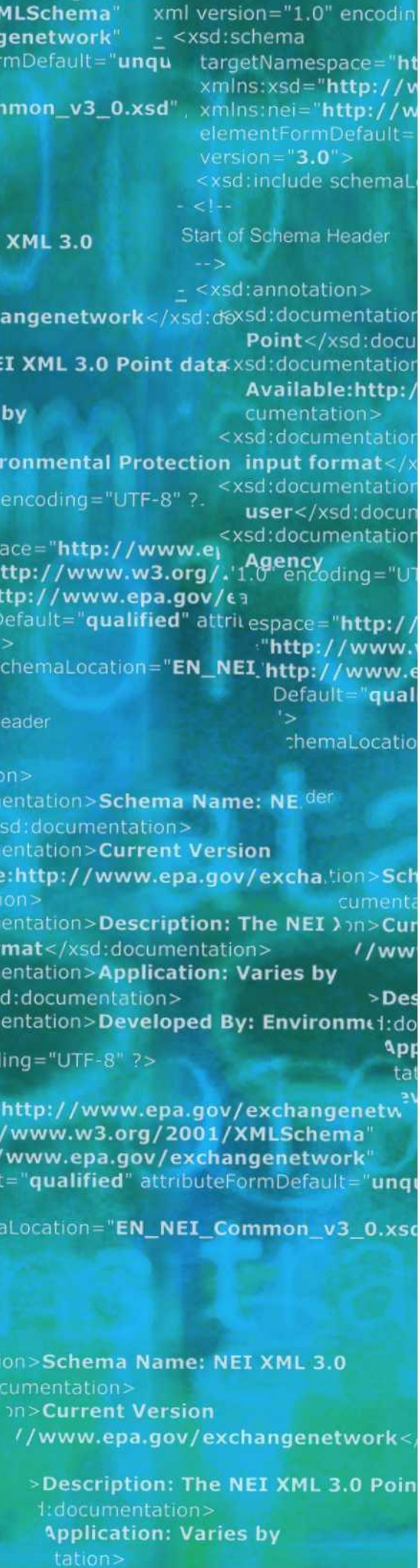


OpenNode2

ICIS-NPDES v5.8 Data Exchange Implementation Guide (Java)

Revision Date: 8/29/2017

Prepared By:

WINDSOR SOLUTIONS

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

Environmental Information exchange Network

Revision History

Date	Author	Changes
9/21/2012	Windsor	Initial version. Supports ICIS-NPDES v4.0 data flow
10/3/2012	Windsor	Minor correction to instructions for setting exchange's Web Info field.
10/29/2012	Windsor	Reorder database setup procedure to place creation of views before procedures
10/30/2012	Windsor	Reorder and clarify "Setting up the Staging Database" section
11/23/2012	Windsor	Minor updates to Result Processing section.
12/17/2012	Windsor	Update to separate out staging database deployment instructions for Oracle and SQL Server and make minor corrections to SQL Server instructions.
4/14/2015	Windsor	Updated for ICIS-NPDES Schema v5 plugin: - Updated screenshots and service/schedule configuration - Updated database setup steps - Updated Appendix A and B to include v5 payload changes - Added Appendix C – Upgrading v4 to v5 - Replaced v4.0 references with v5.0
10/3/2016	Windsor	Updated Appendix B block diagram to reference new tables with version 5.6 XML schema.
11/21/2016	Windsor	Updated Oracle staging database deployment instructions.
8/29/2017	Windsor	Updated to include new/changed payloads for version 5.8.

