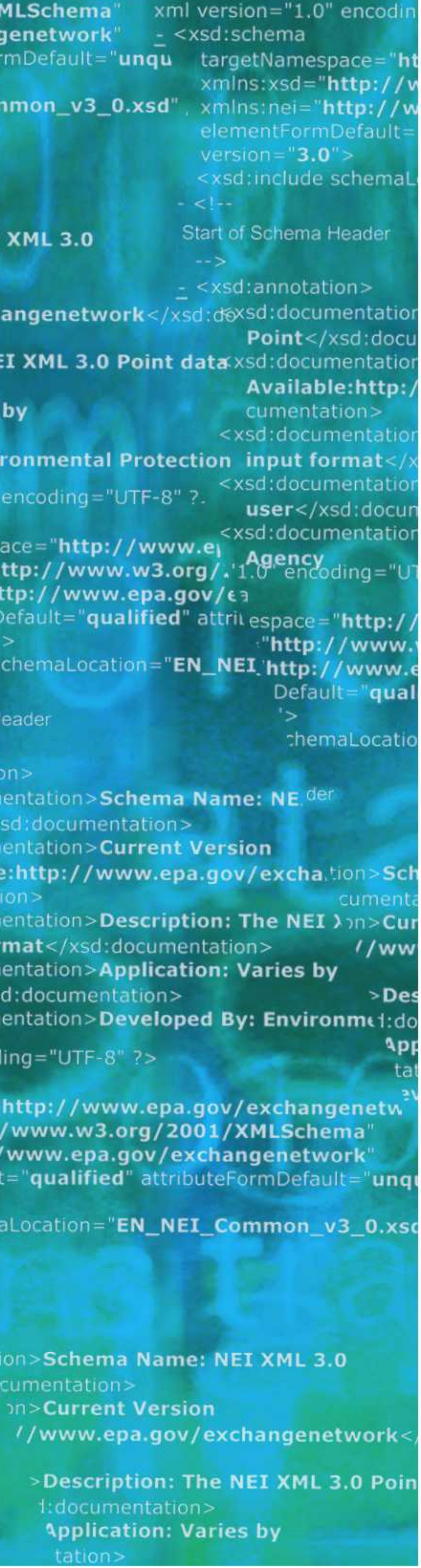


OpenNode2

FacID 3.0 Data Exchange Implementation Guide

Revision Date: 7/9/2014

Prepared By:

WINDSOR SOLUTIONS

4386 SW Macadam Ave, Suite 101
Portland, OR 97239
(503) 675-7833

Environmental Information Exchange Network

Revision History

Date	Author	Changes	Version
4/13/2010	Windsor	Initial version (unreleased)	1.0
2/2/2011	Windsor	Revision for OpenNode2 v1.2	2.0
5/7/2012	Windsor	Updated to correct minor documentation issues.	2.1
9/10/2012	Windsor	Added ER Diagram to Appendix A	2.2
6/10/2013	Windsor	Added section detailing use of polygon data	2.3
7/9/2014	Windsor	Updated Install Plugin section to describe pre-bundled plugin process starting with OpenNode2 v2.6	2.4

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

Data Exchange Overview

The purpose of this document is to provide detailed instructions for the installation and configuration of the Facility Identification (FacID) data exchange on the Microsoft .NET implementation of the Exchange Network OpenNode2 (OpenNode2).

The FacID data exchange offers seven data services that expose facility and environmental interest data to Exchange Network partners through Query and Solicit primitive commands:

- **GetFacilityCount_v3.0:** Provides a simple count of all facilities meeting entered criteria.
- **GetFacilityList_v3.0:** Provides a listing of facility information with a minimum number of details for each facility (useful for large listings of facilities).
- **GetFacility_v3.0:** Provides a fully-detailed listing of facilities and their environmental interests matching entered criteria.
- **GetFacilityInterest_v3.0:** Provides a listing of facility information and their associated environmental interests with few other details.
- **GetFacilityByID_v3.0:** Provides information on a single facility when querying by facility ID.
- **GetFacilityByChangeDate_v3.0:** Show all facilities modified after a given change date (useful for data synchronization processes).
- **GetDeletedFacilityByChangeDate_v3.0:** Show all facilities deleted after a given date (useful for data synchronization processes).

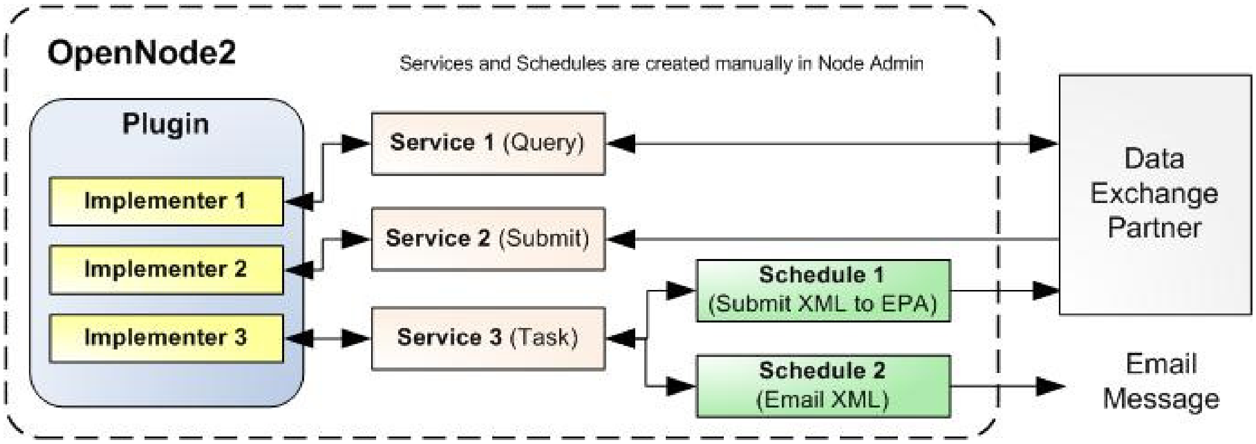
Further detail about the supported FacID data services is available in the Flow Configuration Document published at exchangenetwork.net.

The FacID data exchange services may be exposed to Exchange Network partners, allowing each to obtain data from your facility data repository. The Environmental Protection Agency (EPA) Central Data Exchange (CDX) Node will also periodically call these services to populate the EPA Facility Registry System (FRS) data warehouse.

The FacID data exchange configuration process involves two main steps: 1) create and populate the FacID staging tables and 2) install and configure the FacID data flow. The rest of this document will describe these two processes in detail.

OpenNode2 Plugin Architecture

The diagram below shows the architecture of a typical OpenNode2 plugin and how services that access the plugin's functionality are configured by a Node Administrator.



A plugin contains one or more **implementers**. Implementers are specific functional components that are specific to the data exchange. An implementer performs a specific task, for example, composing XML from a series of staging tables.

A Node Administrator makes available the functionality in an implementer by creating **services**. When a service is created, an implementer must be chosen. Each service may have one or more configuration arguments, defined by the implementer. For example, the service may require that a database connection or Node partner URL be provided. Services can be made available to external partners in the form of a query or solicit or as an inbound submission processor. "Task" services are internal only and are accessed via a **schedule**. Schedules also can have configuration arguments which are required by the data service assigned to the schedule.

Query/Solicit Implementers

There are seven implementers in the FacID plugin that expose Query and/or Solicit services. These services can be called by external partners or invoked by a schedule.

The seven implementers are:

- GetFacilityCount
- GetFacilityList
- GetFacility
- GetFacilityInterest
- GetFacilityByID
- GetFacilityByChangeDate
- GetDeletedFacilityByChangeDate

All seven implementers support the same arguments and are described below:

Service Parameters:

Add Exchange Header: Type either true or false. Indicates whether to include the standard Exchange Network Header in the XML generated by the implementer. Note that the EPA FacID backend processor is currently unable to accept the standard header document format used by EPA CDX and other backend EPA systems.

Author: The value to insert in the Author element in the Exchange Network Header document.

Contact Info: The value to insert in the Contact Info element in the Exchange Network Header document.

Organization: The value to insert in the Organization element in the Exchange Network Header document.

