Water Quality Exchange

Flow Configuration Document

Version 1.0

October 18, 2006

Developed for:
United States Environmental Protection Agency
Office of Environmental Information
1200 Pennsylvania Avenue, NW
Washington, DC 20460

Developed by:

LOCKHEED MARTIN

Systems Engineering Center (SEC)
1010 N. Glebe Road
Arlington, VA 22201

ITS-ESE Task Order No.: TO 08

Contract No.: 68-W-04-005

Environmental Information exchange Network

Table of Contents

1.0 Introduction

1.1 Background

The U.S. Environmental Protection Agency (EPA) Office of Information Collection (OIC), Office of Wetlands, Oceans and Watersheds (OWOW), and Office of Water (OW) are committed to implementing Central Data Exchange (CDX) services and establishing the EPA infrastructure to support an ambient water quality monitoring data exchange. The Water Quality Exchange project is the product of a collaborative effort between OIC, OW, and the Environmental Council of States (ECOS). The project was identified during the development of the Environmental Sampling Analysis and Results (ESAR) data standard for water monitoring.

The project goal is to provide EPA state partners with a means of exchanging water quality monitoring data via CDX, utilizing the ESAR data standard as much as possible. The Office of Water, in partnership with the states, establishes water quality monitoring data exchange elements, business rules for exchanging these elements, and valid domain lists for elements not covered by an existing or proposed standard.

The Water Quality Exchange (WQX) pilot established a data flow through which three initial pilot states (Oregon, Michigan, and Texas), as well as the Wind River Environmental Quality Commission (WREQC), could submit Water Quality Monitoring (WQM) data to EPA via the CDX Exchange Network node.[1]

The WQX pilot provided valuable lessons learned and a solid proof of concept for the new approach to sharing water quality data. The production phase of development will include moving data from the WQX Operational Data Store (ODS) to the STORET Data Warehouse.

The WQX System includes the following types of data:

- The physical conditions in the environment at the time of a site visit.
- The chemical and bacteriological make-up of the water sampled.
- Chemical analyses of fish tissue collected.

The WQX data flow utilizes two mechanisms for exchanging water quality monitoring information:

- Web services-based solution for automated submission utilizing Exchange Network standards.
- Web-based solution for manual submissions (included with an XML Generation Tool to be developed at a later date).

1.2 Document Purpose

This Flow Configuration Document (FCD) is intended to define the supported data services, the approaches and processes that are used to exchange information over the Exchange Network via web services technology. In addition, the FCD serves as a guide for trading partners in understanding the details and challenges associated with a data flow.

The purpose of this FCD is to describe the operation of the Water Quality Monitoring Network Exchange using XML-based data submissions through Node-to-Node or Client-to-Node transfers. This document does not consider use of the alternative ESAR formats or transfer mechanisms. The focus of the document is the core WQX data flow between the state, local, and tribal agencies and EPA.

The Office of Water is separately committed to providing outbound services from the STORET Data Warehouse, which combines WQX data and STORET data. These query web services are outside the scope of this document.

1.3 How to use this FCD

This FCD provides guidance to implement an XML/web-service based model for data submission. For the WQX project, there are two types of submissions, Update-Insert and Delete. This FCD expands upon the usage of the WQX Schema and introduces the implementation of the Document Header.

This document includes the following main sections:

[1] *Note: WREQC is a joint commission of the Arapaho and Shoshone tribes.*

Implementing the Header Document

This section describes how the WQX Network Exchange makes use of the Exchange Network Header Document to describe the payload content of a Network message. This submission structure is used for both Node-to-Node and Client-to-Node submissions as well as the web submission (to be implemented at a later phase).

Data Processing

This section describes the data processing rules in effect in the WQX System.

Submission Processing and Feedback

This section describes the processing steps and status changes that occur as your submission flows through CDX and the WQX System.

Web Service Methods at CDX

This section describes the Web Service Methods at CDX used by the WQX Data Flow.

2.0 Implementing the Header Document

2.1 Overview

The Exchange Network Header Document provides additional information about the contents of a message payload. It is developed to further automate the data exchange process so that data can be more readily identified during transport and managed at its processing destination. The Header Document can describe what a data payload contains, who submitted it, when it was submitted, as well as instructions on processing the payload contents, such as whether the contents are inserts and updates or deletions.

Essentially, the Header Document is an XML wrapper, placed around an XML payload before transmission to CDX. A Header Document toolkit is available in the Tool Box section of the Exchange Network Web site (http://www.exchangenetwork.net), containing additional background about the Header Document, as well as Java and .NET implementation tools.

2.2 Header Document Structure

Any network exchange for WQX must use the Header Document structure in order to meet EPA CDX and WQX processing requirements and prior Exchange Network agreements.

The Header Document contains a Header and Payload section. The Header section contains information about the submission. The Payload contains the WQX data. Although some data flows allow for multiple payloads in one document, the WQX data flow contains only a single payload. The Payload must conform to one of the two WQX standard schemas for data submissions.

The following table describes the Header Document elements and how they are utilized for the purpose of a WQX submission.