Table of Contents

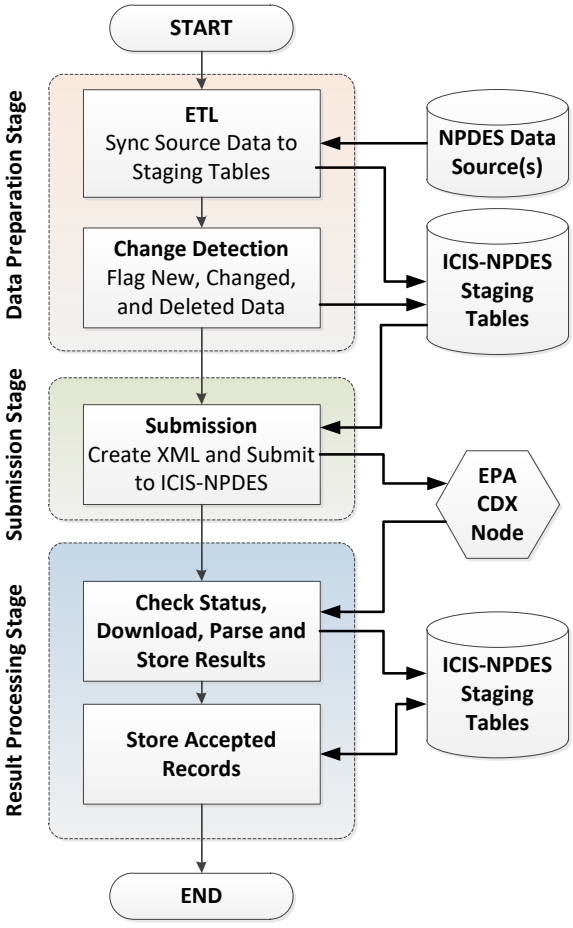
DATA EXCHANGE OVERVIEW 1
STAGING TABLE ARCHITECTURE 3
Staging Table Schemas *3*
Staging Tables *4*
Data Staging Approaches *5*
Data Preparation Stored Procedures *6*
KEY ASPECTS OF PLUGIN DESIGN 7
SETTING UP THE STAGING DATABASE 9
Oracle Database Deployment *9*
SQL Server Database Deployment *11*
INSTALL AND CONFIGURE ICIS-NPDES PLUGIN 12
Create ICIS-NPDES Data Exchange *12*
Add CDX to the OpenNode2 Partner List *14*
Create ICIS-NPDES Data Services *14*
Create Data Exchange Schedules *20*
ADDITIONAL ACTIVITIES 22
Contact CDX to Establish Data Exchange Settings *22*
Monitor Flow Activity *22*
APPENDIX A: PAYLOAD STAGING TABLES 23
APPENDIX B - TABLE/MODULE DIAGRAMS 26
Permit, Limit, DMR Module *26*
Permit Components *27*
Program Reports *28*
Compliance Monitoring (Inspections) *29*
Enforcement *30*
Violations *30*
Linkages *31*

Data Exchange Overview

The purpose of this document is to provide detailed instructions for the installation and configuration of the ICIS-NPDES Full Batch data exchange on the Java implementation of the Exchange Network OpenNode2 (OpenNode2).

More information on this data exchange can be found on the Exchange Network website. Before implementing this exchange, it is highly recommended that the user review the ICIS-NPDES Flow Configuration Document (FCD) and XML Schema User's Guide available on the Exchange Network website, as well as the ICIS-NPDES Full Batch Plugin Design Specification document available from the OpenNode2 Google Code repository.

The following diagram illustrates the three overall workflow stages and their sub-phases.



The **Data Preparation Stage** refreshes the staging database with the latest data from the state's NPDES information system. A change detection database stored procedure is bundled with the plugin that automatically determines what data from the staging database needs to get sent to ICIS-NPDES, alleviating the ETL logic from needing to perform this task. See the following sections for more information.

In the **Submission Stage**, the plugin retrieves new, changed, or deleted data from the staging tables, converts the data to ICIS-NPDES XML, validates the XML, and transmits the XML to EPA's Central Data Exchange (CDX). This stage is executed by a scheduled task within OpenNode2.

In the **Result Processing Stage**, the plugin retrieves ICIS-NPDES processing reports from EPA's CDX system and parses the accepted and rejected transactions into a tracking table. Accepted records are copied to a set of staging tables that are used to reflect the current data in ICIS and used to determine the data that needs to be sent in subsequent submissions. This stage is executed by a scheduled task within OpenNode2.

Each stage is executed in isolation and is only responsible for one aspect of the exchange. This design is intended to maintain a separation of concerns, allowing each component to focus only on a single task. By extension, each component should have little or no dependency on the specific tasks performed by the other stages.

The workflow is tracked within the staging database. The workflow tracking table prevents the stages from being executed out of sequence and halts processing if an error occurs that requires manual intervention to resolve.

Staging Table Architecture

Like most OpenNode2 exchanges, the ICIS-NPDES exchange leverages a series of staging tables. These staging tables serve as a "parking lot" for data that is ready to be sent via OpenNode2 to CDX. These staging tables closely match the structure of the XML schema for the exchange.

