.3.1. General Versioning Principles	10
2.3.2. Versioning Data Services	14
2.3.3. Upgrading Existing Exchange Components	15
2.4. Exchange Documentation	16
2.4.1. The Flow Configuration Document (FCD)	16
2.4.2. The Data Exchange Template (DET)	18
2.5. Query and Solicit Services	18
2.5.1. Service Names	18
2.5.2. Service Parameters	20
2.5.3. Service Documentation	21
2.6. Exchange Network Header	22
2.6.1. Header Payload Element	24
2.6.2. Header Documentation	25
2.6.3. Header Encryption and Security	26
2.6.4. Header Versions	28
<b>Appendix A – Summary of Design Rules</b>	<b>31</b>
<b>Appendix B – Glossary of Terms</b>	<b>36</b>

THIS PAGE INTENTIONALLY LEFT BLANK

# 1. Introduction

## 1.1. About XML Design Rules and Conventions (DRCs)

This guide establishes design rules and guidelines for the creation of data exchange definitions and documentation for joint use by the U.S. Environmental Protection Agency (EPA), state and local governments, tribes, and territories exchanging data on the National Environmental Information Exchange Network (Network).

### 1.1.1. Audience

The primary audience for this guide is schema and data exchange developers, Integrated Project Teams (IPTs) and implementers. This includes all individuals involved with the design and development of a new or upgraded data exchange including both technical and business experts. This guide applies to all agencies and organizations involved with designing, developing, and implementing data exchange components for the Network.

### 1.1.2. Background

Published in 2003, the XML Design Rules and Conventions v1.0 document was one of the first formal published rules for development and management of data exchanges on the Network. Since that time, the Network has evolved greatly. The Network devised a comprehensive component versioning strategy, the XML namespace conventions were changed, and shared tools such as the Shared Schema Components and Exchange Network Header were developed, to name a few.

With the evolution of the Network, many discrete guidance documents were published. Over time, the volume of documentation swelled, making it difficult for schema and exchange developers to gather and comprehend all of the information needed to properly design an exchange. Complicating matters, information sometimes conflicted between documents.

The release of this document, Exchange Design Rules and Conventions v1.0 (EDRC v1.0), along with the companion document XML Design Rules and Conventions v2.0 (DRC v2.0), consolidates many of the past guidance papers into two comprehensive documents.

The following documents are superseded by the rules and guidelines in DRC v2.0 and EDRC v1.0:

- XML Design Rules and Conventions for the Environmental Information Exchange Network (September, 2003)

- XML Schema Design Rules and Conventions v1.1 – Interim Update for the Exchange Network (April, 2006)
- Principles, Rules, and Procedures for Change Management on the Exchange Network (December, 2006) including Appendix A and B
- Flow Documentation Checklist v2.0 (December, 2007)
- Namespace Organization, Naming and Schema File Location v1.11 (March, 2006)
- Shared Schema Components Technical Reference and Usage Guides v2.0 (May, 2006)
- Exchange Design Guidance and Best Practices for the Exchange Network v1.2 (August, 2006)

Additional discussion and background information on topics covered in the DRC v2.0 and EDRC v1.0 can be found in these documents if needed, however readers should be aware that some concepts presented in the historical documents may be out of date and in conflict with the latest rules and guidance.

## 1.2. Semantic Conventions used in this Document

Each rule or guideline has been distilled into a single, normative statement describing what a schema or exchange developer **MUST**, **MUST NOT**, **SHOULD**, **SHOULD NOT**, or **MAY** do. The specific meaning of these terms was originally defined in the Request for Comments 2119 issued by the Internet Engineering Task Force<sup>1</sup>. The terms are defined as follows:

- **MUST**. This word, or the terms “REQUIRED” or “SHALL,” means that the definition is an absolute requirement of the specification.
- **MUST NOT**. This phrase, or the phrase “SHALL NOT,” means that the definition is an absolute prohibition of the specification.
- **SHOULD**. This word, or the adjective “RECOMMENDED,” means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- **SHOULD NOT**. This phrase, or the phrase “NOT RECOMMENDED,” means that there may exist valid reasons in particular circumstances when the particular behavior

---

<sup>1</sup> Internet Engineering Task Force, Request for Comments 2119, March 1997, [www.ietf.org/rfc/rfc2119.txt?number=2119](http://www.ietf.org/rfc/rfc2119.txt?number=2119).

is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

- **MAY.** This word, or the adjective “OPTIONAL,” means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor believes that it enhances the product, while another vendor may omit the same item. An implementation that does not include a particular option **MUST** be prepared to interoperate with another implementation that does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option **MUST** be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

### 1.3. Component Naming Syntax Conventions

Naming conventions for components such as schema files, namespaces, and documents are used throughout the design rule documentation. The following conventions are used:

Convention	Description
{ }	Placeholder for user-supplied word or term. Text inside the placeholder is the name for the term.
“ ”	Text included in quotes must be used verbatim in the component name.
[ ]	Items in brackets are optional.

For example, the naming convention for a component might be described as:

**{ExchangeIdentifier}”\_v”{MajorVersionNumber}”[{RevisionIdentifier}]**

Below are two examples of component names that properly implement the convention above:

- MyExchange\_v2
- ABC\_v3a

### 1.4. Exceptions to Design Rules

While the rules and guidelines in this document describe specific requirements for designing schema and data exchanges, Network governance recognizes that there will be scenarios where certain rules are not practical or cannot be met. In these cases, schema and exchange developers should document the discrepancies and the justification for deviating from the rules. Exchanges that deviate from rules will most often still be deemed acceptable for use on the Network and will be evaluated on a case-by-case basis during the Exchange Documentation Package review process.

## 2. Exchange Design Rules and Conventions

### 2.1. General Exchange Design

The Network allows states, territories, tribes, and EPA to exchange data using an agreed set of protocols and standards. Each type of information exchanged over the Network is a **data exchange**. A data exchange is often called a flow, data flow, or simply an exchange in Network discussions. A data exchange is the sharing of data between two or more Network partners implemented through a specific set of shared services and functions, typically related to a specific environmental subject, program, or dataset.

#### 2.1.1. Exchange Identification

Each exchange must be given a unique name or identifier. Typically, the exchange identifier is a short term or acronym. Examples of exchange identifiers include the following:

- **TRI** - Toxic Release Inventory
- **WQX** - Water Quality Exchange
- **FACID** - Facility Identification Exchange

The exchange identifier must be used consistently throughout the exchange documentation to identify the exchange. This ensures clarity and consistency across exchange components. In cases where an XML schema is developed specifically for the exchange, the exchange identifier should also be used as the namespace prefix in the schema.

In addition, the exchange identifier must be used as the value for the “dataflow” parameter in all Network primitive operations that require a dataflow to be specified. One special consideration is that the exchange identifier must be suffixed with the major version number of the data exchange, separated from the exchange identifier by an underscore and the letter “v”. For example, the dataflow parameter for first major version of the ABC exchange would be **ABC\_v1**.

It is the responsibility of the exchange development team to choose an exchange identifier and to ensure that the identifier has not already been claimed by an existing exchange.

### *General Exchange Design – Exchange Identification*

<b>Rules and Guidelines</b>	
<b>Rule</b>	<b>Description</b>
[XD1-1]	An exchange <b>MUST</b> be prescribed an exchange identifier in the form of a single term or acronym (e.g. TRI, WQX, FACID).
[XD1-2]	The exchange identifier <b>MUST</b> be used consistently throughout the schema and documentation, including the "dataflow" parameter on all Network primitive methods associated with the exchange, namespace prefixes, and schema file names.
[XD1-3]	The "dataflow" parameter <b>MUST</b> match the exchange identifier prescribed to the exchange and <b>MUST</b> be formatted as follows: {ExchangeIdentifier}_v{MajorVersionNumber}.
<b>Justification</b>	
Prescribing a unique identifier to each exchange helps to ensure the exchange is easily identifiable and that all exchange components can be easily recognized as relating to the exchange.	