Header Section				
Element	**Description**	**Example Value**	**Required**	**Notes**
Author	First and Last Name of individual generating the XML document	Joe Smith	Yes	Reference only
Organization	Name of company, environmental agency or individual generating the XML document	State X Department of Environmental Quality	Yes	Reference only
Title	Type of Submission	Must be “WQX”	Yes	Reference to the flow
Creation Time	Date/Time when the document was generated	2003-01-01T12:12:12 (where date is a valid XML date format string)	Yes	Reference
Comment	Free text description of the message contents.		No	Reference
Data Service	Unused	N/A	No	Unused

Header Section				
Element	**Description**	**Example Value**	**Required**	**Notes**
Contact Info	Name, mailing address, city, state, zip, telephone number, and email address of person who may be contacted with questions concerning the submission.	Joe Smith 123 Main St. Portland, OR 97226 503 123 4567 Joe@deq.statex.gov	Yes	Reference
Notification	Unused	N/A	No	Unused
Sensitivity	Level of Document Sensitivity	N/A	No	Unused
Property	Name Value pairings used to describe specific properties of the document.	N/A	No	Unused
Payload Section				
Element	**Description**	**Example Value**	**Required**	**Notes**
Operation (attribute)	This describes the operation to be performed on the payload. Multiple values are not allowed	Must be either “Update-Insert” or “Delete”	Yes	
Schema Reference	WQX approved schema for submission	Must be either: “WQX_WQX_v1.0.xsd” or “WQX_WQX_Delete_v1.0.xsd”	Yes	

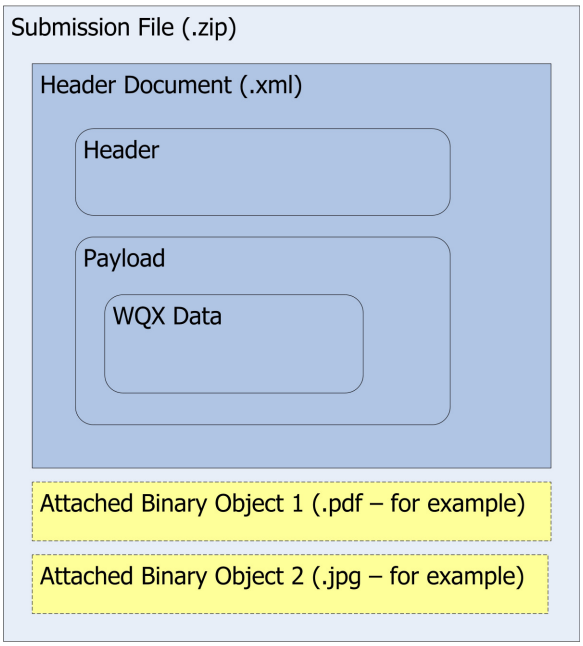
2.3 Submission File Structure

In addition to the XML Header Document (containing the Header and Payload), a submission can also include binary objects (such as images or documents). These binary objects are referenced (by unique name) in the WQX Document, but are provided as individual, external documents. In order to group all related documents for a submission, each WQX submission is provided to CDX in a single compressed file in ‘.zip’ format. This also improves transmission efficiency due to the significantly smaller size of compressed files. Even if a submission does not contain any external binary objects, it is still strongly recommended that it be provided in ‘.zip’ format (although single ‘.xml’ documents will be allowed).

Additionally, the Header Document within the submission is the only file allowed to have an ‘.xml’ extension. This allows CDX to differentiate between the Header Document and any other binary object files included in the submission.

For WQX, a submission file is limited to 80 Megabytes (whether it is compressed or not). “Zipping” a submission file will allow a much greater amount of data to fit within this 80 MB limit.

The following diagram describes the structure of a Submission Zip File for WQX:



2.4 Payload

The Payload Section of the Header Document contains an attribute named “Operation”. This is used to denote the type of processing for a submission. There are two acceptable values: “Update-Insert” or “Delete”. Use of these operators triggers the processing outlined in the *Configuration of the Network Exchange* section of this document. For example, a payload operation of “Update-Insert” informs the back-end application that the payload contains new or existing data. ”Delete” denotes that the payload contains identifiers for data to be deleted.

The Payload section of the document contains the WQX data, which must conform to one of the two WQX standard schemas for data submissions. If the “Operation” attribute is “Update-Insert” then the Payload must conform to the Update-Insert Schema (“WQX_WQX_v1.0.xsd”). If the “Operation” attribute is “Delete” then the Payload must conform to the Delete Schema (“WQX_WQX_Delete_v1.0.xsd”).

3.0 Data Processing

The root element for each of the two valid schemas for WQX is the OrganizationIdentifier. This is a unique identifier for an organization and must be assigned by the EPA before that organization can submit data to the WQX system. An organization cannot be created or deleted via a data submission.

Before processing any submission file, the WQX system will first validate that the Organization Identifier in the file is valid and that the User ID used to authenticate and submit the file to CDX has been authorized to modify data for that organization.

Once the user and organization have been validated, the WQX system will proceed to process the submission file. The following sections explain the processing that occurs for each type of payload: “Update-Insert” and “Delete”.

3.1 Update-Insert

The WQX Schema for the “Update-Insert” operation is described in Exhibit 1. This schema defines one root component for the entire hierarchy: Organization. Each submission file must contain one (and only one) Organization, uniquely identified with an OrganizationIdentifier.

In addition to the Organization, four other primary components exist which must also be uniquely identified (within the organization): Project, Monitoring Location, Activity, and Activity Group. These primary components can be recognized on the Schema Model by their heavy border. A valid WQX submission file only needs one of these primary components to be valid. In other words, you are allowed to submit a file of Projects only, Monitoring Locations only, Activities only, or Activity Groups only. Likewise, it is also valid to submit many of these components in a single file (e.g. Projects, Monitoring Locations, and Activities in one file).

Because these are the only components that have unique identifiers, this is the only level at which updates can occur. All secondary components in the schema must be updated along with their respective parents. For example: The “Result” component is a child of the “Activity” component in the schema. Results are not uniquely identified. Therefore, in order to update a Result that has been previously submitted to the WQX System, you must resubmit the parent Activity and all Results that fall below it (including the individual Result you wish to update).