Staging Table Schemas

The ICIS-NPDES full batch plugin requires that two separate database schemas[1] be created; referred to in this document as **Staging-Local** and **Staging-ICIS**. Staging-Local (named ICS_FLOW_LOCAL) stores data that the agency wishes to flow to ICIS. Staging-ICIS (named ICS_FLOW_ICIS) stores transactions that have been successfully added to ICIS by the plugin and therefore, represents a current snapshot of all data present in ICIS for the agency. The plugin moves data from Staging-Local to Staging-ICIS after the data has been successfully sent to ICIS.

Staging-Local Schema (ICS_FLOW_LOCAL)

Staging-Local is used to store data that the agency wishes to flow to ICIS-NPDES. It is populated by a state-specific Extraction, Transformation, and Loading (ETL) routine on a regular interval.

The Staging-Local tables closely follow the structure of the ICIS-NPDES XML Schema. The data in these tables is populated by a state-specific ETL routine that reads data from the source NPDES information management system(s), translates the data into ICIS-NPDES structures and formats (including use of proper lookup values and business rules), and inserts the transformed data into these tables.

Guidelines for designing an ETL process to populate these tables are included in the ETL Design Considerations section of this document.

Staging-ICIS Schema (ICS_FLOW_ICIS)

The tables in the Staging-ICIS schema are used to store all the records that have been accepted by EPA ICIS-NPDES. The table structures are identical to that of Staging-Local schema, which again, are a reflection of the ICIS-NPDES XML schema.

When an Accepted Transactions XML Report is processed by OpenNode2, accepted data are moved from Staging-Local to Staging-ICIS. Subsequent submissions are generated by comparing the contents of Staging-Local with Staging-ICIS to determine which transactions need to be generated and submitted.

Data in Staging-ICIS should never be manually manipulated. The data in this schema is maintained entirely by logic embedded in the plugin software and associated database routines. While data in this schema should not be altered, the state's ETL routines may wish to read data from this area as a means of helping to determine what data to add, change, or delete in Staging-Local.

[1] For Oracle, two schemas are created and maintained. For SQL Server implementations, two different databases are used. For simplicity, the term "schema" throughout this document.

Staging Tables

Each schema contains the following tables:

- 168 staging tables to store NPDES data
- 1 table to track accepted and rejected transactions (ICS_SUBM_RESULTS)
- 1 table to track plugin workflow execution state (ICS_SUBM_TRACK)
- 1 table to track the version of the ICIS schema database (ICS_ABOUT_DB)

Each type of table is described in more detail in the following sections.

ICIS-NPDES Staging Tables

Almost all the tables in the Staging-Local and Staging-ICIS schemas are used to store NPDES data. The root table is ICS_PAYLOAD. Below this table are 48 child tables, each relating to a different submission type supported by ICIS-NPDES.

Results Tracking Table (ICS_SUBM_RESULTS)

The ICIS-NPDES plugin staging table schema contains a table, ICS_SUBM_RESULTS, used to store the business keys for accepted and rejected transactions along with warning and error messages returned by ICIS-NPDES after a submission file has been processed. Rejected transactions (identified with a RESULT_CODE of 'Error') are retained for all previous submissions. Accepted transactions (identified with a RESULT_CODE of 'Accepted' or 'Warning') are only retained for a brief period while the results are processed and are deleted after the accepted records are copied from Staging-Local to Staging-ICIS as part of the built-in plugin workflow.

Data in this table is maintained by the logic within the OpenNode2 plugin responsible for retrieving and parsing processing results XML files. Data in this table should not be manually manipulated; however this data can be very useful in helping to troubleshoot data that failed processing into ICIS.

While this table is present in both the Staging-Local and Staging-ICIS schemas, only the table in Staging-Local is used. The equivalent table in Staging-ICIS is unused and can be dropped if desired.

Transaction Tracking Table (ICS_SUBM_TRACK)

The ICS_SUBM_TRACK table stores a record for each execution of the full data preparation, submission, and result processing lifecycle. Data in this table is maintained by the built-in data exchange logic and should not be manually manipulated. Detailed information about how this table is used to track lifecycle events is described in the separate ICIS-NPDES Plugin Design Specification document.