#### **2.1.1. Exchange Query and Solicit Services**

Data Exchanges generally follow one of two general designs: either a sender-initiated “push” model, or a consumer-initiated “pull” model. These two models are commonly referred to as a data synchronization exchange and a data publishing exchange, respectively. “Push” models typically are implemented around the Submit primitive method, while “pull” models make use of Query and Solicit methods to initiate partner interactions.

Documents have been written and published on the Exchange Network website that contrast each design as well as document the potential merits and drawbacks of each model. These discussions will not be repeated here.

The Network does not have an official position on when a given approach should be used. Either design approach is acceptable. There is an understanding that the exchange architects are best suited to determine how an exchange should be modeled; however, it is desirable to offer Query and Solicit services as part of the design of any exchange, even if they are not required as part of the core business process.

Query and Solicit services promote the use of the Network for ad hoc querying and increase the overall visibility of data on the Network. Query and Solicit services are understood to be the principle mechanism for growing the Network by continually adding to the library of available datasets and query functions available to Network partners.

### *General Exchange Design – Exchange Query and Solicit Services*

<b>Rules and Guidelines</b>	
<b>Rule</b>	<b>Description</b>
[XD1-4]	Query and Solicit services SHOULD be defined, even for exchanges that are primarily designed for data reporting purposes.
<b>Justification</b>	
Query and Solicit services help the Network to grow by continually adding to the library of available datasets and query functions available to Network partners.	

## **2.2. Exchange Development and Publishing**

This section provides rules and guidance for exchange developers during the exchange development process. This section discusses the following three areas:

- Designing and Testing an Exchange
- Publishing an Exchange
- Communication Guidelines

### **2.2.1. Communication during Exchange Development**

This section describes communication guidelines between exchange designers and Network governance<sup>2</sup> as well as other exchange stakeholders outside the design team.

#### **Communication with Network Governance**

There are two times when IPTs should communicate with Network governance during the exchange design process; first upon the outset of a new exchange development project, and again after the design of the exchange is complete but before beginning development of exchange software or XML schema.

The first communication ensures that governance is aware of the project so that it can be tracked. It also provides an opportunity for governance to provide the exchange design team with information on tools, resources, and prior work products that may be beneficial to the team. This communication also serves to introduce the project team to services offered by governance and to the exchange review processes required for gaining approval of work products for use on the Network.

---

<sup>2</sup> “Governance” is used throughout this document as a general term. Exchange developers should review the Network governance structure on the Network web site or contact the Network coordinator to determine the appropriate governance group to contact for a particular purpose. Governance may refer developers to an authorized third party for some support requests.

The second communication should occur when the conceptual design of new or modified exchange components is complete but before development begins. The purpose of this communication is to provide governance with an opportunity to review the draft exchange design and to provide feedback. This feedback is aimed at leveraging the collective knowledge of governance workgroup members to validate the design and suggest possible improvements. Exchange design teams are not required to make changes based on this feedback; the service is provided for information only.

In addition, Network governance may be able to offer schema review services to flow developers. This can be useful to help identify any significant deviations from Network schema and XML design rules. It is ideal to identify issues before significant development begins against an XML schema that does not comply with design rules.

### **Communication with Exchange Stakeholders**

Communication is important to ensure a smooth transition between versions of an exchange. The IPT is responsible for establishing official channels for communication about the exchange. IPTs may find it advantageous to establish two channels of communication: one channel would serve as a general purpose channel that has a broad subscription of partners interested in receiving information on significant updates, the second channel should be reserved for communication by IPT members for higher volume communication for flow partners actively engaged in development, testing, and implementation.

Once the communications channels have been established, the IPT is responsible for ensuring the Network website administrator has up-to-date information on how interested parties can access the communications channels. The Exchange Network website administrator will send notifications to the Network community of significant events as communicated by the IPT.

The IPT is responsible for informing current exchange implementers of implementation timelines. The IPT needs to communicate to exchange implementers whether an upgrade is required and when it must be implemented. If the current version will continue to be supported, the IPT must provide guidance on the schedule for phasing out prior versions of the exchange.

*Exchange Documentation – Communication Guidelines*

<b>Rules and Guidelines</b>	
<b>Rule</b>	<b>Description</b>
[XD2-1]	IPTs SHOULD include members from all stakeholder groups.
[XD2-2]	IPTs MUST inform Network governance at the outset of a new exchange development project.
[XD2-3]	Exchange developers SHOULD engage Exchange Network governance for review of schema and flow architecture design before beginning schema or software development.
[XD2-4]	After initial planning of an exchange upgrade, the IPT SHOULD publish a pre-change notification that outlines the planned changes, which partners will be impacted by the changes, whether support for previous exchange version will cease, and the expectations of exchange implementers.
[XD2-5]	The IPT MUST notify exchange stakeholders of significant changes and contact information upon release of the new or modified exchange components.
<b>Justification</b>	
<p>Communication between exchange developers and Network governance before and during the development of new or modified exchange components provides significant benefit for both parties.</p> <p>Communication with existing exchange implementers will help ensure a smooth transition and help convey a clear understanding of the implications on existing implementations.</p>	

### 2.2.2. Publishing an Exchange

The Network has enacted a procedure for reviewing, approving, and publishing new or upgraded exchange components. While the specifics of the process are documented separately, the core aspects of the publishing process are described here in the context of the rules and guidelines that either should or must be followed by exchange developers.

Any change to a published exchange must undergo a review process by the appropriate Network governance group. Subject to this process are all new exchanges and any modification to an exchange no matter how slight. All changes to existing exchanges must adhere to the exchange component versioning approach described in Section 2.3.

When a new or modified exchange is ready for review, it must be bundled into a single Exchange Documentation Package (package) and submitted to Network governance. The package must consist of the following components:

- *XML Schema* - The formal definition of the structure and format of the data.

- *Schema Conformance Report* - The document describing the findings resulting from a comparison of the final draft schema against the guidelines for schema design prepared by the Network governance.
- *Data Exchange Template* - The template outlining each data element within the schema along with definitions, validation rules, and example content. This is a more human readable version of the XML schema.
- *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.
- *Valid XML instance file (one or more)* - A sample XML file using the schema developed for the exchange.

Additional information on documentation requirements can be found in Section 2.4 - Exchange Documentation. Exchange developers should visit the Exchange Network website to obtain additional information on the exchange conformance review process.

### **Publishing Exchange Document Revisions**

In cases when only the documentation for an exchange is updated and the modification is classified as a “revision” as described in the exchange component versioning approach, the package may only include the updated FCD and any other revised document.

### **Schema User’s Guide**

The package may include a Schema User’s Guide, although it is not required. This guide can be very useful in situations when the schema maps to an existing paper form, for example.

### **Trading Partner Agreement**

A Trading Partner Agreement (TPA) defines in writing, for specific data exchanges, the partners’ individual and joint responsibilities in stewardship, security, and other items essential for the effective exchange of information between two or more trading partners on the Exchange Network. A TPA may be required if existing documents do not sufficiently define the responsibilities and operational expectations between exchange partners. Specific TPA requirements are defined in the Network policy framework which can be found on the Exchange Network website<sup>3</sup>.

---

<sup>3</sup> See <http://www.exchangenetwork.net/policy/index.htm> for more information.