Payload Operation: The value to insert into the Payload Operation attribute of the Exchange Network Header document.

Result Cache Duration (d.hh:mm:ss): Enter a value corresponding to the amount of time the FacID processor should cache results before refreshing. Set this value to “0.00:00:00” to disable the cache.

Data Source: Set the Data Source to the data source that connects to the FacID staging tables.

Schedule Parameters:

The schedule parameters supported by the implementer are the same as are defined in the FacID 3.0 FCD Query and Solicit services. Please refer to the FCD for descriptions of each parameter, although the names are self explanatory. The corresponding staging table and field name used for the filter is listed below.

Standard Environmental Interest Type: Table FACID_ENVR_INTR column ENVR_INTR_TYPE_TEXT.

ZIP Code: Table FACID_FAC column ADDR_POST_CODE_VAL (starts with)

Tribal Land Code: Table FACID_FAC column TRIB_LAND_INDI

Federal Facility: Table FACID_FAC column FED_FAC_INDI

Facility Name: Table FACID_FAC column FAC_NAME (contains)

Facility Status: Table FACID_FAC column FAC_ACTIVE_INDI

SIC Code: Table FACID_FAC_FAC_SIC column SIC_CODE (starts with)

NAICS Code: Table FACID_FAC_FAC_NAICS column NAICS_CODE (starts with)

City Name: Table FACID_FAC column LOCA_NAME

State: Table FACID_FAC column STA_CODE

County Name: Table FACID_FAC column CNTY_NAME

N Bounding Latitude: Table FACID_FAC_PRI_GEO_LOC_DESC column LATITUDE (Less than or equal to)

S Bounding Latitude: Table FACID_FAC_PRI_GEO_LOC_DESC column

LATITUDE (Greater than or equal to)

E Bounding Longitude: Table FACID_FAC_PRI_GEO_LOC_DESC column LONGITUDE (Less than or equal to)

W Bounding Longitude: Table FACID_FAC_PRI_GEO_LOC_DESC column LONGITUDE (Greater than or equal to)

Change Date: Table FACID_FAC column LAST_UPDT_DATE (Greater than or equal to)

Facility Site Identifier: Table FACID_FAC column FAC_SITE_IDEN_VAL

Originating Partner Name: Table FACID_FAC column ORIG_PART_NAME

Information System Acronym Name: Table FACID_FAC column INFO_SYS_ACRO_NAME

Deleted Date: Table FACID_FAC column DELETED_ON_DATE (Greater than or equal to)

ValidateXML: Enter value of “True” or “False”

Other Implementers

SubmissionProcessor Implementer

This implementer is used to process XML data submitted to OpenNode2 by an external partner. The data will then be deserialized and inserted into the node staging database. This may useful for testing purposes or if a node needs to be configured to receive BEACHES data from some other partner.

To use this implementer, a service must be created with a name of * and a service type of Submit, and set for public access. When a file is submitted to the OpenNode2 endpoint with a dataflow of “FacID_v3.0”, this service will run automatically to process the file.

Since this implementer is not used in the typical flow of data to EPA CDX, this service is not described in the *Implementing the FacID Exchange* section below.

Service Parameters: **Delete Existing Data Before Insert (true or false)**: Setting this value to True will result in all staging table data being deleted before insertion of data from the XML file. Deletion of data will occur starting at the FACID_FAC_DTLS table and all child tables.

Data Source: Set the Data Source to the data source that connects to the FacID staging tables.

Schedule Parameters: None. This service should not be executed by a schedule.

ExtractXMLFilesFromDatabase

This implementer is used to extract data from the FacID staging tables, serialize the data to FacID XML files and save the files to a network location specified in the schedule used to execute the service.

This implementer can only be exposed as a Task to be run by the scheduler and is therefore not available to external node partners.

Since this implementer is not used in the typical flow of data to EPA CDX, this service is not described in the *Implementing the FacID Exchange* section below.

Service Parameters: See the service parameter descriptions for the Query/Solicit Implementers above.

Schedule Parameters: **XmlFolderPath**: The path to which the serialized XML data will be saved.

AddHeader: Set to true or false. Overrides the service parameter with the same name. Indicates whether or not to add a Header to the XML file.

LoadXMLFileIntoDatabase

Since this implementer is not used in the typical flow of data to EPA CDX, this service is not described in the *Implementing the FacID Exchange* section below.

This implementer is used to read a FacID XML file from a location on a local or network share and deserialized the payload into the FacID staging tables.

This implementer can only be exposed as a Task to be run by the scheduler and is therefore not available to external node partners.

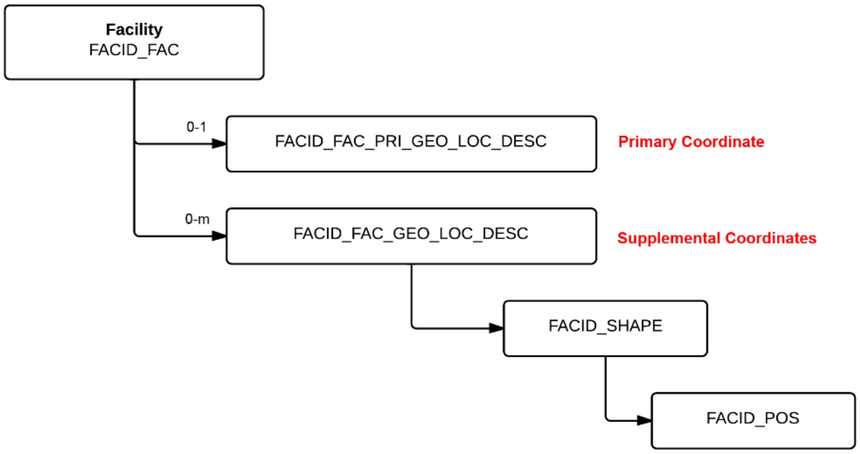
Service Parameters: **Data Source:** Set the Data Source to the data source that connects to the FacID staging tables.

Schedule Parameters: **XmlFolderPath**: The path from which the XML data will be read for loading into the staging database.

Facility Geographic Data

The FacID v3.0 XML schema allows for two sets of geographic data to be sent for each facility; the facility's primary location expressed as a single decimal latitude/longitude coordinate, and an optional list of additional geographic features associated with the facility. For simplicity, the first list is referred to as the **primary coordinate** and the second as **supplemental coordinates** in this document.

The following diagram illustrates the relationship of these two sets of information in the OpenNode2 FacID staging tables:



Primary and Suppliemental Coordinates in the FacID Staging Tables

Primary Coordinate

The primary coordinate relates to the FacilityPrimaryGeographicLocationDescription element in the FacID schema. The equivalent staging table is FACID_FAC_PRI_GEO_LOC_DESC. While the staging tables support multiple primary coordinates per facility, only one record should be associated with a facility.

The primary coordinate must be a geographic point, expressed as a decimal latitude/longitude pair. These values must be loaded into the LATITUDE and LONGITUDE fields respectively. The plugin will construct the correct GeoRSS XML representation of a Point element using this information.

Supplemental Coordinates

Supplemental coordinates relate to the FacilityGeographicLocationDescription element in the FacID XML schema. The equivalent staging table is FACID_FAC_GEO_LOC_DESC. The FacID XML schema allows for zero or more supplemental coordinates to be supplied per facility.

Supplemental coordinates can be expressed as a GML GeoRSS Point, LineString, Envelope, or Polygon shape. For more information on GeoRSS shape types, please see the GML and GeoRSS documentation.

If the supplemental coordinate is a Point data type, simply load the coordinate into the LATITUDE and LONGITUDE fields in the FACID_FAC_GEO_LOC_DESC staging table. However, if LineString, Envelope or Polygon data is to be sent, the FACID_SHAPE and FACID_POS staging tables must also be loaded.

In the FACID_SHAPE table, specify the TYPE as one of the supported shape types: “LineString”, “Envelope”, or “Polygon”. Then add the list of latitude/longitude coordinates into the FACID_POS table, being sure to set to ORDER_INDEX so that the latitude/longitude values are sequenced correctly in the resulting XML.