The WQX System will automatically determine if the primary component you have submitted is to be Inserted or Updated in the database. The following procedure is used to make this determination: the unique identifier for each Project, Monitoring Location, Activity, and Activity Group in the payload is compared with the identifiers for these respective items in the WQX database. If the identifier is found, an “Update” is performed, replacing the record in the database with the copy from the payload. If it is not found, an “Insert” is performed, creating a new record in the database. Accurately updating the WQX System is only possible if you have uniquely identified each of the primary components in your submission file.

Once a primary component has been updated, the following procedure defines how its children are affected: if any children are included in the payload, then all children in the database are deleted before inserting the new children from the payload. The existence of a child in the payload is defined by the existence of the child’s tag. For example: If an Activity is included in the payload, but no Results are provided along with it (not even the <Result> tag) then the Activity will be updated but all existing Results in the database will be unaffected. However, if you do provide one or more Results, then all Results for that Activity will be deleted from the database before inserting the ones provided in the payload. In other words, if you include one Result in the payload, you must provide them all. One idiosyncrasy is worth noting: if you provide an empty Result tag (i.e. <Result></Result> or just <Result/>) then all Results for the Activity will be deleted from the database and no new ones will be inserted (leaving no Results).

The WQX System performs a commit (i.e. save) each time it successfully loads all of the data belonging to one of these primary components. If there are any errors encountered while loading that component (or its children), all changes will be rolled back (i.e. not saved). For example: if one Result has an error in it, then its Activity (which is the primary component) and all other Results on that Activity will fail and not be saved. This error, however, only affects this one Activity. All other Activities in the file that are loaded error-free will still be saved. This rule relates to each of the primary components. Each Project, Monitoring Location, Activity, and Activity Group that is free of errors will be loaded into the WQX System. Those containing errors will not be loaded.

If any errors occur while loading the submission file into the WQX System, the submission status will be set to "Failed" at CDX. Likewise, the Processing Report will include an itemized list of any primary components that failed to load. See Appendix A for an example Processing Report.

If a submission fails to load successfully, there are two approaches to correcting it.

1) Resubmit the entire data set again (after correcting the data that originally failed). In this case, any data that originally loaded successfully will now be treated as an "Update". Any data that originally failed will be treated as an "Insert".
2) Submit a new, corrected, data set that only includes the components that originally failed. This new file will be more efficient to process because it is likely much smaller than the complete data set.

3.1.1 *Data Rules*

The following is a summary of the data validation rules enforced on the XML submission document.

- When ElectronicAddressText or ElectronicAddressTypeName is reported, both must be reported.
- When TelephoneNumberText or TelephoneNumberTypeName is reported, both must be reported.
- When LocationAddressText or AddressTypeName is reported, both must be reported.
- When HorizonitalCollectionMethodName is "INTERPOLATION-MAP", SourceMapScaleNumeric must be reported.
- When VerticalMeasure's MeasureValue is reported, the following also must be reported:
  VerticalMeasure's MeasureUnitCode,
  VerticalCollectionMethodName,
  VerticalCoordinateReferenceSystemDatumName.
- Either ProjectDescriptionText or Project's BinaryObjectFileName must be reported.
- Activity Depth can be reported in only one of the following two ways (but not both):
  - Specific depth using ActivityDepthHeightMeasure's MeasureValue.
  - Depth Range using ActivityTopDepthHeightMeasure's MeasureValue and ActivityBottomDepthHeightMeasure's MeasureValue.
    - This method must be used when ActivityTypeCode is "Sample-Integrated Vertical Profile".
- When ActivityTypeCode contains the word 'Logger', DataLoggerLineName must be reported.
- When SampleTissueTaxonomicName or SampleTissueAnatomyName is reported, both must be reported.
- When ActivityTypeCode contains the word 'Sample', SampleCollectionMethod's MethodIdentifier must be reported.
- When ActivityMediaName contains the word 'Tissue', SampleTissueTaxonomicName and SampleTissueAnatomyName must be reported.
- When ResultDetectionConditionText contains the word 'Quantification', DetectionQuantitationLimitTypeName must be reported.
- When ResultDetectionConditionText is 'Not Detected', DetectionQuantitationLimitTypeName and DetectionQuantitationLimitMeasure must be reported.
- Either ResultMeasure's ResultMeasureValue or ResultDetectionConditionText must be reported, but not both.
- When ResultMeasure's ResultMeasureValue is reported, CharacteristicName and ResultStatusIdentifier must be reported.
- When DetectionQuantitationLimit's MeasureValue is reported, DetectionQuantitationLimit's MeasureUnitCode must be reported.
- ActivityDescription's MonitoringLocationIdentifier may be required depending on the value provided for ActivityTypeCode. See the domain value list for ActivityTypeCode for more information.
- ResultAnalyticalMethod may be required depending on the value provided for ActivityTypeCode. See the domain value list for ActivityTypeCode for more information.
- ResultSampleFractionText may be required depending on the value provided for CharacteristicName. See the domain value list for CharacteristicName for more information.
- If ResultAnalyticalMethod's MethodIdentifierContext does not match a value from the AnalyticalMethodContext domain list then it must be the same as the OrganizationIdentifier.
- If ResultAnalyticalMethod's MethodIdentifierContext matches a value from the AnalyticalMethodContext domain list, then ResultAnalyticalMethod's MethodIdentifier and MethodName must match a value from the AnalyticalMethod domain list.

- ProjectIdentifier, MonitoringLocationIdentifier, ActivityIdentifier, and ActivityGroupIdentifier will be treated as case-insensitive by WQX. For example, the following three identifiers would be treated as identical: “Mx571”, “mx571”, “MX571”.
- If Sample Preparation block is reported, then either ChemicalPreservativeUsedName or ThermalPreservativeUsedName must be reported.