*Exchange Documentation – Publishing an Exchange*

<b>Rules and Guidelines</b>	
<b>Rule</b>	<b>Description</b>
[XD2-6]	All exchange packages submitted for review by Exchange Network governance <b>MUST</b> include the XML Schema, Schema Conformance Report, Data Exchange Template, Flow Configuration Document, and one or more valid example XML instance files.
[XD2-7]	Exchange developers <b>MAY</b> include a Schema User's Guide in the Exchange Documentation Package.
[XD2-8]	A Trading Partner Agreement <b>MUST</b> be included in the exchange documentation package if required.
[XD2-9]	Upgraded exchanges <b>MUST</b> include schema change control documentation in the exchange documentation package, describing the differences between the current and previous version of the XML schema.
[XD2-10]	All developers <b>MUST</b> document the reasons for both their use and rejection of SSC in the schema conformance report.
[XD2-11]	The exchange flow schema and documentation <b>MUST</b> be approved and posted to the Exchange Network website before the production exchange is implemented.
[XD2-12]	Exchanges <b>MUST</b> be tested by at least two independent implementations prior to publishing.
[XD2-13]	Copies of all published exchange components <b>MUST</b> be accessible from the Exchange Network web site.
<b>Justification</b>	
Rules and guidelines relating to the publishing of Exchange documentation and components have been formally established by Network governance to ensure that implementers have access to complete and consistent exchange components.	

## 2.3. Exchange Component Versioning

### 2.3.1. General Versioning Principles

The Exchange Component Versioning principles have been enacted to ensure that there is strict delineation between releases of a given exchange component. The versioning principles described herein apply specifically to schema and documents associated with an exchange. Any deliverable associated with an exchange is referred to as a “component” in this section.

There are three types of changes that can be made to any component; **major**, **minor**, or **revision**. The version of a component is always represented as a suffix to the component name using the following format:

“\_v{MajorVersionNumber}”.{MinorVersionNumber}[{RevisionIdentifier}]

For example, a component suffixed with “\_v2.3a” represents major version 2, minor version 3, revision “a”. Revision identifiers are optional in component names.

## **Major Changes, Minor Changes, and Revisions**

### Major Changes

Major version changes denote a significant upgrade from the previous iteration of the resource. Major version upgrades occur when significant new features are introduced, or there is a change in technology or business process that renders the requirements of a resource incompatible with previous versions. Major changes should typically be driven by a business process upgrade or improvement. Major changes are almost always accompanied by a revised XML schema.

### Minor Changes

Minor version changes denote an upgrade that includes one or more additional features or enhancements, but does not fundamentally alter the design or function of the previous version. Minor version changes on the EN must be “backwards compatible” meaning the change is technically interoperable with items developed using the same major version number.

Minor changes are almost always accompanied by a revised XML schema. In order for a schema change to qualify as a minor revision it must be backward compatible with the previous minor version. For this to be true, the revised schema may only add new optional elements, make required elements optional, or loosen restrictions (such as relaxing or removing length or range limitations or add a choice compositor to an existing element).

### Revisions

Revisions only apply to documents, not schema. Revisions may be used for the purpose of fixing mistakes, clarifying language, or adding additional information. Revisions may also be used to add new data services to an exchange. Revisions most often apply to FCDs, although it is possible to use a revision letter on a DET, schema user’s guide or other supporting document for the exchange. Document revisions may be sent to the Exchange Network website coordinator for publishing without a formal review.

## Version Alignment of Exchange Components

All exchange components must share a common major and minor version number. This means that a schema update (major or minor) must also be accompanied by an update to the DET and FCD. These procedures are discussed in further detail below for each component type.

### Versioning Rules by Component Type

Special versioning rules apply based on the type of component being modified.

#### Schema

The specific versioning requirements for schema are listed in the XML Design Rules and Conventions for the Exchange Network. As such, this section specifically addresses the implications of a schema revision on other exchange components. As mentioned previously, a schema update (major or minor) must also be accompanied by an update to the DET and FCD. The DET, serving primarily as a data dictionary for the schema, must be updated to reflect the schema changes. The FCD may or may not require updates to the narrative, but at a minimum, the FCD must record the schema update in the *Component Alignment and Version History* section.

Updated schema are also required to include an updated sample XML instance file as well as a change control log, describing the specific differences between the current and revised schema.

#### FCD

The Flow Configuration Document (FCD) is the central information resource for an exchange. Users should be able to identify the current version of all exchange components by reading the current version of the FCD. The FCD contains the change history for all exchange components in its initial pages, and information about the data exchange including a list of current and supported schema versions and Data Services. Any time any exchange component is changed the FCD must also be updated to reflect and document those changes. As the document of record for an exchange, users must always be able to refer to the change history tables in the FCD to determine the version of schema, DET, and other documentation.

The FCD revision identifier must be incremented if an exchange adds a new data service or if there is a change to business rules. If the change to the FCD could cause existing implementations to fail based on the previous version of the FCD, then a major version upgrade to the FCD and the schema is required<sup>4</sup>. If the FCD change does not cause any

---

<sup>4</sup> The requirement of creating and releasing a new version of a schema that may have no changes from the previous version may seem overly burdensome, but it is necessary to adhere to the overarching Network

disruption for existing flow implementations (e.g. necessitating additional coding) and does not require any changes to the schema (e.g., relaxing flow business rules, or adding data services) then only a revision letter increment is required in the FCD. In situations where it is unclear what versioning increment a development change should trigger, the decision should be made by the IPT with the understanding that versions of exchange components should always contain the same major and minor version number.

## DET

The Data Exchange Template contains a listing of schema elements and additional information about their origin, use, and may include business rules. Since the DET serves as a data element dictionary for the schema, the major and minor version of the DET must align with the schema it describes. In the event of a correction or business rule change, the DET only requires an incremented revision identifier on the DET file name.

## **Versioning of Draft Components**

It is common for many versions of a draft document or schema to circulate during the development of a new or upgraded exchange. To ensure that versions of draft components are easily identified and tracked, the use of a special revision identifier is required in the component name. This provides a mechanism for indicating that a resource in development has changed, without incrementing the major or minor version number. The format for a draft revision is

“\_v”{MajorVersionNumber}”.”{MinorVersionNumber}”draft”{DraftNumber}

For example, a draft schema might be named **ABC\_shared\_v1.0draft3.xsd**.

### *Exchange Component Versioning – General Versioning Principles*

<b>Rules and Guidelines</b>	
<b>Rule</b>	<b>Description</b>
[XD3-1]	A major version increment <b>MUST</b> be performed when a change to any component renders the exchange incompatible with previous versions.
[XD3-2]	A minor version increment <b>MUST</b> be performed when a change to any component is backwards compatible with the previous version of the exchange.
[XD3-3]	A revision increment <b>MUST</b> be performed when changing exchange documentation for the purpose of fixing mistakes, clarifying language, or adding additional information where its related schema does not change.

versioning strategy. This ensures version interoperability and version alignment between all exchange components.

[XD3-4]	All exchange components <b>MUST</b> share the same major and minor version number. Any time any component is changed, the Flow Configuration Document <b>MUST</b> also be updated to reflect and document those changes.
[XD3-5]	Exchange document filenames <b>MUST</b> be formatted as: {ExchangeIdentifier}"_{DocumentType}"_v{MajorVersionNumber}."_{MinorVersionNumber}[_{Revision}][_draft"{DraftNumber}].
[XD3-6]	A change to an exchange component that has not been published as final <b>MUST</b> include "Draft" in the component name.
[XD3-7]	Exchange documents in a draft state <b>SHOULD</b> be clearly labeled as a "DRAFT" on the title page and throughout the document with a watermark.
[XD3-8]	All draft indications <b>MUST</b> be removed from exchange components before the resource can be considered "final" and officially released for use on the Network.
<b>Justification</b>	
Versioning rules maintain clear delineation between all releases of a given exchange component and ensure compatibility across exchange implementations.	
Versioning rules for draft documents help mitigate the risk that a draft document is thought to be final by a reader. These rules also provide a simple, intuitive way of denoting draft status and tracking revisions to draft components.	

### 2.3.2. Versioning Data Services

Data Services are the operations supported by an exchange, typically initiated through the invocation of a Query, Solicit, Submit or Execute primitive command as defined in the exchange FCD. It is important that when a change is made to a data service that the change is implemented in a manner that will not break existing implementations.