Implementing the FacID Exchange

Step 1: Create and Populate FacID Staging Tables

OpenNode2 uses a plugin-based architecture to support data exchanges with EPA and other Exchange Network partners. Data must first be loaded into a set of staging tables before it can be extracted by the plugin and shared through the FacID data exchange. This section outlines the steps required to set up the FacID data exchange database staging tables.

Establish Staging Database

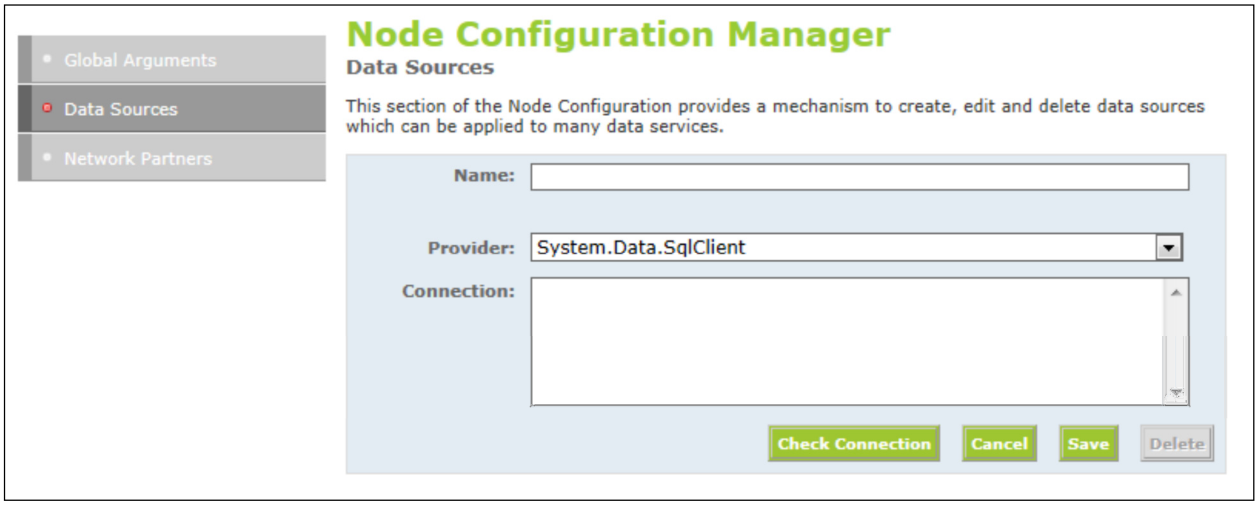
1. The first step is to create the staging database itself if one has not already been established to support another data exchange (typically named NODE_FLOW).
2. Once the staging database itself is created, a Database Definition Language (DDL) script included in the OpenNode2 deployment package can be executed to create the staging tables themselves that will be used to store the data being made available through the FacID data exchange.
3. With the staging environment established, data must now be mapped from the source database to the equivalent fields in the FacID staging tables. The staging tables closely reflect the structure and naming of the FacID XML schema, and it is recommended that the Data Exchange Template (DET) published at exchangenetwork.net be used to facilitate this mapping.

   Mapping Tip: Begin by inserting one row into the FACID_FAC_DTLS table. This is the root table used by the plugin from which all XML is generated. Two child tables, FACID_AFFL and FACID_FAC, must reference this ID in order for any XML to be generated by the plugin.
4. Once the mapping is complete, a database routine should be developed to populate the tables in the staging database using the mapping prepared during the earlier step. This should be a repeatable process that will empty and replace all of the data in the staging tables, or a procedure that will incrementally add, update and remove data as it changes in the source system.
5. Once the data extract process has been developed, it should be automated to execute on a regular schedule as appropriate to the needs of the organization for submissions to EPA.

Define OpenNode2 Data Source

Once the staging database has been established, a data connection must be created in the Node Admin to allow the OpenNode2 data services to access the database.

1. After logging into the Node Admin, click the **Configuration** tab on the top navigation bar.
2. From the **Configuration** tab, click the **Data Sources** button on the left navigation bar.
3. Click the **Add Data Source** button. The Data Source Manager screen will be displayed.
4. Type the desired name of the data source in the **Name** field.
5. Select the appropriate database provider from the **Provider** dropdown list.
6. Type the appropriate database connection string information in the **Connection** field.
7. Click **Check Connection** to confirm the configuration information.
8. Click **Save** to save the data source.



Node Configuration Manager

Data Sources

This section of the Node Configuration provides a mechanism to create, edit and delete data sources which can be applied to many data services.

Name:

Provider: System.Data.SqlClient

Connection:

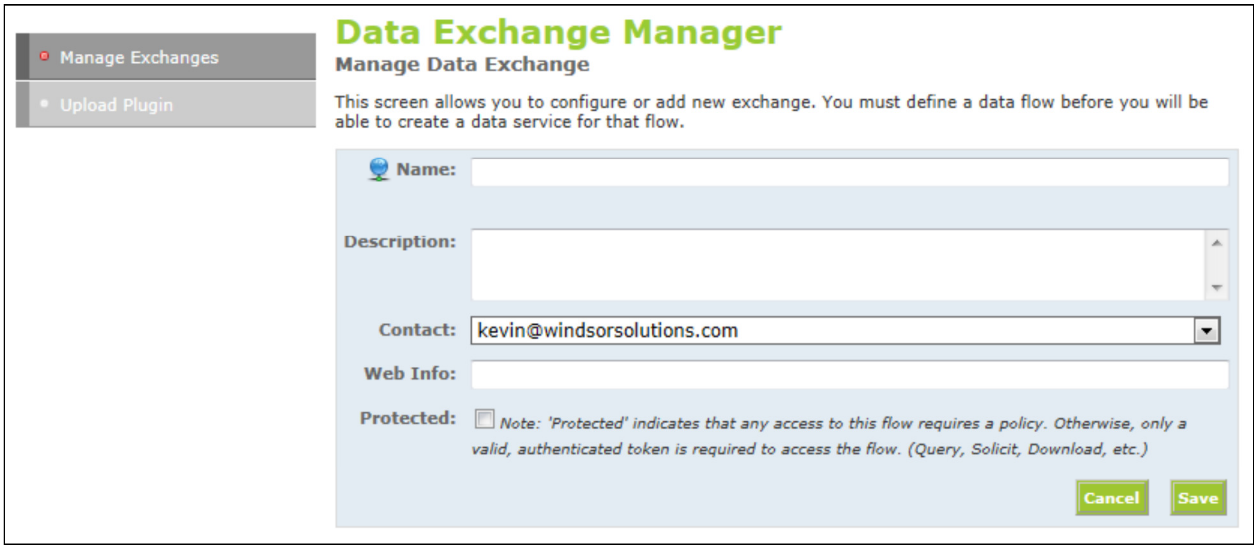
Step 2: Install and Configure the FacID Data Exchange

This section describes the steps required to install and configure the FacID data exchange on the Microsoft .NET implementation of the OpenNode2 using the OpenNode2 Administration Web application (Node Admin).

Create FacID Data Exchange

The first step to implement the FacID data exchange on the OpenNode2 is to create the data exchange using the Node Admin Data Exchange Manager.

1. After logging into the Node Admin, click the **Exchange** tab on the top navigation bar.
2. Click the **Add Exchange** button. The Data Exchange Manager screen will be displayed:



Data Exchange Manager

Manage Data Exchange

This screen allows you to configure or add new exchange. You must define a data flow before you will be able to create a data service for that flow.

Name:

Description:

Contact: kevin@windsorsolutions.com

Web Info:

Protected: ☐ *Note: 'Protected' indicates that any access to this flow requires a policy. Otherwise, only a valid, authenticated token is required to access the flow. (Query, Solicit, Download, etc.)*

3. Type *FacID_v3.0* in the **Name** field.
4. Type a short description in the **Description** field, e.g., *Facility Identification data exchange*.
5. Select a user account name from the **Contact** drop-down menu. Contacts are populated with all accounts that have been set up on the OpenNode2. See the **Security** tab for a list of available accounts.
6. Type any valid URL in the **Web Info** field. Ideally, this will be the page on the Exchange Network Web site that describes the FacID data exchange:

   http://www.exchangenetwork.net/exchanges/cross/facility.htm
7. It is recommended that the **Protected** checkbox remain unchecked. This will enable all authenticated Node users to access the FacID data services without needing special data exchange specific security permissions.
8. Click **Save** to save the data exchange.