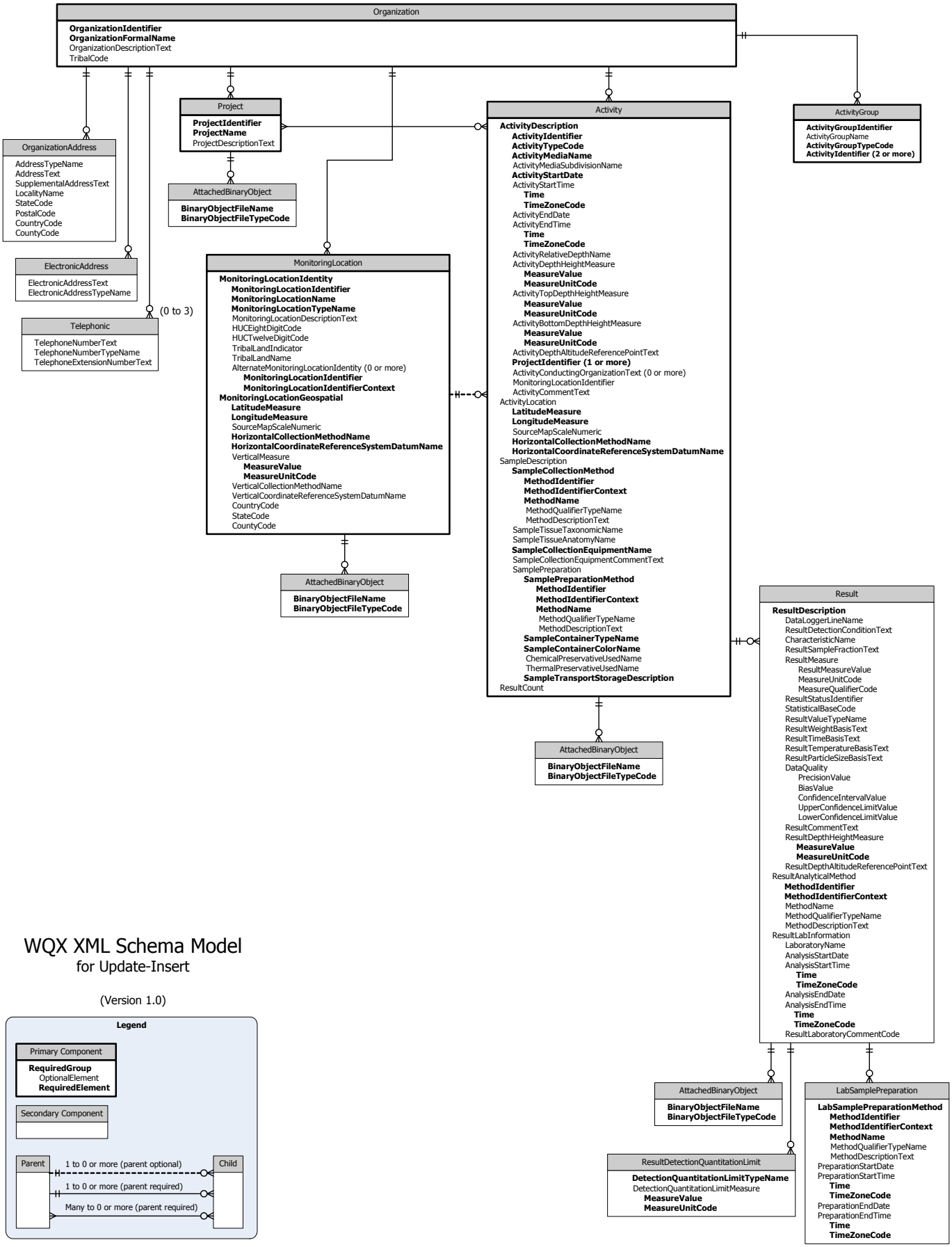


Exhibit 1 – Update-Insert Schema

3.2 Delete

The XML Schema for the "Delete" operation is very simple (see Exhibit 2): It includes the identifiers for the five primary components in the system.

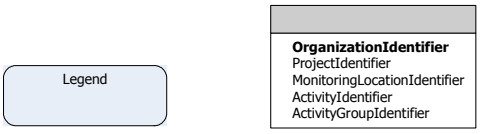


Exhibit 2 – Delete Schema

OrganizationIdentifier is required (and must be provided only once). The OrganizationIdentifier is required for context information only (organizations cannot be deleted).

In addition to the OrganizationIdentifier, at least one of the other four identifiers (e.g. ProjectIdentifier, MonitoringLocationIdentifer, ActivityIdentifier, or ActivityGroupIdentifier) must be provided and can be repeated as many times as necessary to identify all of the records to be deleted from the WQX Database.

The following procedure defines the delete process:

The unique identifier for each Project, Monitoring Location, Activity, and Activity Group in the payload is compared with the identifiers for these respective items in the WQX Database. If the identifier is not found, a warning is added to the Processing Report, and the identifier is ignored. If the identifier is found, then the matching record in the database is deleted.

When one of these primary components is deleted from the system, all child data is deleted as well.

For example:

- AttachedBinaryObjects are deleted whenever the parent Project, MonitoringLocation, Activity or Result is deleted;
- Results are deleted whenever their parent Activity is deleted;
- Activites are deleted whenever their parent Project or MonitoringLocation is deleted.