While this table is present in both the Staging-Local and Staging-ICIS schemas, only the table in Staging-Local is used. The equivalent table in Staging-ICIS is unused and can be dropped if desired.

Data Staging Approaches

A custom database Extract, Transformation, and Load (ETL) routine will need to be built by the implementing agency to load ICIS data from the source database to the OpenNode2 staging database. The ICIS-NPDES plugin supports three different approaches to staging and submitting data. An agency will need to decide the approach that best supports its own needs. The chosen method determines how the agencies ETL routines are written. The approaches are:

- Full Data Synchronization
- Incremental Data with Automatic Change Detection
- Incremental Data with Manual Change Detection

Approach 1: Full Data Synchronization

Using this approach, the ETL process updates the NPDES data in Staging-Local to represent a reflection of *all* the current NPDES data in the state system. States may wish to completely rebuild the data each time the ETL runs, or alternatively, incrementally insert, update, or delete records to bring the data in the staging schema up to date.

When this approach is used, the AUTO_GEN_DELETES field in ICS_PAYLOAD must be set to 'Y' for the modules where this approach is used. This triggers the built-in change-detection process to automatically create Delete or Mass Delete transactions whenever data in Staging-Local is missing, but data for the same business key has already been successfully sent to ICIS-NPDES is the past, as determined by presence of the data in the Staging-ICIS schema.

For example, imagine an agency accidentally marks an internal outfall on a permit as an external outfall. The agency's ETL process picks up this outfall and adds it to the staging database. The outfall data is then sent to and is accepted by ICIS-NPDES. The agency then realizes the mistake and updates the outfall to be marked as internal. The subsequent ETL run will no longer pick up this outfall and it is therefore not loaded in the staging environment on the next execution. If auto-detection of deletes is turned on for the module, the plugin's change detection process will automatically insert a Delete transaction for the outfall, thus removing it from ICIS-NPDES.

Approach 2: Incremental Data with Automatic Change Detection

Using this alternative, the agency decides to only populate the staging database with a specific set of data to be sent. This approach would be typical for DMR data, where only the data received or changed in the previous week, month, or other timeframe is staged at a time.

This approach requires that the ETL process be designed more intricately, since only new or changed data from the last successful submission is populated in the staging database. The most recent successful submission date can be determined by finding the most recent ETL Completed Date/Time for a successful submission tracked in the ICS_SUBM_TRACK table. It also requires that the agency's NPDES system contains audit fields on every record (record last updated date) so that data changes can be reliably detected. Any record with a created or updated data greater than the last successful ETL date should be transformed into the staging database. Lastly, it requires that the state system also track what data was deleted and when, so that the necessary transactions can be sent to ICIS to delete the data in ICIS as well.

If this approach is used, the AUTO_GEN_DELETES field must be set to "N". Otherwise, the plugin will attempt to delete all data in ICIS that was successfully sent to ICIS in the past but is not present in the Staging-Local schema after the most recent ETL execution!

Approach #3: Incremental Data with Manual Change Detection

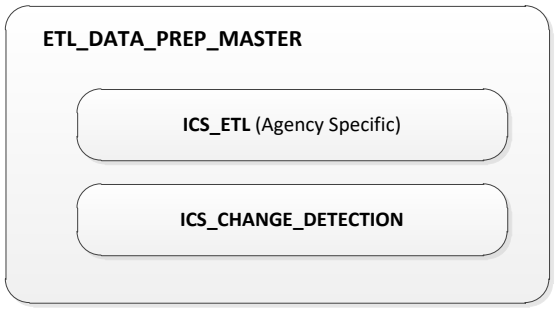
This approach is similar to #2 above, however the agency chooses to set the transaction codes using their own developed ETL logic instead of leveraging the built-in change detection procedure. This approach requires that the state has developed its own mechanism to track which data needs to be added, updated, or removed from ICIS in the batch dataflow.

If this approach is used, the agency should remove all the following components from their staging environment:

- The entire Staging-ICIS schema and all tables within it.
- The change detection stored procedure in Staging-Local (ICS_CHANGE_DETECTION)
- All change detection views (CDV_* database objects in Staging-Local)
- The ICS_PROCESS_ACCEPTED_TRANS procedure in Staging-Local

Data Preparation Stored Procedures

The ICIS-NPDES full batch plugin comes bundled with a series of scripts to create the necessary tables, views and stored procedures needed to support the full batch data flow. Among these are three stored procedures that together comprise the data preparation stage. The diagram below illustrates the relationship between these procedures.