Install FacID Plugin

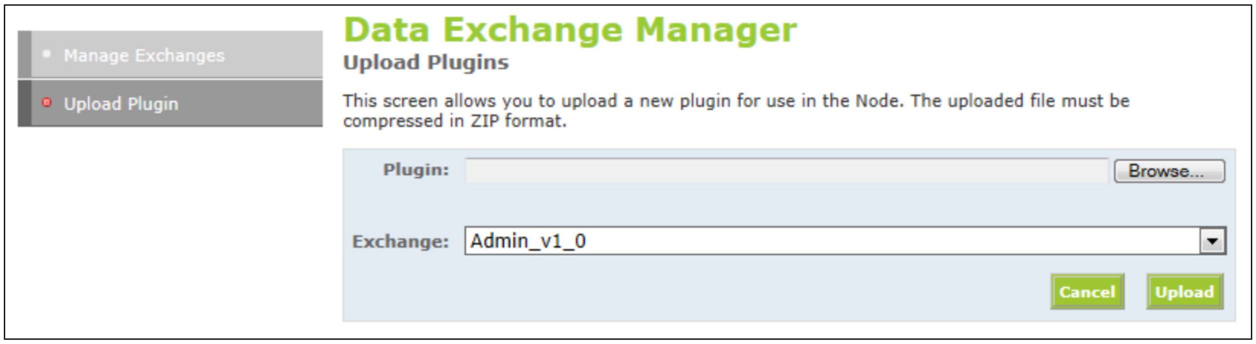
Once the FacID data exchange has been created, the next step is to upload the FacID plugin provided by Windsor into the OpenNode2 plugin repository.

Note: If you are using OpenNode2 v2.6 or higher, this step is not necessary. Starting with v2.6, all plugins are pre-installed with the OpenNode2 software installation package. By creating the exchange above, the plugin will automatically be loaded and associated with the exchange. To validate that the plugin was installed automatically, follow the steps below:

1. From the **Exchange** tab, scroll down the list of installed data exchanges until the WQX exchange is located.
2. Click the **Add Service** button located just beneath the WQX data exchange record. If the Implementer drop down box is not empty, then the plugin has been installed successfully.

If the steps above reveal that the plugin is not installed, perform the following steps to install it.

1. Navigate to the plugin directory in the **Plugins\[Flow Name]\[version number]** directory included with the OpenNode2 installation files.
2. Create a new zip file containing the two Windsor.Node2008.WNOSPlugin.[Flow name].dll and .pdb files.
3. From the **Exchange** tab, click the **Upload Plugin** button on the left side navigation bar.



Data Exchange Manager

Upload Plugins

This screen allows you to upload a new plugin for use in the Node. The uploaded file must be compressed in ZIP format.

Exchange: Admin_v1_0

4. Click the **Browse** button.
5. Locate and select the compressed (zipped) file containing the code component for the FacID plugin that you created in step 2 above.
6. Select *FacID_v3.0* from the **Exchange** drop-down menu. If this data exchange name is not available, ensure that the previous step was completed.
7. Click the **Upload** button to install the FacID plugin.

The newly uploaded plugin code will be placed in the OpenNode2 plugin repository. Any previous plugin versions will be retained in the repository but won't be accessible through the Node Admin. Only the latest version of any one plugin is made available during the next step to establish data services.

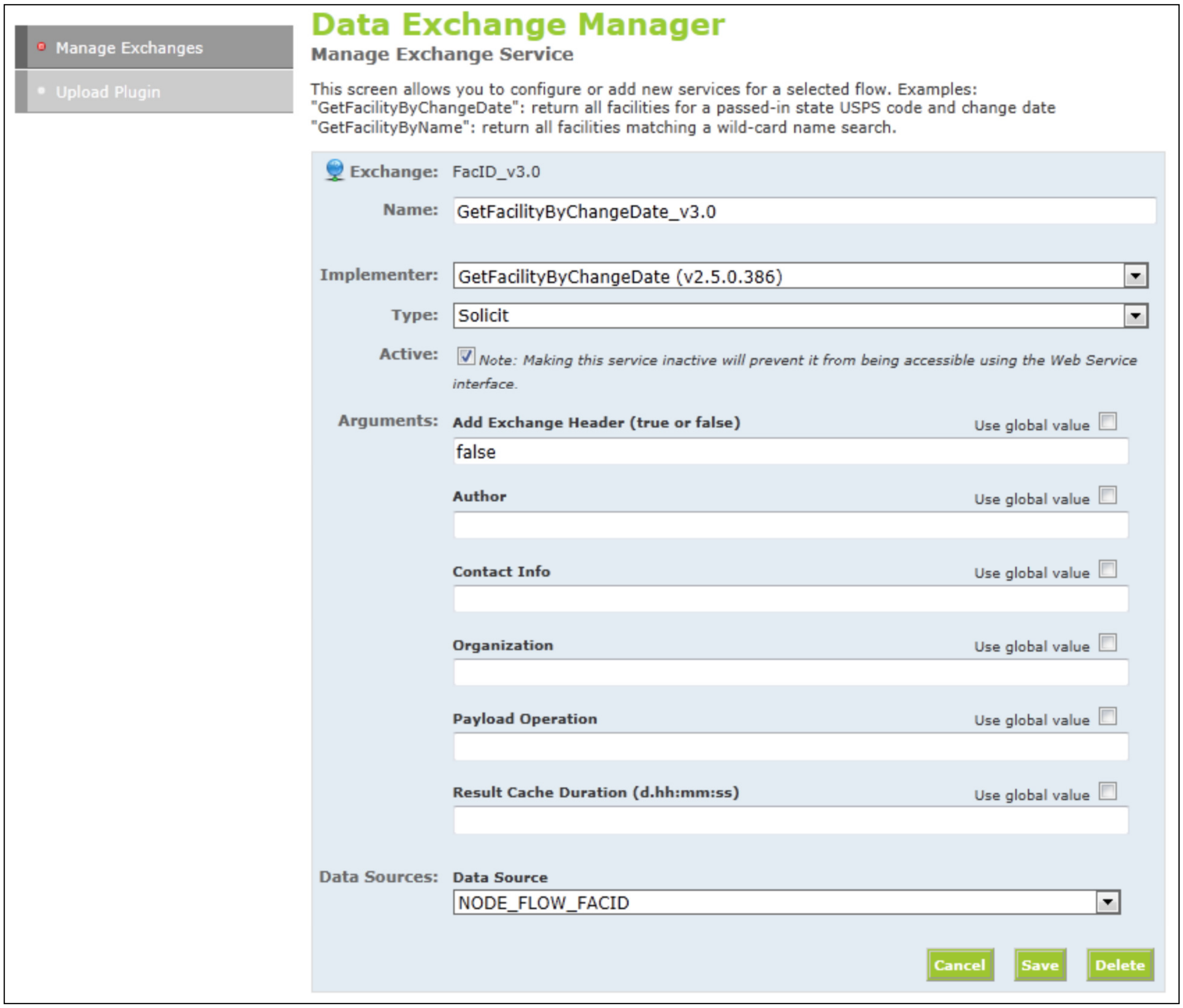
Step 3: Create FacID Data Services

Data services are distinct functions provided by a plugin to support a given data exchange. For the FacID data exchange, there are seven specific data services provided by the plugin:

- GetFacilityCount_v3.0
- GetFacilityList_v3.0
- GetFacility_v3.0
- GetFacilityInterest_v3.0
- GetFacilityByID_v3.0
- GetFacilityByChangeDate_v3.0
- GetDeletedFacilityByChangeDate_v3.0

Each of these data services must be created and configured before they can be accessed through the OpenNode2 endpoints. The following instructions should be repeated for each of the data services that the organization wishes to make available.

1. From the **Exchange** tab, locate the FacID data exchange in the list of available exchanges.
2. Click the **Add Service** button located just beneath the FacID data exchange entry. The following page will be displayed to allow a new data service to be added.