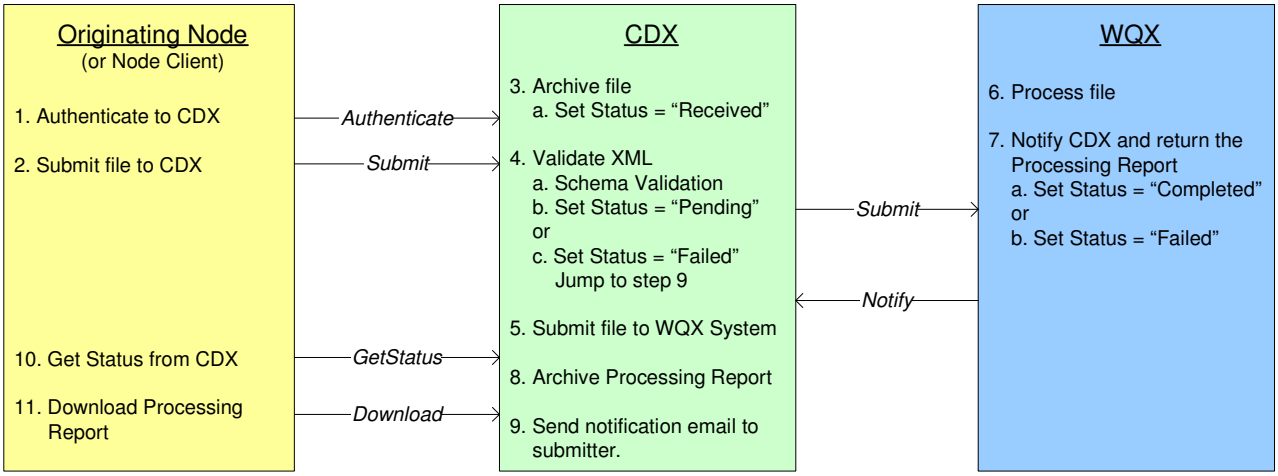
Two exceptions exist:

1. Activities can relate to more than one Project.
   In this case, when only one of the Projects it relates to is deleted, the Activity will remain (only the relationship between the Project and the Activity will be deleted). When an Activity relates to only one Project, it will be deleted when that Project is deleted.
2. Activities may or may not relate to a Monitoring Location.
   When an Activity relates to a Monitoring Location it will be deleted when its Monitoring Location is deleted. When it does not relate to a Monitoring Location (such as for certain Quality Control Samples) it will be unaffected by any Monitoring Location deletes.

4.0 Submission Processing and Feedback

Before a WQX submission is made to CDX, a Node User must register with Network Authentication Authorization (NAAS) and obtain a user account. Furthermore, a NAAS policy is established that allows the account to invoke specific methods on the CDX Node for the WQX exchange.

Submission of Exchange Network Documents to the WQX System via the CDX Node follows these processing steps:



1. The Originating Node/Node Client calls the *Authenticate* method to initiate a session with CDX.
    a. If authentication is successful, a Security Token will be returned.
2. The *Submit* method is called to submit a WQX file for processing.
    a. A Transaction ID is returned, indicating that the file transfer was successful.
3. The submission file (.zip or .xml) is archived at CDX.
    a. The status of the submission is set to "Received".
4. The XML submission file is validated at CDX:
    a. The file is compared with the Header Schema and the relevant WQX Schema to confirm that it complies with the required structure.
    b. If the XML file passes validation, the submission status is set to "Pending".
    c. Otherwise, a Processing Report is generated and the submission status is set to "Failed". No further processing occurs (skip to step 9).
5. CDX submits the file to the WQX System (by calling its *Submit* method).
6. The WQX System processes the submission file (i.e. performs database inserts, updates, and deletes as indicated in the file) and loads any attached binary objects.
7. The WQX System calls the *Notify* method at CDX to update the submission status and return a Processing Report.
    a. If the file was processed without errors, then the status is set to "Completed".
    b. If there were errors, then the status is set to "Failed".
8. The Processing Report is archived at CDX.
9. An email is sent to the submitter notifying him/her of the final status of the submission ("Failed" or "Completed").
10. The Originating Node/Node Client calls the *GetStatus* method to determine the status of the submission. If the Originating Node is manually controlled and the submitter is already aware of the status (from the email in step 9), then this step could be skipped.

11. Once the status is either “Completed” or “Failed”, the *Download* method is called to retrieve the Processing Report from CDX.

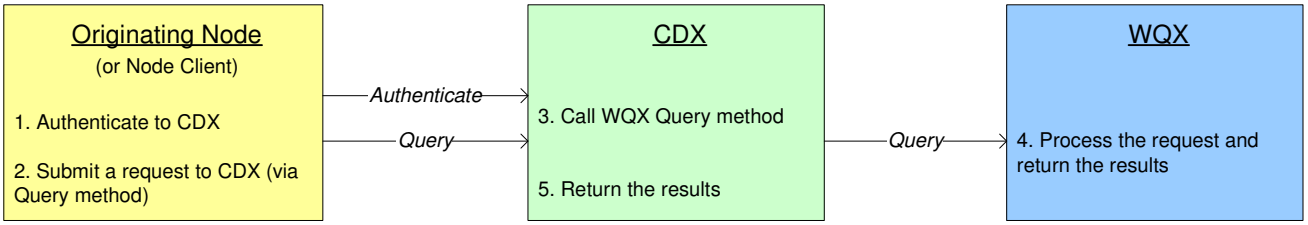
    a. The Processing Report is an XML-based summary of the processing that occurred on the WQX System (including processing time, insert/update/delete counts and any error or warning messages). See Appendix A for an example Processing Report.

5.0 Query Processing and Feedback

A query allows a State/Regional/Tribal node to request data back from the WQX system. There are two methods for retrieving this data: the Query method, which returns the results immediately, and the Solicit method which creates a data file (offline) which can be downloaded (or submitted back to the requestor's node) at a later time. The Query method is ideal for smaller requests and may have restrictions on the size of the results returned. The Solicit method can accommodate larger requests.