Examples of data service changes include:

- Changing the parameters associated with a Query or Solicit command
- Changing the return schema associated with a Query or Solicit command
- Changing the processing logic associated with a Submit or Execute command

To ensure that changes to data services do not affect existing implementations, any time a data service undergoes a change, the data service must be given a unique name. Under no circumstances should the name of the updated data service remain unchanged.

The data service name for a Query or Solicit command is the value provided in the "request" parameter. The data service name for a Submit command is the value provided for the "flowOperation" parameter. Lastly, the data service name for an Execute command is the unique combination of "interfaceName" and "methodName".

The data service versioning guidelines are integral to the data service naming conventions for Query and Solicit Services. Please review Section 2.5.1 for more information.

*Exchange Component Versioning –Versioning Data Services*

<b>Rules and Guidelines</b>	
<b>Rule</b>	<b>Description</b>
[XD3-9]	A data service definition SHOULD NOT change. In the case where an existing data service needs to be modified, a new data service SHOULD be created with a unique name.
<b>Justification</b>	
Ensuring that unique versions of data services are given a unique name helps to ensure that existing implementations will not break when new versions of services are implemented for an exchange.	

### 2.3.3. Upgrading Existing Exchange Components

When an exchange undergoes an upgrade, there are often two competing considerations. First, there is the desire to implement potentially significant changes that increase the usefulness and versatility of the exchange. Competing with this objective is the desire to minimize the effort required by existing exchange implementers to transition to the upgraded exchange. If the exchange upgrade is too conservative, there is a risk of a missed opportunity to expand the exchange in new and useful ways. However, if changes are too radical, the cost and effort required by existing exchange implementers to upgrade may compel some to abandon the exchange.

Each situation is unique, and therefore there cannot be universal guidance on what exchange developers should do in this situation. It is recommended that the exchange designers find the appropriate balance based on the specific situation.

When an exchange is upgraded, it is an opportunity for the exchange components to be revised to meet the latest Network rules and guidelines. The Exchange Design Rules allow for discretion on the part of the exchange designers to choose what upgrades to pursue, however, if the exchange is undergoing a significant redesign, all exchange components must be upgraded to meet the latest standards. “Significant redesign” may or may not correspond to a major version upgrade as defined in the Exchange Component Versioning principles. It is the discretion of the exchange designer as to whether the upgrade constitutes a significant redesign.

In most instances, updating exchange components to meet current rules should not require changing any functionality of the exchange. Most often, it is an exercise in using the latest document templates and ensuring that the documentation is more thorough and

complete. Schema design rules must be reviewed to ensure compliance with the latest standards as well.

*Exchange Component Versioning – Upgrading Existing Exchange Components*

<b>Rules and Guidelines</b>	
<b>Rule</b>	<b>Description</b>
[XD3-10]	Developers SHOULD strive for co-existence and “grandfathering” of existing versions of flow components when transitioning from one version to another.
[XD3-11]	Exchange components SHOULD be made compliant with current Exchange Network standards as part of major or minor version changes.
[XD3-12]	When an exchange undergoes a significant redesign, all exchange components MUST be upgraded to comply with the latest Network schema and exchange design rules.
<b>Justification</b>	
Requiring that significant exchange upgrades incorporate the latest templates and design rules will help ensure the Network continues to evolve and that exchanges are designed consistently.	

## 2.4. Exchange Documentation

This section discusses rules and guidelines for developing exchange documentation such as the FCD and DET.

### 2.4.1. The Flow Configuration Document (FCD)

The FCD is the primary information source about an exchange. It lists contact information, describes the history and purpose of the exchange, and most importantly, it describes the specific services and operations performed by partners in the exchange. Because the FCD contains a revision history for all exchange components, it serves as the document that binds all other exchange components together (including the schema, DET, and schema user’s guide).

Network governance has developed a FCD Template that must be used for all new exchanges and all existing exchanges that undergo a significant redesign. Using the FCD template increases the likelihood that the requisite content will be included in the exchange FCD. While the FCD Template should be adequate for many exchanges in its existing form, exchange designers are permitted to alter the FCD Template structure and content to more clearly convey the specifics of the exchange, so long as the requisite portions are present.

Exchange designers are encouraged to examine existing FCDs for structural examples. Designers should contact Network governance for a list of reference FCDs that are

considered to be of good quality. Network governance may also be able to suggest an FCD that has similarities to the exchange being designed.

FCDs should clearly describe the overall exchange architecture. This includes the roles of the different partners in the exchange. For example, there may be “data providers” (senders) and “data consumers” (receivers). EPA is a data consumer for many exchanges, for example. The FCD should describe the services that must be offered by each partner, based on their role in the exchange. For example, a partner fulfilling a “sender” role should not be required to receive and process submissions. Furthermore, the FCD should indicate which services are required or optional for partners to implement, specific to the role the partner plays in the exchange.

Please also see section 2.5 regarding Query and Solicit Services since the FCD is required to contain this information.

*Exchange Documentation – Flow Configuration Document (FCD)*

<b>Rules and Guidelines</b>	
<b>Rule</b>	<b>Description</b>
[XD4-1]	The Flow Configuration Document <b>MUST</b> be based on the most recent Exchange Network-approved Flow Configuration Document Template.
[XD4-2]	The exchange Flow Configuration Document <b>SHOULD</b> include a flowchart of processing steps, if applicable.
[XD4-3]	The exchange Flow Configuration Document or Schema User's Guide <b>SHOULD</b> include a high-level schema diagram outlining each major data block for each root schema associated with the exchange.
[XD4-4]	The exchange Flow Configuration Document <b>SHOULD</b> document unresolved issues or ideas for future enhancements to the exchange in an appendix.
[XD4-5]	The exchange Flow Configuration Document <b>MUST</b> specify the steps that must be taken for a new partner to implement and participate in the exchange.
[XD4-6]	The exchange Flow Configuration Document must indicate whether implementation of each service or operation is required or optional, based on the role of the partner in the exchange.
[XD4-7]	The exchange Flow Configuration Document <b>MUST</b> describe the specific meaning of all possible status responses returned for a given network operation.
<b>Justification</b>	
FCD rules and guidelines help ensure consistent and thorough exchange documentation.	

## 2.4.2. The Data Exchange Template (DET)

The Data Exchange Template contains a listing of schema elements and additional information about their origin and use and may include a listing of business rules. It also can provide a mapping to the database fields in a specific receiving system. Implementers also can use the DET to document mapping from a source database to the schema.

Exchanges that target a specific receiving system are often subject to a number of business rules. Since schema cannot enforce cross-element validation, the business rules must be documented separately. The DET is the preferred document to describe the business rules since the rules can be listed alongside the element(s) that implement the rule.

In some cases, the business rules are already documented separately. For example, an exchange that targets a specific system may already list the business rules in an existing document. In this situation, a reference or link should be provided in the DET leading the user to the location where the business rules can be located.

Some exchanges, such as those designed for data publishing or peer-to-peer exchanges may not have business rules. In this case, it is acceptable to not include any business rules in the DET.

### *Exchange Documentation – The Data Exchange Template (DET)*

Rules and Guidelines	
Rule	Description
[XD4-8]	Exchange developers SHOULD document or provide a reference to applicable data processing rules within the Data Exchange Template.
Justification	
The DET provides a consistent and practical location to document business rules, where applicable.	

## 2.5. Query and Solicit Services

Data Services are the operations supported by an exchange, typically initiated through the invocation of a Query, Solicit, Submit or Execute primitive command as defined in the exchange FCD. This section specifically describes the rules and guidelines for Query and Solicit data services. Query and Solicit services may take parameters, allowing the user to customize the request with user-supplied criteria.