Data Exchange Manager

Manage Exchange Service

This screen allows you to configure or add new services for a selected flow. Examples:
"GetFacilityByChangeDate": return all facilities for a passed-in state USPS code and change date
"GetFacilityByName": return all facilities matching a wild-card name search.

Exchange: FacID_v3.0

Name: GetFacilityByChangeDate_v3.0

Implementer: GetFacilityByChangeDate (v2.5.0.386)

Type: Solicit

Active: ☑ *Note: Making this service inactive will prevent it from being accessible using the Web Service interface.*

Arguments:

Add Exchange Header (true or false) Use global value □
false

Author Use global value □

Contact Info Use global value □

Organization Use global value □

Payload Operation Use global value □

Result Cache Duration (d.hh:mm:ss) Use global value □

Data Sources: **Data Source**
NODE_FLOW_FACID

3. In the **Name** field, enter the name of the service being configured (see above).
4. From the **Implementer** drop down box, select the appropriate value for this data service. This implementer provides the functionality for all of the FacID data services. The following table specifies the relevant implementer for each of the data services to be configured:

Service	Plugin
GetFacilityCount_v3.0	GetFacilityCount
GetFacilityList_v3.0	GetFacilityList
GetFacility_v3.0	GetFacility
GetFacilityInterest_v3.0	GetFacilityInterest
GetFacilityByID_v3.0	GetFacilityByID
GetFacilityByChangeDate_v3.0	GetFacilityByChangeDate
GetDeletedFacilityByChangeDate_v3.0	GetDeletedFacilityByChangeDate

5. From the **Type** drop-down menu, select how you wish to make the services available. The options available will also be obtained by the Node Admin from the plugin. It is recommended that the FacID data services allow *Query or Solicit*.
6. Enable the service by checking the **Active** checkbox.
7. Based on the selection made from the implementer drop-down menu, the Node Admin will determine what argument and data source requirements the plugin has and will refresh the page to display the relevant data entry fields as follows.
8. Enter the service parameter settings as follows. Note that these settings relate to the optional inclusion of the Exchange Network Header document on the generated XML files. The EPA FacID processor is currently unable to accept the standard header document format used by EPA CDX and other EPA systems, and the header document is not required by the FacID FCD.

   i. In the argument labeled **Add Exchange Header (true or false)**, enter *false*.

   ii. In the argument labeled **Author**, type the name of the developer of the data service.

   iii. In the argument labeled **Contact Info**, type the name of the person who should be contacted regarding any submission created from the data service. Also include the person's email address and phone number. For example, enter *John Smith, (999) 999-9999, john@smith.com*, etc.

   Alternatively if a global variable has been set up to provide this value, check the **Use global value** checkbox and select the appropriate variable name from the drop down box that appears in place f the textbox.

   iv. In the argument labeled **Organization**, type the name of the organization that is providing submissions created from the data service. For example, enter *Smith, Inc.*, etc.

   Alternatively if a global variable has been set up to provide this value, check the **Use global value** checkbox and select the appropriate variable name from the drop down box that appears in place of the textbox.

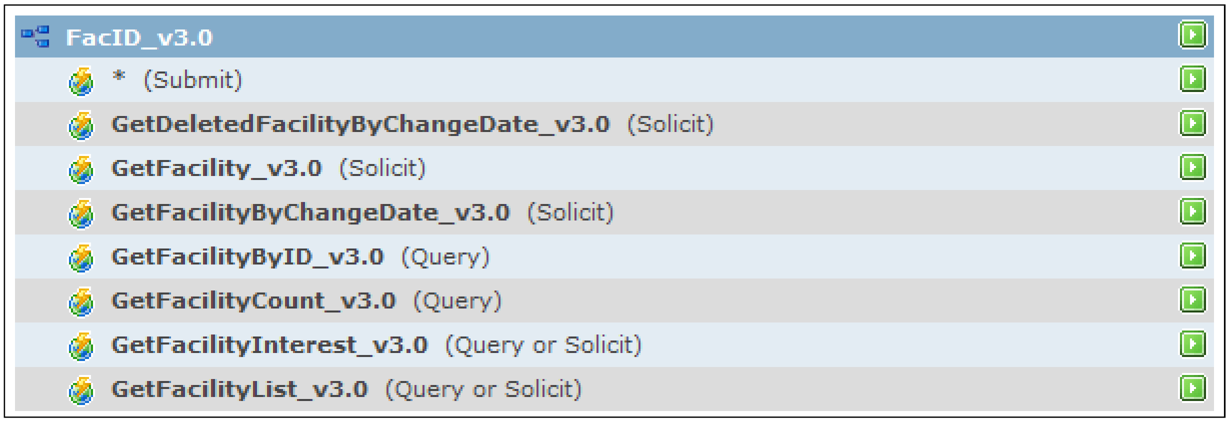
   v. In the argument labeled **Payload Operation**, leave blank.

   vi. In the argument labeled **Result Cache Duration (d.hh:mm:ss)**, enter a value corresponding to the amount of time the FacID processor should cache results before refreshing. Set this value to "0.00:00:00" to disable the cache.

9. Set the Data Source to the data source that connects to the FacID staging tables. Reference the Creating the FacID Staging Tables section in this document for further details.
10. Click the **Save** button to save the data service.

Repeat these steps for the remaining data services.

The **Manage Exchanges** page for the FacID_v3.0 data exchange should now appear as follows:



FacID_v3.0

* (Submit)

GetDeletedFacilityByChangeDate_v3.0 (Solicit)

GetFacility_v3.0 (Solicit)

GetFacilityByChangeDate_v3.0 (Solicit)

GetFacilityByID_v3.0 (Query)

GetFacilityCount_v3.0 (Query)

GetFacilityInterest_v3.0 (Query or Solicit)

GetFacilityList_v3.0 (Query or Solicit)

Step 4: Define Data Exchange Schedules

Scheduled jobs can be configured in the OpenNode2 to perform automated tasks, for example, submitting data to external Exchange Network partners or processing received files. EPA CDX node can set up a solicit schedule to retrieve FacID data from State and Tribal production nodes. For testing, EPA has indicated that FacID data can be submitted to the CDX test node. If a state or tribe wishes to test using this method, the data exchange must be named "FacID_v3.0" and the submit operation for 2.0 endpoints must be set to "ProcessFRSDoc" in the OpenNode2 schedule's result process section.

Please see the OpenNode2 Administration User Guide for more information on scheduling data exchanges.

Step 5: Contact CDX to Establish Data Exchange Settings

Once the FacID data exchange is installed and configured, inform the EPA CDX Node Helpdesk that the FacID flow is available. You will need to provide your OpenNode2 endpoint URL and establish a process to update the FacID staging tables. Reference the Creating the FacID Staging Tables section in this document for more information.