5.1 Query Process

The Query process follows these processing steps:



1. The Originating Node/Node Client calls the Authenticate method to initiate a session with CDX.
    a. If authentication is successful, a Security Token will be returned.
2. The *Query* method is called to request data from CDX.

*3. CDX forwards the request on to the WQX System (via the *Query* method).

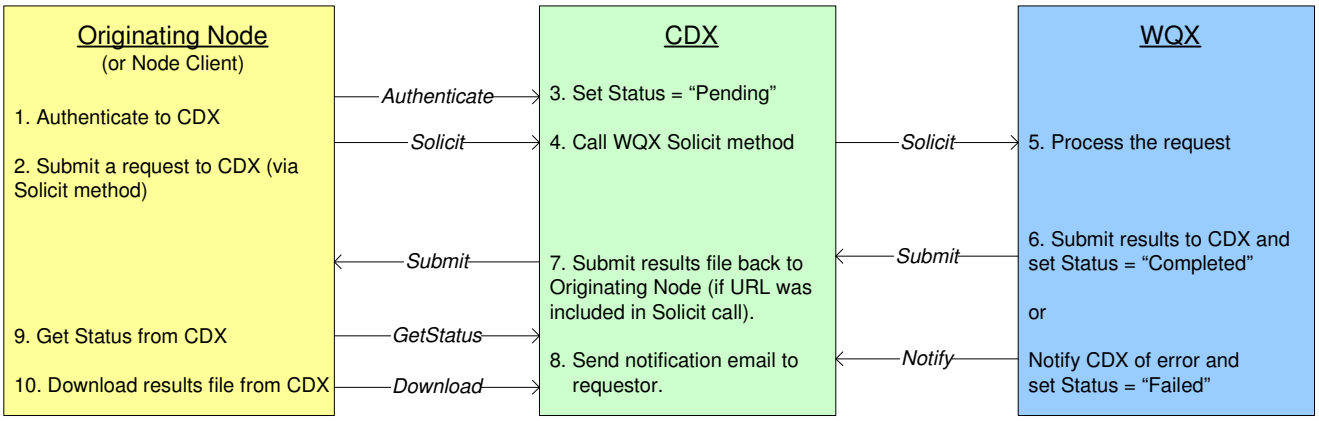
*4. The WQX System processes the request and returns the results to CDX.

*5. CDX returns the results to the Originating Node/Node Client.

* Steps 3, 4, and 5 are all transparent to the Originating Node. The Query method call (in step 2) returns the results (mentioned in step 5).

5.2 Solicit Process

The Solicit process follows these processing steps:



1. The Originating Node/Node Client calls the *Authenticate* method to initiate a session with CDX.
    a. If authentication is successful, a Security Token will be returned.
2. The *Solicit* method is called to request data from CDX.
    a. A Transaction ID is returned.
3. The Transaction Status is set to "Pending".
4. CDX calls the *Solicit* method on the WQX System.

5. The WQX System processes the request and creates a results file corresponding to the criteria it was passed.
6. If the request is successful, then the WQX System calls the *Submit* method to return the results file and set the status to "Completed".

   If the request fails, then the WQX System calls the *Notify* method to return the error message and set the status to "Failed".
7. If the *Solicit* request from the Originating Node includes a return URL, then CDX will submit the results file back using that node's *Submit* method. This cannot be done with a Node Client.
8. An email is sent to the requestor notifying him/her of the final status of the solicit request ("Failed" or "Completed").
9. The Originating Node/Node Client calls the *GetStatus* method to determine the status of the transaction. If the Originating Node is manually controlled and the requestor is already aware of the status (from the email in step 8), then this step could be skipped.
10. Once the status is either "Completed" or "Failed", the *Download* method is called to retrieve the results file.

6.0 Web Service Methods at CDX

The following sections discuss the web service methods available at CDX for the WQX data flow. This reference can be used by the developers of a State/Regional/Tribal Node in implementing the interface between their node and CDX. For more information, see the Network Node Functional Specification document available on the Exchange Network web site (www.exchangenetwork.net):

6.1 Authenticate

The authenticate method is used to obtain a security token from the Network Authentication Authorization Service (NAAS). This token will be passed in all subsequent method calls in the exchange between your node and CDX. Please note that security tokens will expire 20 minutes after the token is issued.

Parameters:

- userId: the User ID identifying your node.
- credential: the password you were issued along with the User ID.
- authenticationMethod: the method used to authenticate. Currently only "password" is allowed

Returns: a securityToken used to identify your session with CDX.

6.2 Submit

The Submit method is used to send your WQX submission file to CDX.

Parameters:

- securityToken: A security token issued by the NAAS and returned from the Authenticate method.
- transactionId: unused.
- dataflow: The name of target dataflow: "WQX"
- documents: An array of type nodeDocument. Each nodeDocument structure contains a single submission document/file. For WQX, only a single instance of nodeDocument is submitted.

Returns: a transactionId which identifies your submission. This can be used in the GetStatus and Download methods (described below).

6.3 GetStatus

The GetStatus method is used to check on the current status of a submission file being processed or a solicitation for WQX data. Generally this is used to followup on a previous call to the Submit or Solicit method.

Parameters:

- securityToken: A security token issued by the NAAS and returned from the Authenticate method.

- transactionId: A Transaction ID returned by the Submit or Solicit method.

Returns: the status of the specified transaction: Received, Pending, Completed, or Failed.

6.4 Download

The Download method is used to download documents from the CDX node relating to a specific transactionId. Generally, this method is used after the GetStatus returns either “Completed” or “Failed”.