### 2.5.1. Service Names

The Network recommends that Query and Solicit data services use the following naming convention:

**{MethodName}{ObjectName}["By"{ParameterName(s)}]"\_v"{SchemaVersionReturned}**

Note that each word in the service name should be capitalized, commonly referred to as “upper camel case” in programming parlance. Each component of the data service name is described in detail below:

Component Name	Description	Example
Method Name	The verb describing the type of action to perform.	Get, Set
Object Name	The principle entity or module affected by the operation. May contain multiple words.	Facility, Permit, Policy
Parameter Name(s)	Parameter(s) describe the constraint by which the Service is executed. It is optional and can be used at the discretion of the Data Service creator. Do not include this component if the service does not take any parameters.	<u>Fixed parameter services:</u> ByName, BySIC, ByZipAndCounty <u>Dynamic-parameter services:</u> ByParameters
Schema Version Returned	The version of the schema to which the query result file conforms. The version must include the major and minor version number.	v2.1

Examples of a valid data service names are as follows:

- GetManifestList\_v1.0
- GetPermitByPermitNumber\_v1.0
- GetFacilityByNameAndZipCode\_v2.3
- GetBeachByParameters\_v3.0

The Parameter Name(s) component should be explicitly named if there are only one or two parameters for the data service. If a data service accepts more than two parameters, the Data Service name can use “ByParameters” for this component.

The schema version specified in the data service name ties it to the version of schema to which the response document conforms. As a result, publishing a new major or minor version of the schema will require a change to the “Schema Version Returned portion of the Service name. However, if an existing Data Service is changed (by, for example, adding a parameter), a new Data Service must be created with a new name.

*Query and Solicit Services – Service Names*

Rules and Guidelines	
Rule	Description
[XD5-1]	Data Service names SHOULD be formatted as: {Method}{Object}["By"{ParameterName(s)}] "_v"{SchemaVersionReturned}
[XD5-2]	Data Service names MUST be in upper camel case and MUST not include spaces.
Justification	
A common approach to creating data service names provides consistency and predictability for exchange implementers.	

**2.5.2. Service Parameters**

Parameters provide a means of passing criteria to a Query or Solicit data service. Most often, data service parameters are used to filter a result set for only the records that the requester is interested in obtaining. Rules regarding the formulation of parameters are important to ensure that services are implemented consistently across nodes.

**XML Documents as Service Parameters**

While most services implement a list of specific, named parameters, the Node 2.0 Specification supports using a custom XML document to specify criteria in a service request. XML offers the advantage of creating a highly flexible way of providing input, but it suffers one significant disadvantage; it prevents generic node clients from dynamically binding to the data service. Use of XML parameters requires that the requester have foreknowledge of the parameter structure, likely only possible through hard-coded logic in a custom Network node client application.

One additional disadvantage is that significantly more work is required to document the XML input structure and how it must be implemented. The XML schema must also be included with the Exchange Documentation Package when submitting for review and approval by Network governance.

**Parameter Documentation for Node v1.1-compatible Exchanges**

Query and Solicit parameter declaration changed substantially between Node Specification v1.1 and v2.0. Node v1.1 simply passed parameters as an array of string values, whereas Node v2.0 now requires that each parameter name and value be declared explicitly, along with optionally listing data type and encoding type. This change requires that detailed parameter documentation be provided for exchanges that support both v1.1 and v2.0 endpoints. For v1.1-compatible exchanges, the ordinal position of each

parameter must be stated. Rules also require that node v1.1 exchanges only allow the pipe symbol (“|”) to be used to separate multiple values provided for a single parameter in a request.

### *Query and Solicit Services – Service Parameters*

<b>Rules and Guidelines</b>	
<b>Rule</b>	<b>Description</b>
[XD5-3]	XML documents SHOULD NOT be embedded into Query and Solicit parameters.
[XD5-4]	If a data service accepts an XML document as a query parameter, the XML schema MUST be included in the Exchange Documentation Package and MUST be fully documented in the Flow Configuration Document.
[XD5-5]	Each parameter specified in a data service request MUST be treated as a logical AND condition while multiple values provided for the same parameter MUST be treated as an OR condition.
[XD5-6]	Exchanges that support the Node v1.1 Specification MUST only allow a pipe symbol (“ ”) as a delimiter between multiple values for a given data service parameter.
[XD5-7]	The exchange Flow Configuration Documents MUST indicate the ordinal position of each parameter in the parameter array if the exchange is compatible with the Node v1.1 Specification.
<b>Justification</b>	
XML-schema based parameters are discouraged because they do not allow for dynamic binding of node client software and increase complexity of an exchange.	
Parameter documentation rules help ensure that services can be implemented consistently on both Node v1.1 and v2.0 implementations.	

### **2.5.3. Service Documentation**

Data services must be documented in the exchange FCD. The data service documentation must be thorough, ensuring that an exchange implementer or an exchange network client can invoke a data service with confidence on how the operation will be interpreted and processed by all parties in the exchange.

The following aspects of each data service must be documented:

- Whether the service must be supported by Query, Solicit, or both
- Whether the service must be supported by all implementers of the exchange or only nodes that fulfill a certain role in the exchange (such as a data providers or data receivers/processors)

- The message schema name and version to which a returned XML document conforms.
- The element in the return schema that constitutes a “row” for the purposes of responding to `rowId` and `maxRows` in the Query request.

Each parameter supported by the data service must document the following:

- The parameter name (without spaces)
- The parameter description
- The allowed occurrence of the parameter (0-1, 1, 1-*n*, or 0-*n*)
- Formatting requirements for dates or other types passed as strings
- Wildcards:
  - Whether wildcards are supported for the parameter or whether wildcard behavior is implemented for the parameter by default.
  - If user-supplied wildcards are supported, indicate the wildcard character.

*Query and Solicit Services –Service Documentation*

Rules and Guidelines	
Rule	Description
[XD5-8]	The Flow Configuration Document MUST fully describe each data service including parameter names, data types, allowable occurrence, wildcard behavior, and return schema.
[XD5-9]	The Flow Configuration Document MUST specify which XML element is defined as a "row" for each data service in an exchange.
Justification	
Thorough data service documentation is required if all nodes are to implement services consistently and reliably.	

## 2.6. Exchange Network Header

The Exchange Network Header (Header) is an XML schema that serves as a wrapper to one or more XML instance documents. The Header is also commonly referred to as the Exchange Network Document in some Network documentation.

The purpose of the Header is to provide metadata about a submission or “payload”. The payload is always an instance document conforming to a message schema defined by an exchange. The Header describes the author, time and date of file creation, and other information.

The Header is primarily used in two scenarios:

1. To carry processing instructions for submission of data to a receiving system, or
2. To describe the request parameters used to produce the payload result file from a Network Query or Solicit event.