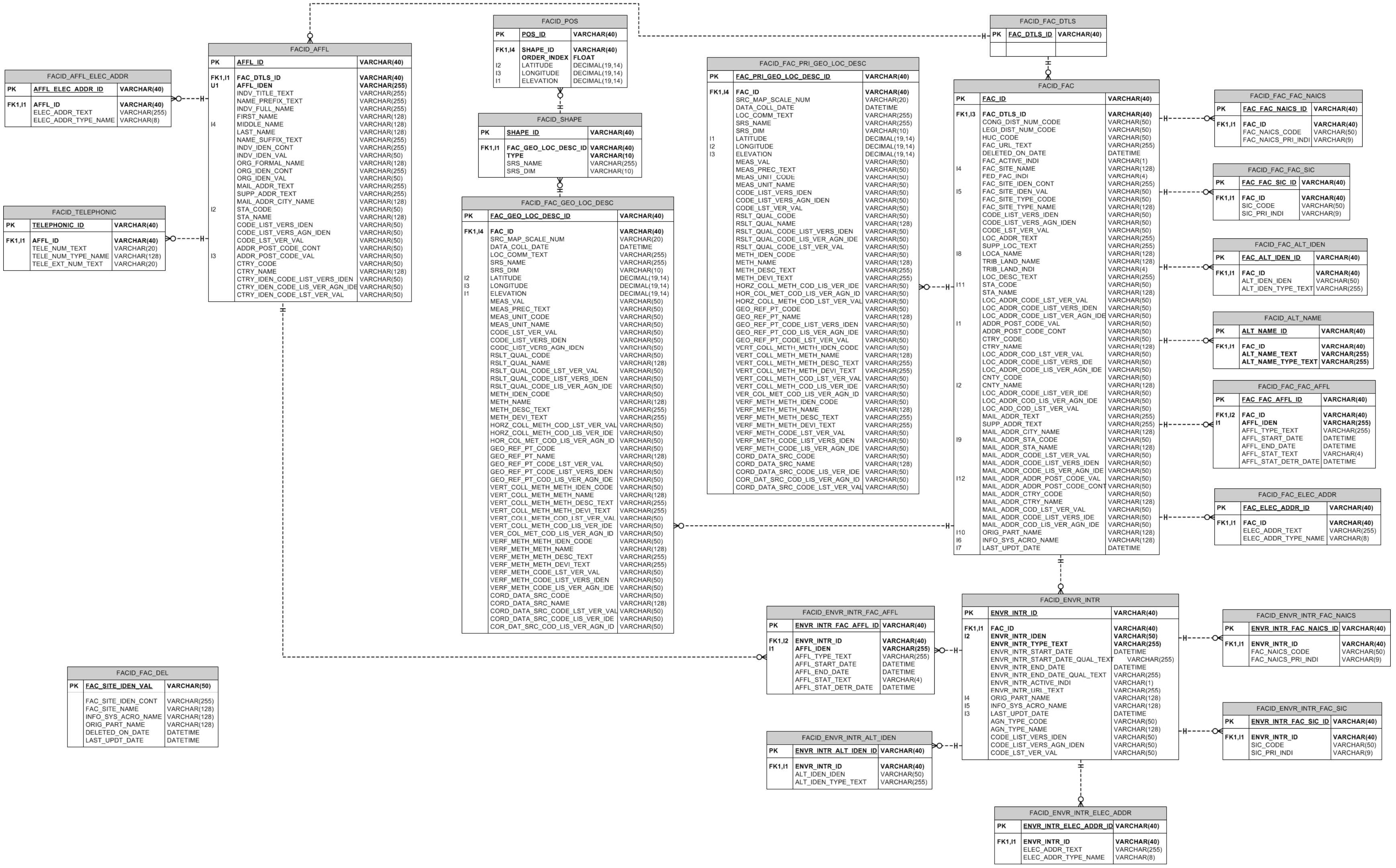
Step 6: Establish Email Notifications

If desired, using the Node Admin, a Node administrator may create NAAS accounts for one or more users and set up email notifications for the any OpenNode2 events related to the FacID data exchange. Please see the OpenNode2 Administration User Guide for more information on creating data exchange email notifications.

Step 7: Monitor Flow Activity

The OpenNode2 will track all FacID data exchange activity and can be accessed to monitor and debug related flow activities. Please see the OpenNode2 Administration User Guide for more information on accessing and searching the available OpenNode2 activity reports.

Appendix A – Staging Table ER Diagram



FACID_AFFL_ELEC_ADDR		
PK	**AFFL_ELEC_ADDR_ID**	**VARCHAR(40)**
FK1,I1	**AFFL_ID**	**VARCHAR(40)**
	ELEC_ADDR_TEXT	VARCHAR(255)
	ELEC_ADDR_TYPE_NAME	VARCHAR(8)

FACID_TELEPHONIC		
PK	**TELEPHONIC_ID**	**VARCHAR(40)**
FK1,I1	**AFFL_ID**	**VARCHAR(40)**
	TELE_NUM_TEXT	VARCHAR(20)
	TELE_NUM_TYPE_NAME	VARCHAR(128)
	TELE_EXT_NUM_TEXT	VARCHAR(20)

FACID_FAC_DEL		
PK	**FAC_SITE_IDEN_VAL**	**VARCHAR(50)**
	FAC_SITE_IDEN_CONT	VARCHAR(255)
	FAC_SITE_NAME	VARCHAR(128)
	INFO_SYS_ACRO_NAME	VARCHAR(128)
	ORIG_PART_NAME	VARCHAR(128)
	DELETED_ON_DATE	DATETIME
	LAST_UPDT_DATE	DATETIME

FACID_AFFL		
PK	**AFFL_ID**	**VARCHAR(40)**
FK1,I1	**FAC_DTLS_ID**	**VARCHAR(40)**
U1	**AFFL_IDEN**	**VARCHAR(255)**
	INDV_TITLE_TEXT	VARCHAR(255)
	NAME_PREFIX_TEXT	VARCHAR(255)
	INDV_FULL_NAME	VARCHAR(255)
	FIRST_NAME	VARCHAR(128)
I4	MIDDLE_NAME	VARCHAR(128)
	LAST_NAME	VARCHAR(128)
	NAME_SUFFIX_TEXT	VARCHAR(255)
	INDV_IDEN_CONT	VARCHAR(255)
	INDV_IDEN_VAL	VARCHAR(50)
	ORG_FORMAL_NAME	VARCHAR(128)
	ORG_IDEN_CONT	VARCHAR(255)
	ORG_IDEN_VAL	VARCHAR(50)
	MAIL_ADDR_TEXT	VARCHAR(255)
	SUPP_ADDR_TEXT	VARCHAR(255)
	MAIL_ADDR_CITY_NAME	VARCHAR(128)
I2	STA_CODE	VARCHAR(50)
	STA_NAME	VARCHAR(128)
	CODE_LIST_VERS_IDEN	VARCHAR(50)
	CODE_LIST_VERS_AGN_IDEN	VARCHAR(50)
	CODE_LST_VER_VAL	VARCHAR(50)
	ADDR_POST_CODE_CONT	VARCHAR(50)
I3	ADDR_POST_CODE_VAL	VARCHAR(50)
	CTRY_CODE	VARCHAR(50)
	CTRY_NAME	VARCHAR(128)
	CTRY_IDEN_CODE_LIST_VERS_IDEN	VARCHAR(50)
	CTRY_IDEN_CODE_LIS_VER_AGN_IDE	VARCHAR(50)
	CTRY_IDEN_CODE_LST_VER_VAL	VARCHAR(50)

FACID_POS		
PK	**POS_ID**	**VARCHAR(40)**
FK1,I4	**SHAPE_ID**	**VARCHAR(40)**
	ORDER_INDEX	**FLOAT**
I2	LATITUDE	DECIMAL(19,14)
I3	LONGITUDE	DECIMAL(19,14)
I1	ELEVATION	DECIMAL(19,14)

FACID_SHAPE		
PK	**SHAPE_ID**	**VARCHAR(40)**
FK1,I1	**FAC_GEO_LOC_DESC_ID**	**VARCHAR(40)**
	TYPE	**VARCHAR(10)**
	SRS_NAME	VARCHAR(255)
	SRS_DIM	VARCHAR(10)