There are two purposes for the Download method:

1. Retrieve the Processing Report after a submission file has been processed.
    a. This report describes the processing that occurred on your submission and any errors or warnings that arose.
2. Retrieve the results of a request for WQX data using the Solicit method.

Parameters:

- securityToken: A security token issued by the NAAS and returned from the Authenticate method.
- transactionId: A transaction ID returned by the Submit or Solicit method.
- dataflow: The name of the dataflow: “WQX”.
- documents: (optional).

  If this parameter is left blank, then all documents relating to this transaction will be returned. This parameter is useful to avoid downloading things you don’t need. For example: pass in the Processing Report name to download just the processing report, and avoid downloading your original submission file along with it.

  The documents parameter is made up of the following three fields:

  - name: the name of the document you wish to download.
  - type: the file type (XML, ZIP, OTHER).
  - content: not used.

Returns: a set of documents.

The following documents are available for download:

<u>For a *Submit* transaction</u>:

1. The original submission document.

2a. The Schema Validation Error Report (this will only exist in cases where CDX found a problem with the format of your XML submission file).
(name = “Validation Results”, type = “XML”)

or

2b. The Processing Report from the WQX System.
(name = “DocumentStatus_ +[original document name]”, type = “XML”)

<u>For a *Solicit* transaction</u>:

1. The results file.
(name = “Results.zip”, type = “ZIP”).

6.5 Query

The Query method is used to query the WQX system and retrieve data from it. Queries have restrictions to keep the size of the output reasonably small. For requests that may return a larger set of data, please use the Solicit method. Section 6.6.1 describes the Query and Solicit Requests that are currently support by the WQX system.

Parameters:

- securityToken: A security token issued by the NAAS and returned from the Authenticate method.
- request: The name of the query to be performed (see section 6.6.1 for more details).
- rowId: not used.
- maxRow: not used.
- parameters: An array of parameter values for the query to be performed. Parameters are specific to the request that is made (see section 6.6.1 for details).

Returns: a set of data in an XML instance document. The returned data set is dependant on the request made.

6.6 Solicit

The Solicit method is used to query the WQX system and retrieve data from it. Unlike the Query method, which returns the results immediately, the Solicit method processes the request offline and creates a zipped XML document that can be downloaded (or submitted back to the requestor's node) at a later time. The Solicit method is also able to handle requests for a larger volume of data than the Query method. Section 6.6.1 describes the requests that are currently support by the WQX system.

Parameters:

- securityToken: A security token issued by the NAAS and returned from the Authenticate method.
- returnURL: A Node address where results can be submitted. CDX will call the requestor node's *Submit* method at this URL once the results are ready. If returnURL is empty, then it is the requestor's responsibility to download the result from CDX.
- request: The name of the query to be performed (see section 6.6.1 for more details).
- parameters: An array of parameter values for the query to be performed. Parameters are specific to the request that is made (see section 6.6.1 for details).

Returns: a transactionId which identifies your request. This can be used with the *GetStatus* and *Download* methods to retrieve your results.

6.6.1 *Query/Solicit Requests*

WQX will support the following Requests available via the *Query* or *Solicit* Web Service Methods:

- **WQX.GetActivityByParameters_v1.0**
  - Description: Returns a collection of Activities and a count of the number of Results on each Activity.
  - Available via: Query or Solicit
    - Query requests are limited to 500 Activities.
  - Parameters:

Pos	Name	Type	Required	Notes
1	organizationIdentifier	String	Required	
2	monitoringLocationIdentifier	String	Optional	
3	projectIdentifier	String	Optional	

4	activityStartDateBegin	DateTime	Optional	Example format: 2005-10-16T14:00:00-06:00 or 2005-10-16
5	activityStartDateEnd	DateTime	Optional	Example format: 2005-10-16T14:00:00-06:00 or 2005-10-16
6	activityIdentifier	String	Optional	

- Return Schema: WQX_WQX_v1.0.xsd (Activity Section only).

- **WQX.GetActivityGroupByParameters_v1.0**
  - Description: Returns a collection of Activity Groups.
  - Available via: Query or Solicit.
  - Parameters:

Pos	Name	Type	Required	Notes
1	organizationIdentifier	String	Required	
2	activityGroupTypeCode	String	Optional	
3	activityGroupIdentifier	String	Optional	

  - Return Schema: WQX_WQX_v1.0.xsd (ActivityGroup Section only).

- **WQX.GetDomainValueByElementName_v1.0**
  - Description: Returns a list of domain values for the elementName passed in.
  - Available via: Query or Solicit
    - Query requests are limited to one Domain List (i.e. elementName is required).
  - Parameters:

Pos	Name	Type	Required	Notes
1	elementName	String	Required for Query Optional for Solicit	If the elementName is null, then all domain value lists are returned. Otherwise, elementName must be one of the following: ActivityGroupType ActivityMedia ActivityMediaSubDivision ActivityRelativeDepth ActivityType AddressType AnalyticalMethod AnalyticalMethodContext Characteristic Country County DetectionQuantitationLimitType ElectronicAddressType HorizontalCollectionMethod HorizontalCoordinateReferenceSystemDatum MeasureUnit

				MonitoringLocationType ResultDetectionCondition ResultLaboratoryComment ResultMeasureQualifier ResultMeasureValuePickList ResultSampleFraction ResultStatus ResultTemperatureBasis ResultTimeBasis ResultValueType ResultWeightBasis SampleCollectionEquipment SampleContainerColor SampleContainerType SampleTissueAnatomy Taxon State StatisticalBase TelephoneNumberType TimeZone ThermalPreservativeUsed Tribe VerticalCollectionMethod VerticalCoordinateReferenceSystemDatum

- Return Schema: WQX_DomainValues_v1.0.xsd.