A useful aspect of the Header is that it includes an array of user-defined name/value pairs. Exchanges have historically used the name/value pairs in a variety of ways such as:

- specifying notification email addresses
- providing back-end submission processing instructions
- providing a copy of the query criteria used to return the attached payload

In the following example, a sample XML instance document that conforms to the Header v2.0 is shown:

```
<hdr:Document xmlns:hdr="http://www.exchangenetwork.net/schema/header/2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="ID02">
  <hdr:Header>
    <hdr:AuthorName>John Smith</hdr:AuthorName>
    <hdr:OrganizationName>Environmental Quality Group, Ltd.</hdr:OrganizationName>
    <hdr:DocumentTitle>Emission Submission 2006</hdr:DocumentTitle>
    <hdr:CreationDateTime>2006-04-05T09:30:47-05:00</hdr:CreationDateTime>
    <hdr:Keywords>Emission, Clean Air</hdr:Keywords>
    <hdr:Comment>This is a comment</hdr:Comment>
    <hdr:DataServiceName>GetSubmissionDataByFlow</hdr:DataServiceName>
    <hdr:SenderContact>PO Box 431112, Washington, DC 20001</hdr:SenderContact>
    <hdr:SenderAddress>mailto:jsmith@example.com</hdr:SenderAddress>
    <hdr:Property>
      <hdr:PropertyName>GeographicCoverageState</hdr:PropertyName>
      <hdr:PropertyValue>01</hdr:PropertyValue>
    </hdr:Property>
    <hdr:Property>
      <hdr:PropertyName>FiscalQuarter</hdr:PropertyName>
      <hdr:PropertyValue>2</hdr:PropertyValue>
    </hdr:Property>
  </hdr:Header>
  <hdr:Payload operation="Insert/Update">
    <!--information removed for example purposes-->
  </hdr:Payload>
</hdr:Document>
```

*Exchange Network Header - General*

<b>Rules and Guidelines</b>	
<b>Rule</b>	<b>Description</b>
[XD6-1]	The Exchange Network Header MAY be implemented on Query or Solicit responses.
[XD6-2]	Implementation of the Exchange Network Header MAY vary among Query and Solicit data services within an exchange.
[XD6-3]	The Exchange Network Header MUST be implemented on all Submit operations.
[SD5-P]	Schema developers MAY implement the EN Header as a method of adding metadata to one or more schema payloads.
[SD5-Q]	The Header MAY be used to include message metadata (sender identification, transaction type such as INSERT or DELETE, and other message-level parameters) or to bundle multiple XML messages into a single XML document.
<b>Justification</b>	
The Header provides a simple, common mechanism for describing the contents of a payload or processing instructions. The Header fulfills a common need among exchanges for documenting payload metadata and for describing payload processing.	

**2.6.1. Header Payload Element**

If the Header is used to wrap a submission file for processing by a specific receiving system, the Payload element provides a way of specifying processing commands outside of the message schema itself. By separating data from other processing information, the schema can be designed to represent data in a situation-agnostic fashion, thereby broadening its potential usefulness to scenarios beyond the exchange for which it was originally designed.

One way to use the Header is to separate transaction types into different payloads within the same Header, each indicating a different transaction type (such as “Insert/Update” and “Delete”). For example, the first payload may represent inserts and updates while the second payload represents data to be deleted in the receiving system. In this case, the Payload element’s operation attribute may be used to indicate the operation to be performed.

In the following example, a sample XML instance document that uses the Header v2.0 is shown with two different payloads, each with a different operation:

```

<hdr:Document xmlns:hdr="http://www.exchangenetwork.net/schema/header/2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="ID02">
  <hdr:Header>
    <!--information removed for example purposes-->
  </hdr:Header>

  <hdr:Payload operation="Insert/Update">
    <!--information removed for example purposes-->
  </hdr:Payload>

  <hdr:Payload operation="Delete">
    <!--information removed for example purposes-->
  </hdr:Payload>
</hdr:Document>

```

The operation attribute is optional and therefore may be omitted.

### *Exchange Network Header - Header Payload Element*

Rules and Guidelines	
Rule	Description
[XD6-4]	If an exchange supports multiple payloads within an Exchange Network Header, the <Payload> ID attribute SHOULD contain a unique identifier for each payload.
[XD6-5]	The first and only child element beneath the Exchange Network Header's <Payload> element SHOULD be the root element of the XML instance file being transmitted.
[XD6-6]	Allowable values for The Exchange Network Header <Payload> operation attribute MUST be documented in the exchange Flow Configuration Document along with a precise description of how each operation affects payload processing.
[XD6-7]	If the Header is implemented on the response to a Query or Solicit, the payload operation attribute SHOULD contain either "Query" or "Solicit" to indicate the type of operation that was performed to retrieve the result.
Justification	
The Header fulfills a common need among exchanges for documenting payload metadata and for describing payload processing.	

## 2.6.2. Header Documentation

Since the processing logic associated with an exchange must be definitive and precise, it is important that header implementation details are fully documented in the exchange Flow Configuration Document. When processing instructions are documented clearly, there is little room for ambiguity in an exchange. Flow developers are encouraged to carefully document Header implementation details.

### *Header Documentation*

<b>Rules and Guidelines</b>	
<b>Rule</b>	<b>Description</b>
[XD6-8]	Data Service-specific implementation details of the Exchange Network Header MUST be documented in the exchange Flow Configuration Document.
[XD6-9]	The exchange Flow Configuration Document MUST indicate which Exchange Network Header elements are required and how data included in the Header is used to determine document processing logic.
[XD6-10]	The exchange Flow Configuration Document MUST specify whether multiple payloads are supported and how they must be structured (within the Exchange Network Header, compressed attachments, or as an array of documents)
<b>Justification</b>	
Thorough documentation minimizes the likelihood of incurring additional costs that may be caused by misinterpretation of expected exchange behavior or ambiguity in exchange documentation.	

#### **2.6.3. Header Encryption and Security**

The Exchange Network Header specification supports document security using XML signature and XML encryption. This section discusses how encryption and signature should be applied to the header and payload.

If encryption is required for a data exchange, as documented in the FCD, only the payload should be encrypted, i.e., the first child of the payload element should be encrypted using the applicable receiver's public key. The header part of the document must not be encrypted as it does not contain sensitive information.

When only one signature is required, the entire document, including header and payload, should be signed using the enveloped signature defined in the XML signature specification. This ensures the integrity and non-repudiation of the whole document. The signature should be inserted in the document header. The hash of an enveloped signature is calculated using the entire document excluding the signature element. Therefore, insertion of the signature in the header will not change the hash value and thus will not invalidate the signature.

The following example demonstrates how to use XML encryption and XML signature in the Header. Note that the payload is encrypted first and the entire document, including the header, is signed using the sender's private key afterwards.

```

<hdr:Document xmlns:hdr="http://www.exchangenetwork.net/schema/header/2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="ID02">
  <hdr:Header>
    <!--information removed for example purposes-->
    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#" Id="_35df5cb9-f033-4533-bf4f-
b3a36ae892d6">
      <SignedInfo>
        <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
        <Reference URI="">
          <Transforms>
            <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
          </Transforms>
          <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <DigestValue>XIFT/GUkOlxrdhfvWsgY+XacjA=</DigestValue>
        </Reference>
      </SignedInfo>
      <SignatureValue>
        gwkMbTxke2iZov/FE+7FjiSfNF0ABdJluN2+6fladWtv0i0JAOvWKDaE0zDnV0kO
        ltr5zZRG6ME4XFPqhaY5ZUSjarv2Fmjy6l1BBW35fuS4OX2RHGqVzblhfQJKxvh+
        wKqdcqr9ZxaH5SiPVRpOoPPzssvVBOd36to2vboT5Y=
      </SignatureValue>
      <KeyInfo>
        <KeyName>US, DC, Washington, Exchange Network, Security Service, Yunhao
        Zhang</KeyName>
        <KeyValue>
          <RSAKeyValue>
            <Modulus>
              qZr11ilbLPwhBVe4OpiZRzj/8FwasGppEjBymlnQApMr9ZsdRR9VrpZ7m4+iMPB
              WTC24bOaUefA7QygGXDM4WIEKK6YQGGDNRqilFxlYRIPCzPSaT+utfrfFEpiFrQ
              9M6HYa1bjb/uOOTo/UXQ9hzpOrnXgSLMpmuM6B0tikuc=
            </Modulus>
            <Exponent>Aw==</Exponent>
          </RSAKeyValue>
        </KeyValue>
      </KeyInfo>
    </Signature>
  </hdr:Header>
  <hdr:Payload operation="Original" id="MyPayload000134">
    <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
Type="http://www.w3.org/2001/04/xmlenc#Element">
      <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
          <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <KeyName>US, DC, Washington, Exchange Network, Security Service, Yunhao
            Zhang</KeyName>
          </KeyInfo>
        </EncryptedKey>
      </KeyInfo>
    </EncryptedData>
  </hdr:Payload>