FACID_FAC_GEO_LOC_DESC		
PK	**FAC_GEO_LOC_DESC_ID**	**VARCHAR(40)**
FK1,I4	**FAC_ID**	**VARCHAR(40)**
	SRC_MAP_SCALE_NUM	VARCHAR(20)
	DATA_COLL_DATE	DATETIME
	LOC_COMM_TEXT	VARCHAR(255)
	SRS_NAME	VARCHAR(255)
	SRS_DIM	VARCHAR(10)
I2	LATITUDE	DECIMAL(19,14)
I3	LONGITUDE	DECIMAL(19,14)
I1	ELEVATION	DECIMAL(19,14)
	MEAS_VAL	VARCHAR(50)
	MEAS_PREC_TEXT	VARCHAR(50)
	MEAS_UNIT_CODE	VARCHAR(50)
	MEAS_UNIT_NAME	VARCHAR(50)
	CODE_LST_VER_VAL	VARCHAR(50)
	CODE_LIST_VERS_IDEN	VARCHAR(50)
	CODE_LIST_VERS_AGN_IDEN	VARCHAR(50)
	RSLT_QUAL_CODE	VARCHAR(50)
	RSLT_QUAL_NAME	VARCHAR(128)
	RSLT_QUAL_CODE_LST_VER_VAL	VARCHAR(50)
	RSLT_QUAL_CODE_LIST_VERS_IDEN	VARCHAR(50)
	RSLT_QUAL_CODE_LIS_VER_AGN_IDE	VARCHAR(50)
	METH_IDEN_CODE	VARCHAR(50)
	METH_NAME	VARCHAR(128)
	METH_DESC_TEXT	VARCHAR(255)
	METH_DEVI_TEXT	VARCHAR(255)
	HORZ_COLL_METH_COD_LST_VER_VAL	VARCHAR(50)
	HORZ_COLL_METH_COD_LIS_VER_IDE	VARCHAR(50)
	HOR_COL_MET_COD_LIS_VER_AGN_ID	VARCHAR(50)
	GEO_REF_PT_CODE	VARCHAR(50)
	GEO_REF_PT_NAME	VARCHAR(128)
	GEO_REF_PT_CODE_LST_VER_VAL	VARCHAR(50)
	GEO_REF_PT_CODE_LIST_VERS_IDEN	VARCHAR(50)
	GEO_REF_PT_COD_LIS_VER_AGN_IDE	VARCHAR(50)
	VERT_COLL_METH_METH_IDEN_CODE	VARCHAR(50)
	VERT_COLL_METH_METH_NAME	VARCHAR(128)
	VERT_COLL_METH_METH_DESC_TEXT	VARCHAR(255)
	VERT_COLL_METH_METH_DEVI_TEXT	VARCHAR(255)
	VERT_COLL_METH_COD_LST_VER_VAL	VARCHAR(50)
	VERT_COLL_METH_COD_LIS_VER_IDE	VARCHAR(50)
	VER_COL_MET_COD_LIS_VER_AGN_ID	VARCHAR(50)
	VERF_METH_METH_IDEN_CODE	VARCHAR(50)
	VERF_METH_METH_NAME	VARCHAR(128)
	VERF_METH_METH_DESC_TEXT	VARCHAR(255)
	VERF_METH_METH_DEVI_TEXT	VARCHAR(255)
	VERF_METH_CODE_LST_VER_VAL	VARCHAR(50)
	VERF_METH_CODE_LIST_VERS_IDEN	VARCHAR(50)
	VERF_METH_CODE_LIS_VER_AGN_IDE	VARCHAR(50)
	CORD_DATA_SRC_CODE	VARCHAR(50)
	CORD_DATA_SRC_NAME	VARCHAR(128)
	CORD_DATA_SRC_CODE_LST_VER_VAL	VARCHAR(50)
	CORD_DATA_SRC_CODE_LIS_VER_IDE	VARCHAR(50)
	COR_DAT_SRC_COD_LIS_VER_AGN_ID	VARCHAR(50)

FACID_FAC_PRI_GEO_LOC_DESC		
PK	**FAC_PRI_GEO_LOC_DESC_ID**	**VARCHAR(40)**
FK1,I4	**FAC_ID**	**VARCHAR(40)**
	SRC_MAP_SCALE_NUM	VARCHAR(20)
	DATA_COLL_DATE	DATETIME
	LOC_COMM_TEXT	VARCHAR(255)
	SRS_NAME	VARCHAR(255)
	SRS_DIM	VARCHAR(10)
I1	LATITUDE	DECIMAL(19,14)
I2	LONGITUDE	DECIMAL(19,14)
I3	ELEVATION	DECIMAL(19,14)
	MEAS_VAL	VARCHAR(50)
	MEAS_PREC_TEXT	VARCHAR(50)
	MEAS_UNIT_CODE	VARCHAR(50)
	MEAS_UNIT_NAME	VARCHAR(50)
	CODE_LIST_VERS_IDEN	VARCHAR(50)
	CODE_LIST_VERS_AGN_IDEN	VARCHAR(50)
	CODE_LST_VER_VAL	VARCHAR(50)
	RSLT_QUAL_CODE	VARCHAR(50)
	RSLT_QUAL_NAME	VARCHAR(128)
	RSLT_QUAL_CODE_LIST_VERS_IDEN	VARCHAR(50)
	RSLT_QUAL_CODE_LIS_VER_AGN_IDE	VARCHAR(50)
	RSLT_QUAL_CODE_LST_VER_VAL	VARCHAR(50)
	METH_IDEN_CODE	VARCHAR(50)
	METH_NAME	VARCHAR(128)
	METH_DESC_TEXT	VARCHAR(255)
	METH_DEVI_TEXT	VARCHAR(255)
	HORZ_COLL_METH_COD_LIS_VER_IDE	VARCHAR(50)
	HOR_COL_MET_COD_LIS_VER_AGN_ID	VARCHAR(50)
	HORZ_COLL_METH_COD_LST_VER_VAL	VARCHAR(50)
	GEO_REF_PT_CODE	VARCHAR(50)
	GEO_REF_PT_NAME	VARCHAR(128)
	GEO_REF_PT_CODE_LIST_VERS_IDEN	VARCHAR(50)
	GEO_REF_PT_COD_LIS_VER_AGN_IDE	VARCHAR(50)
	GEO_REF_PT_CODE_LST_VER_VAL	VARCHAR(50)
	VERT_COLL_METH_METH_IDEN_CODE	VARCHAR(50)
	VERT_COLL_METH_METH_NAME	VARCHAR(128)
	VERT_COLL_METH_METH_DESC_TEXT	VARCHAR(255)
	VERT_COLL_METH_METH_DEVI_TEXT	VARCHAR(255)
	VERT_COLL_METH_COD_LST_VER_VAL	VARCHAR(50)
	VERT_COLL_METH_COD_LIS_VER_IDE	VARCHAR(50)
	VER_COL_MET_COD_LIS_VER_AGN_ID	VARCHAR(50)
	VERF_METH_METH_IDEN_CODE	VARCHAR(50)
	VERF_METH_METH_NAME	VARCHAR(128)
	VERF_METH_METH_DESC_TEXT	VARCHAR(255)
	VERF_METH_METH_DEVI_TEXT	VARCHAR(255)
	VERF_METH_CODE_LST_VER_VAL	VARCHAR(50)
	VERF_METH_CODE_LIST_VERS_IDEN	VARCHAR(50)
	VERF_METH_CODE_LIS_VER_AGN_IDE	VARCHAR(50)
	CORD_DATA_SRC_CODE	VARCHAR(50)
	CORD_DATA_SRC_NAME	VARCHAR(128)
	CORD_DATA_SRC_CODE_LIS_VER_IDE	VARCHAR(50)
	COR_DAT_SRC_COD_LIS_VER_AGN_ID	VARCHAR(50)
	CORD_DATA_SRC_CODE_LST_VER_VAL	VARCHAR(50)