- **WQX.GetMonitoringLocationByParameters_v1.0**
  - Description: Returns a collection of Monitoring Locations.
  - Available via: Query or Solicit
    - Query requests are limited to 1000 Monitoring Locations.
  - Parameters:

Pos	Name	Type	Required	Notes
1	organizationIdentifer	String	Required	
2	monitoringLocationIdentifier	String	Optional	

  - Return Schema: WQX_WQX_v1.0.xsd (MonitoringLocation Section only).

- **WQX.GetProjectByParameters_v1.0**
  - Description: Returns a collection of Projects for the organization passed in.
  - Available via: Query or Solicit
    - Query requests are limited to 1000 Projects.
  - Parameters:

Pos	Name	Type	Required	Notes
1	organizationIdentifier	String	Required	
2	projectIdentifier	String	Optional	

  - Return Schema: WQX_WQX_v1.0.xsd (Project Section only).

- **WQX.GetResultByParameters_v1.0**

- Description: Returns a collection of Activities and Results.
- Available via: Query or Solicit.
  - Query requests are limited to 500 Results.
- Parameters:

Pos	Name	Type	Required	Notes
1	organizationIdentifier	String	Required	
2	monitoringLocationIdentifier	String	Optional	
3	projectIdentifier	String	Optional	
4	activityStartDateBegin	DateTime	Optional	Example format: 2005-10-16T14:00:00-06:00 or 2005-10-16
5	activityStartDateEnd	DateTime	Optional	Example format: 2005-10-16T14:00:00-06:00 or 2005-10-16
6	activityIdentifer	String	Optional	

- Return Schema: WQX_WQX_v1.0.xsd (Activity & Results Sections only).

- **WQX.GetTransactionHistoryByParameters_v1.0**
  - Description: Returns a summary of the transactions processed by the WQX System.
  - Available via: Query or Solicit.
  - Parameters:

Pos	Name	Type	Required	Notes
1	organizationIdentifier	String	Conditionally Required	One of these three parameters must have a non-null value
2	userIdentifier	String	Conditionally Required	
3	transactionIdentifier	String	Conditionally Required	
4	transactionDateBegin	DateTime	Optional	Example format: 2005-10-16T14:00:00-06:00 or 2005-10-16
5	transactionDateEnd	DateTime	Optional	Example format: 2005-10-06T14:00:00-06:00 or 2005-10-16

  - Return Schema: WQX_TransactionHistory_v1.0.xsd.

Appendix A. – Example Processing Report

```
<?xml version="1.0" encoding="UTF-8"?>
<Notify xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="notification.xsd">
  <Status>FAILED</Status>
  <Operation>SUBMIT</Operation>
  <DataFlow>WQX</DataFlow>
  <Timestamp>2006-12-12T12:12:12.0000000-05:00</Timestamp>
  <TransactionID>59df421d-d086-4e3e-a780-40f7514d9c7b</TransactionID>
  <flowSpecificInfo>
    <ProcessingReport>
      <ProcessingSoftware Version=”1.00.2167.30307” Parse And Load></ProcessingSoftware>
      <ProcessingSoftware Version=”1.00.00” WQX Database></ProcessingSoftware>
      <Counts>
        <Error>1</Error>
        <Warning>1</Warning>
        <Project Action="Insert">1</Project>
        <Project Action="Update">0</Project>
        <Project Action="Delete">0</Project>
        <MonitoringLocation Action="Insert">83</MonitoringLocation>
        <MonitoringLocation Action="Update">0</MonitoringLocation>
        <MonitoringLocation Action="Delete">0</MonitoringLocation>
        <Activity Action="Insert">1718</Activity>
        <Activity Action="Update">10</Activity>
        <Activity Action="Delete">0</Activity>
        <ActivityGroup Action="Insert">7</ActivityGroup>
        <ActivityGroup Action="Update">1</ActivityGroup>
        <ActivityGroup Action="Delete">0</ActivityGroup>
        <Result Action="Insert">32021</Result>
        <Result Action="Update">0</Result>
        <Result Action="Delete">0</Result>
      </Counts>
      <Log>
        <LogDetail>
          <Type>Message</Type>
          <Text>Parse and Load started at 01/24/2006 03:29:37 PM</Text>
          <Context></Context>
        </LogDetail>
        <LogDetail>
          <Type>Warning</Type>
          <Text>Activity could not be deleted.  Activity ID "25792446" not found.</Text>
          <Context>MonitoringLocationDelete/Line 22, Activity ID = 25792446</Context>
        </LogDetail>
        <LogDetail>
```

```
          <Type>Error</Type>
          <Text>File "Station_CBC050.jpg" cannot be found.</Text>
          <Context>AttachedBinaryObject/Line 274</Context>
        </LogDetail>
        <LogDetail>
          <Type>Message</Type>
          <Text>Parse and Load completed at 01/24/2006 03:45:40 PM</Text>
          <Context></Context>
        </LogDetail>
      </Log>
      <ProcessingFailures>
        <ProjectIdentifier>982</ProjectIdentifier>
        <ProjectIdentifier>678</ProjectIdentifier>
        <MonitoringLocationIdentifier>288892A</MonitoringLocationIdentifier>
        <MonitoringLocationIdentifier>293848X</MonitoringLocationIdentifier>
        <ActivityIdentifier>1928389822</ActivityIdentifier>
        <ActivityIdentifier>3823239822</ActivityIdentifier>
        <ActivityGroupIdentifier>19228</ActivityGroupIdentifier>
        <ActivityGroupIdentifier>19228</ActivityGroupIdentifier>
      </ProcessingFailures>
    </ProcessingReport>
  </flowSpecificInfo>
</Notify>
```