```

```

tIIHsDqxM67AergCiM/RxgWbtEo5fOScQbN6wjes7qw=
</CipherValue>
</CipherData>
</EncryptedKey>
</KeyInfo>
<CipherData>
<CipherValue>
cmy7z0vnl3P3MokHsBD2M6U7iuKDNR0pqLUOgxMrF6QaoJ9iGA8O3PHTCp1ZQvss
T97RfageaPuqAh36rvUSyMXzOIxxzOEH/AG2IcTs61P3MpWDmA74W0I0uV+ZH1uf
svx2sp0sBVY1ZZMEI5rSETnJk+RDAIPADpcbCg==
</CipherValue>
</CipherData>
</EncryptedData>
</hdr:Payload>
</hdr:Document>

```

### *Exchange Network Header - Header Encryption and Security*

Rules and Guidelines	
Rule	Description
[XD6-11]	If encryption is required for an exchange, the Exchange Network Header <b>MUST NOT</b> be encrypted, although the payload <b>SHOULD</b> be.
[XD6-12]	For digitally signed submissions, the entire document including the Exchange Network Header and payload <b>SHOULD</b> be signed and inserted into the Header's <Signature> element. The hash <b>SHOULD</b> be calculated excluding the signature element to avoid invalidating the signature.
Justification	
The encryption and signature capabilities of the Header provide a mechanism for securing exchange payloads. Providing these capabilities in a standardized Network schema reduce the effort needed to develop a separate document encryption or signature schema independently.	

#### 2.6.4. Header Versions

The Header is updated periodically. New versions build upon implementation experience of previous versions. Two versions of the Header are available:

Version	Location
v0.9	<a href="http://www.exchangenetwork.net/schema/v1.0/ExchangeNetworkDocument.xsd">http://www.exchangenetwork.net/schema/v1.0/ExchangeNetworkDocument.xsd</a>
v2.0	<a href="http://www.exchangenetwork.net/schema/header/2/header_v2.0.xsd">http://www.exchangenetwork.net/schema/header/2/header_v2.0.xsd</a>

New exchanges are required to implement the latest version of the Header while existing, upgraded exchanges are encouraged to continue supporting the original version of the header, if implemented.

Both versions of the Header are similar. The table below contrasts the two versions:

Header v0.9	Header v2.0	Occurs	Element Description
<b>Document\Header</b>		<b>1</b>	
Author	AuthorName	1	Originator of the document. This should be the name of a person or a network node ID if the document is automatically generated.
Organization	OrganizationName	1	The organization to which the author belongs. It may be a state name, an organization name or a company name. For submissions to the CDX node, this should be the name of the organization.
Title	DocumentTitle	1	Title of the document.
CreationTime	CreationDateTime	1	This is a timestamp that marks when the document, including payloads and header part, was created.
	Keywords	0-1	Words that best describe the payload. Multiple keywords should be separated by commas. This is for transaction categorization and searching.
Comment	Comment	0-1	Additional comments for processors.
	DataFlowName	0-1	The name of the data flow associated with the payload. It could be the name of the data source for Query results.
DataService	DataServiceName	0-1	Name of a data service that generated the document. This is the name of the procedure that was used to initiate the creation of the payload. This would apply only for Query and Solicit and would not be applicable for Download and Submit.
ContactInfo	SenderContact	0-1	The sender's additional contact information. It could contain sender's electronic address and/or telephone numbers where the author can be reached.
Sensitivity		0-1	Level of document sensitivity.
	ApplicationUserIdentifier	0-1	The user ID for the backend system if it is different from the NAAS user ID.
Notification	SenderAddress	0-n	A well-formed URI where result/report can be sent. Currently the Network will make use of the Notification mechanism at the Document Level as described in the Protocol and Specification. Note that this could contain multiple addresses, including that of the submitter and/or other technical people related to contents of the payload.
	Signature	0-1	An XML signature associated with the document (Use <a href="http://www.w3.org/2000/09/xmldsig#">http://www.w3.org/2000/09/xmldsig#</a> )

<b>Document\Header\Property</b>		<b>0-n</b>	Other properties of the document (use name value pairs). This is an extension mechanism to cover any other elements that are not defined in the specification.
name	PropertyName	1	An identifier for an exchange-specific property
value	Property\Value	1	The supplied value for the exchange-specific property.
<b>Document\Property</b>			
Operation (attribute)	operation (attribute)		Operation to be performed

*Exchange Network Header - Header Versions*

<b>Rules and Guidelines</b>	
<b>Rule</b>	<b>Description</b>
[XD6-13]	New exchanges <b>MUST</b> implement the latest approved Exchange Network Header.
[XD6-14]	Upgraded exchanges <b>SHOULD</b> continue to support the Exchange Network Header version used in the existing version of the exchange.
[XD6-15]	When an exchange undergoes a significant redesign, the latest approved Exchange Network Header <b>SHOULD</b> be implemented.
<b>Justification</b>	
Requiring usage of the latest Header for new exchanges ensures that the latest advances in the header are employed. Maintaining the current header for upgraded exchanges maintains a higher degree of backward compatibility and requires less implementation work for partners.	

## Appendix A – Summary of Design Rules

This appendix summarizes the design rules found in this document. This appendix is intended as a quick reference for developers. For additional information on the feature or on the advantages, disadvantages, and justification for a particular rule, please see the corresponding section for the full commentary as noted by the rule prefix.

<b>Rules and Guidelines</b>	
<b>General Exchange Design</b>	
[XD1-1]	An exchange <b>MUST</b> be prescribed an exchange identifier in the form of a single term or acronym (e.g. TRI, WQX, FACID).
[XD1-2]	The exchange identifier <b>MUST</b> be used consistently throughout the schema and documentation, including the "dataflow" parameter on all Network primitive methods associated with the exchange, namespace prefixes, and schema file names.
[XD1-3]	The "dataflow" parameter <b>MUST</b> match the exchange identifier prescribed to the exchange and <b>MUST</b> be formatted as follows: {ExchangeIdentifier}"_v"{MajorVersionNumber}.
[XD1-4]	Query and Solicit services <b>SHOULD</b> be defined, even for exchanges that are primarily designed for data reporting purposes.
<b>Exchange Development and Publishing</b>	
[XD2-1]	IPTs <b>SHOULD</b> include members from all stakeholder groups.
[XD2-2]	IPTs <b>MUST</b> inform Network governance at the outset of a new exchange development project.
[XD2-3]	Exchange developers <b>SHOULD</b> engage Exchange Network governance for review of schema and flow architecture design before beginning schema or software development.
[XD2-4]	After initial planning of an exchange upgrade, the IPT <b>SHOULD</b> publish a pre-change notification that outlines the planned changes, which partners will be impacted by the changes, whether support for previous exchange version will cease, and the expectations of exchange implementers.
[XD2-5]	The IPT <b>MUST</b> notify exchange stakeholders of significant changes and contact information upon release of the new or modified exchange components.
[XD2-6]	All exchange packages submitted for review by Exchange Network governance <b>MUST</b> include the XML Schema, Schema Conformance Report, Data Exchange Template, Flow Configuration Document, and one or more valid example XML instance files.