FACID_FAC_DTLS		
PK	**FAC_DTLS_ID**	**VARCHAR(40)**

FACID_FAC		
PK	**FAC_ID**	**VARCHAR(40)**
FK1,I3	**FAC_DTLS_ID**	**VARCHAR(40)**
	CONG_DIST_NUM_CODE	VARCHAR(50)
	LEGI_DIST_NUM_CODE	VARCHAR(50)
	HUC_CODE	VARCHAR(50)
	FAC_URL_TEXT	VARCHAR(255)
	DELETED_ON_DATE	DATETIME
	FAC_ACTIVE_INDI	VARCHAR(1)
I4	FAC_SITE_NAME	VARCHAR(128)
	FED_FAC_INDI	VARCHAR(4)
	FAC_SITE_IDEN_CONT	VARCHAR(255)
I5	FAC_SITE_IDEN_VAL	VARCHAR(50)
	FAC_SITE_TYPE_CODE	VARCHAR(50)
	FAC_SITE_TYPE_NAME	VARCHAR(128)
	CODE_LIST_VERS_IDEN	VARCHAR(50)
	CODE_LIST_VERS_AGN_IDEN	VARCHAR(50)
	CODE_LST_VER_VAL	VARCHAR(50)
	LOC_ADDR_TEXT	VARCHAR(255)
	SUPP_LOC_TEXT	VARCHAR(255)
I8	LOCA_NAME	VARCHAR(128)
	TRIB_LAND_NAME	VARCHAR(128)
	TRIB_LAND_INDI	VARCHAR(4)
	LOC_DESC_TEXT	VARCHAR(255)
I11	STA_CODE	VARCHAR(50)
	STA_NAME	VARCHAR(128)
	LOC_ADDR_CODE_LST_VER_VAL	VARCHAR(50)
	LOC_ADDR_CODE_LIST_VERS_IDEN	VARCHAR(50)
	LOC_ADDR_CODE_LIST_VER_AGN_IDE	VARCHAR(50)
I1	ADDR_POST_CODE_VAL	VARCHAR(50)
	ADDR_POST_CODE_CONT	VARCHAR(50)
	CTRY_CODE	VARCHAR(50)
	CTRY_NAME	VARCHAR(128)
	LOC_ADDR_COD_LST_VER_VAL	VARCHAR(50)
	LOC_ADDR_CODE_LIST_VERS_IDE	VARCHAR(50)
	LOC_ADDR_CODE_LIS_VER_AGN_IDE	VARCHAR(50)
	CNTY_CODE	VARCHAR(50)
I2	CNTY_NAME	VARCHAR(128)
	LOC_ADDR_CODE_LIST_VER_IDE	VARCHAR(50)
	LOC_ADDR_COD_LIS_VER_AGN_IDE	VARCHAR(50)
	LOC_ADD_COD_LST_VER_VAL	VARCHAR(50)
	MAIL_ADDR_TEXT	VARCHAR(255)
	SUPP_ADDR_TEXT	VARCHAR(255)
	MAIL_ADDR_CITY_NAME	VARCHAR(128)
I9	MAIL_ADDR_STA_CODE	VARCHAR(50)
	MAIL_ADDR_STA_NAME	VARCHAR(128)
	MAIL_ADDR_CODE_LST_VER_VAL	VARCHAR(50)
	MAIL_ADDR_CODE_LIST_VERS_IDEN	VARCHAR(50)
	MAIL_ADDR_CODE_LIS_VER_AGN_IDE	VARCHAR(50)
I12	MAIL_ADDR_ADDR_POST_CODE_VAL	VARCHAR(50)
	MAIL_ADDR_ADDR_POST_CODE_CONT	VARCHAR(50)
	MAIL_ADDR_CTRY_CODE	VARCHAR(50)
	MAIL_ADDR_CTRY_NAME	VARCHAR(128)
	MAIL_ADDR_COD_LST_VER_VAL	VARCHAR(50)
	MAIL_ADDR_CODE_LIST_VERS_IDE	VARCHAR(50)
	MAIL_ADDR_COD_LIS_VER_AGN_IDE	VARCHAR(50)
I10	ORIG_PART_NAME	VARCHAR(128)
I6	INFO_SYS_ACRO_NAME	VARCHAR(128)
I7	LAST_UPDT_DATE	DATETIME

FACID_FAC_FAC_NAICS		
PK	**FAC_FAC_NAICS_ID**	**VARCHAR(40)**
FK1,I1	**FAC_ID**	**VARCHAR(40)**
	FAC_NAICS_CODE	VARCHAR(50)
	FAC_NAICS_PRI_INDI	VARCHAR(9)

FACID_FAC_FAC_SIC		
PK	**FAC_FAC_SIC_ID**	**VARCHAR(40)**
FK1,I1	**FAC_ID**	**VARCHAR(40)**
	SIC_CODE	VARCHAR(50)
	SIC_PRI_INDI	VARCHAR(9)

FACID_FAC_ALT_IDEN		
PK	**FAC_ALT_IDEN_ID**	**VARCHAR(40)**
FK1,I1	**FAC_ID**	**VARCHAR(40)**
	ALT_IDEN_IDEN	VARCHAR(50)
	ALT_IDEN_TYPE_TEXT	VARCHAR(255)

FACID_ALT_NAME		
PK	**ALT_NAME_ID**	**VARCHAR(40)**
FK1,I1	**FAC_ID**	**VARCHAR(40)**
	ALT_NAME_TEXT	**VARCHAR(255)**
	ALT_NAME_TYPE_TEXT	**VARCHAR(255)**

FACID_FAC_FAC_AFFL		
PK	**FAC_FAC_AFFL_ID**	**VARCHAR(40)**
FK1,I2	**FAC_ID**	**VARCHAR(40)**
I1	**AFFL_IDEN**	**VARCHAR(255)**
	AFFL_TYPE_TEXT	VARCHAR(255)
	AFFL_START_DATE	DATETIME
	AFFL_END_DATE	DATETIME
	AFFL_STAT_TEXT	VARCHAR(4)
	AFFL_STAT_DETR_DATE	DATETIME

FACID_FAC_ELEC_ADDR		
PK	**FAC_ELEC_ADDR_ID**	**VARCHAR(40)**
FK1,I1	**FAC_ID**	**VARCHAR(40)**
	ELEC_ADDR_TEXT	VARCHAR(255)
	ELEC_ADDR_TYPE_NAME	VARCHAR(8)

FACID_ENVR_INTR_FAC_AFFL		
PK	**ENVR_INTR_FAC_AFFL_ID**	**VARCHAR(40)**
FK1,I2	**ENVR_INTR_ID**	**VARCHAR(40)**
I1	**AFFL_IDEN**	**VARCHAR(255)**
	AFFL_TYPE_TEXT	VARCHAR(255)
	AFFL_START_DATE	DATETIME
	AFFL_END_DATE	DATETIME
	AFFL_STAT_TEXT	VARCHAR(4)
	AFFL_STAT_DETR_DATE	DATETIME

FACID_ENVR_INTR_ALT_IDEN		
PK	**ENVR_INTR_ALT_IDEN_ID**	**VARCHAR(40)**
FK1,I1	**ENVR_INTR_ID**	**VARCHAR(40)**
	ALT_IDEN_IDEN	VARCHAR(50)
	ALT_IDEN_TYPE_TEXT	VARCHAR(255)

FACID_ENVR_INTR		
PK	**ENVR_INTR_ID**	**VARCHAR(40)**
FK1,I1	**FAC_ID**	**VARCHAR(40)**
I2	**ENVR_INTR_IDEN**	**VARCHAR(50)**
	ENVR_INTR_TYPE_TEXT	**VARCHAR(255)**
	ENVR_INTR_START_DATE	DATETIME
	ENVR_INTR_START_DATE_QUAL_TEXT	VARCHAR(255)
	ENVR_INTR_END_DATE	DATETIME
	ENVR_INTR_END_DATE_QUAL_TEXT	VARCHAR(255)
	ENVR_INTR_ACTIVE_INDI	VARCHAR(1)
	ENVR_INTR_URL_TEXT	VARCHAR(255)
I4	ORIG_PART_NAME	VARCHAR(128)
I5	INFO_SYS_ACRO_NAME	VARCHAR(128)
I3	LAST_UPDT_DATE	DATETIME
	AGN_TYPE_CODE	VARCHAR(50)
	AGN_TYPE_NAME	VARCHAR(128)
	CODE_LIST_VERS_IDEN	VARCHAR(50)
	CODE_LIST_VERS_AGN_IDEN	VARCHAR(50)
	CODE_LST_VER_VAL	VARCHAR(50)

FACID_ENVR_INTR_ELEC_ADDR		
PK	**ENVR_INTR_ELEC_ADDR_ID**	**VARCHAR(40)**
FK1,I1	**ENVR_INTR_ID**	**VARCHAR(40)**
	ELEC_ADDR_TEXT	VARCHAR(255)
	ELEC_ADDR_TYPE_NAME	VARCHAR(8)

FACID_ENVR_INTR_FAC_NAICS		
PK	**ENVR_INTR_FAC_NAICS_ID**	**VARCHAR(40)**
FK1,I1	**ENVR_INTR_ID**	**VARCHAR(40)**
	FAC_NAICS_CODE	VARCHAR(50)
	FAC_NAICS_PRI_INDI	VARCHAR(9)

FACID_ENVR_INTR_FAC_SIC		
PK	**ENVR_INTR_FAC_SIC_ID**	**VARCHAR(40)**
FK1,I1	**ENVR_INTR_ID**	**VARCHAR(40)**
	SIC_CODE	VARCHAR(50)
	SIC_PRI_INDI	VARCHAR(9)