[XD2-7]	Exchange developers MAY include a Schema User's Guide in the Exchange Documentation Package.
[XD2-8]	A Trading Partner Agreement MUST be included in the exchange submission package if required.
[XD2-9]	Upgraded exchanges MUST include schema change control documentation in the exchange package, describing the differences between the current and previous version of the XML schema.
[XD2-10]	All developers MUST document the reasons for both their use and rejection of SSC in the schema conformance report.
[XD2-11]	The exchange flow schema and documentation MUST be approved and posted to the Exchange Network website before the production exchange is implemented.
[XD2-12]	Exchanges MUST be tested by at least two independent implementations prior to publishing.
[XD2-13]	Copies of all published exchange components MUST be accessible from the Exchange Network web site.
<b>Exchange Component Versioning</b>	
[XD3-1]	A major version increment MUST be performed when a change to any component renders the exchange incompatible with previous versions.
[XD3-2]	A minor version increment MUST be performed when a change to any component is backwards compatible with the previous version of the exchange.
[XD3-3]	A revision increment MUST be performed when changing exchange documentation for the purpose of fixing mistakes, clarifying language, or adding additional information where its related schema does not change.
[XD3-4]	All exchange components MUST share the same major and minor version number. Any time any component is changed, the Flow Configuration Document MUST also be updated to reflect and document those changes.
[XD3-5]	Exchange document filenames MUST be formatted as: {ExchangeIdentifier}_"_{DocumentType}"_v"{Major}"."{Minor}">{Revision}][_draft]{DraftNumber}].
[XD3-6]	A change to an exchange component that has not been published as final MUST include "Draft" in the component name.
[XD3-7]	Exchange documents in a draft state SHOULD be clearly labeled as a "DRAFT" on the title page and throughout the document with a watermark.
[XD3-8]	All draft indications MUST be removed from exchange components before the resource can be considered "final" and officially released for use on the Network.
[XD3-9]	A data service definition SHOULD NOT change. In the case where an existing data service needs to be modified, a new data service SHOULD be created with a

	unique name.
[XD3-10]	Developers SHOULD strive for co-existence and “grandfathering” of existing versions of flow components when transitioning from one version to another.
[XD3-11]	Exchange components SHOULD be made compliant with current Exchange Network standards as part of major or minor version changes.
[XD3-12]	When an exchange undergoes a significant redesign, all exchange components MUST be upgraded to comply with the latest Network schema and exchange design rules.
<b>Exchange Documentation</b>	
[XD4-1]	The Flow Configuration Document MUST be based on the most recent Exchange Network-approved Flow Configuration Document Template.
[XD4-2]	The exchange Flow Configuration Document SHOULD include a flowchart of processing steps, if applicable.
[XD4-3]	The exchange Flow Configuration Document or Schema User's Guide SHOULD include a high-level schema diagram outlining each major data block for each root schema associated with the exchange.
[XD4-4]	The exchange Flow Configuration Document SHOULD document unresolved issues or ideas for future enhancements to the exchange in an appendix.
[XD4-5]	The exchange Flow Configuration Document MUST specify the steps that must be taken for a new partner to implement and participate in the exchange.
[XD4-6]	The exchange Flow Configuration Document must indicate whether implementation of each service or operation is required or optional, based on the role of the partner in the exchange.
[XD4-7]	The exchange Flow Configuration Document MUST describe the specific meaning of all possible status responses returned for a given network operation.
[XD4-8]	Exchange developers SHOULD document or provide a reference to applicable data processing rules within the Data Exchange Template.
<b>Query and Solicit Services</b>	
[XD5-1]	Data Service names SHOULD be formatted as: {Method}{Object}["By"{ParameterName(s)}] "_v"{SchemaVersionReturned}
[XD5-2]	Data Service names MUST be in upper camel case and MUST not include spaces.
[XD5-3]	XML documents SHOULD NOT be embedded into Query and Solicit parameters.
[XD5-4]	If a data service accepts an XML document as a query parameter, the XML schema MUST be included in the Exchange Documentation Package and MUST be fully documented in the Flow Configuration Document.
[XD5-5]	Each parameter specified in a data service request MUST be treated as a logical

	AND condition while multiple values provided for the same parameter MUST be treated as an OR condition.
[XD5-6]	Exchanges that support the Node v1.1 Specification MUST only allow a pipe symbol (" ") as a delimiter between multiple values for a given data service parameter.
[XD5-7]	The exchange Flow Configuration Documents MUST indicate the ordinal position of each parameter in the parameter array if the exchange is compatible with the Node v1.1 Specification.
[XD5-8]	The exchange Flow Configuration Document MUST fully describe each data service including parameter names, data types, allowable occurrence, wildcard behavior, and return schema.
[XD5-9]	The Flow Configuration Document MUST specify which XML element is defined as a "row" for each data service in an exchange.
[SD5-P]	Schema developers MAY implement the EN Header as a method of adding metadata to one or more schema payloads.
[SD5-Q]	The Header MAY be used to include message metadata (sender identification, transaction type such as INSERT or DELETE, and other message-level parameters) or to bundle multiple XML messages into a single XML document.
<b>Exchange Network Header</b>	
[XD6-1]	The Exchange Network Header MUST be implemented on all Submit operations.
[XD6-2]	The Exchange Network Header MAY be implemented on Query or Solicit responses.
[XD6-3]	Implementation of the Exchange Network Header MAY vary among Query and Solicit data services within an exchange.
[XD6-4]	If an exchange supports multiple payloads within an Exchange Network Header, the ID attribute SHOULD contain a unique identifier for each payload.
[XD6-5]	The first and only child element beneath the Exchange Network Header's element SHOULD be the root element of the XML instance file being transmitted.
[XD6-6]	Allowable values for The Exchange Network Header operation attribute MUST be documented in the exchange Flow Configuration Document along with a precise description of how each operation affects payload processing.
[XD6-7]	If the Header is implemented on the response to a Query or Solicit, the payload operation attribute SHOULD contain either "Query" or "Solicit" to indicate the type of operation that was performed to retrieve the result.
[XD6-8]	Data Service-specific implementation details of the Exchange Network Header MUST be documented in the exchange Flow Configuration Document.
[XD6-9]	The exchange Flow Configuration Document MUST indicate which Exchange Network Header elements are required and how data included in the Header is

	used to determine document processing logic.
[XD6-10]	The exchange Flow Configuration Document <b>MUST</b> specify whether multiple payloads are supported and how they must be structured (within the Exchange Network Header, compressed attachments, or as an array of documents)
[XD6-11]	If encryption is required for an exchange, the Exchange Network Header <b>MUST NOT</b> be encrypted, although the payload <b>SHOULD</b> be.
[XD6-12]	For digitally signed submissions, the entire document including the Exchange Network Header and payload <b>SHOULD</b> be signed and inserted into the Header's element. The hash <b>SHOULD</b> be calculated excluding the signature element to avoid invalidating the signature.
[XD6-13]	New exchanges <b>MUST</b> implement the latest approved Exchange Network Header.
[XD6-14]	Upgraded exchanges <b>SHOULD</b> continue to support the Exchange Network Header version used in the existing version of the exchange.
[XD6-15]	When an exchange undergoes a significant redesign, the latest approved Exchange Network Header <b>SHOULD</b> be implemented.

## Appendix B – Glossary of Terms

The following terms are used throughout this document.

Term	Definition
Data Exchange Template (DET)	A document that lists all the elements within a schema and additional information about their definition, origin, and use, and may include a listing of business rules.
Data Exchange	The sharing of data between two or more Network partners implemented through a specific set of shared services and functions, typically related to a specific environmental subject, program, or dataset.
Data Service	The operations supported by an exchange, typically initiated through the invocation of a Query, Solicit, Submit or Execute primitive command as defined in the exchange FCD.
Exchange Documentation Package	A bundled set of XML schemas and documentation submitted to Network governance for review and approval. The Exchange Documentation Package must be reviewed, approved and posted to the Exchange Network web site before an exchange may be implemented in production.
Flow Configuration Document (FCD)	A document that conveys the detailed data exchange processing rules governing the data exchange.
Integrated Project Team (IPT)	An individual or group of individuals responsible for developing a new exchange or enhancing an existing exchange. An IPT is typically composed of stakeholders from multiple organizations and interests and includes both business area experts and technical staff.
Trading Partner Agreement (TPA)	An agreement in the form of documents formally adopted by two or more partners for the purpose of defining the responsibilities of each party, the legal standing (if any) of the proposed exchange, and the technical details necessary to initiate and conduct electronic information exchange.