ETL_DATA_PREP_MASTER is the parent stored procedure that encapsulates the agency-specific ICS_ETL procedure and the bundled ICS_CHANGE_DETECTION procedure. The workflow for the Master procedure is as follows:

1. Check the ICS_SUBM_TRACK table to ensure that no existing workflows have a status of 'Pending'. If any exist, exit without error, returning a NULL ICS_SUBM_TRACK_ID in the procedure's output parameter.
2. Begin a new transaction.
3. Insert a new record in the ICS_SUBM_TRACK table to begin a new workflow with a status of 'Pending'.
4. Call the agency-specific ICS-ETL procedure.
5. Update the workflow record with the completion date/time of the ETL process.
6. Call the ICS_CHANGE_DETECTION procedure.
7. Update the workflow record with the completion date/time of the change detection process.
8. Commit the transaction.
9. Return the ICS_SUBM_TRACK_ID in an output parameter from the procedure.
10. If an error occurs, rollback the transaction and raise the database error so it can be handled by the external process that executed the Master procedure.

An agency may choose to remove the change detection stored procedure if they choose to manually set transaction codes in their own ETL process (described in Approach 3 in the previous section).

Key Aspects of Plugin Design

This section lists the key aspects of the ICIS-NPDES Full Batch plugin design that have important implications for flow implementers to understand how the overall workflow functions. This section is not comprehensive. Plugin implementers should review the ICIS-NPDES Full Batch Plugin Design Specification to fully understand the plugin functionality.

Data Integrity

- Agencies should only update data within the Staging-Local schema. Data in the Staging-ICIS schema should never be altered or manipulated after the flow is in production. Data in Staging-ICIS is managed entirely by the plugin.

Root Staging Tables

- The root staging table is ICS_PAYLOAD. A record should exist in this table for every submission type to be submitted to EPA. The only time when a record should be added or removed from this table is if the agencies is adding or removing payload types from their data flow.
- The ICS_PAYLOAD table has an AUTO_GEN_DELETES column. If set to 'Y', the change detection procedure will automatically create delete transactions for records that are not present in Staging-Local but are present in Staging-ICIS.
- "Root" staging tables exist for all 46 ICIS payload types. See Appendix A for a list of supported payload types and their corresponding root staging table name.
- The root staging tables have unique constraints set on the business key fields. This ensures that each business entity (permit, limit set, etc.) only exists once in each staging schema/database. This is by design. These constraints should not be removed. See Appendix A for a list of business key fields.
- Each root table contains a column labeled SRC_SYSTM_IDENT. This field is ignored by the plugin. It is meant to provide the agency with a convenient place to store the primary key value for the source record from the agency's source NPDES database that maps to the staging record. This can be convenient for ETL design and data auditing purposes.

ETL Design Considerations

- If a workflow fails, the agency's ETL procedure must be able to restore the staged data to the same state it was before the failed workflow occurred, assuming no changes were made to the source data. This ensures there are no gaps in the submitted data.
- Some staging tables have multiple foreign keys. Examples include ICS_CONTACT and ICS_TELEPHONE. Only one FK should be populated at a time. For example, one contact or phone record should not be referenced by multiple parent tables.

Workflow Management

A Workflow is the term given to a single execution of the end-to-end submission lifecycle. A workflow begins with the data preparation stage and ends with the completion of downloading/parsing of processing results or failure of the workflow at some point in its execution.

- Overall workflow is managed in the ICS_SUBM_TRACK table. The Master data preparation procedure is responsible for creating a new workflow. New workflows are allowed only if there are no existing 'Pending' workflows.
- A workflow can be set to Failed at many different points in the execution of a workflow. Some examples of events that result in a failed workflow are XML validation failure, submission failure, or a processing failure at CDX. Failure reason messages are stored in the ICS_SUBM_TRACK table's WORKFLOW_STAT_MESSAGE column and in the Activity Log within OpenNode2.
- A workflow is set to complete when the plugin successfully retrieves a "Complete" status from CDX, downloads the processing reports, parses the results, and successfully copies accepted transactions from Staging-Local to Staging-ICIS.
- If a failure occurs any time after submission of a file to CDX, the workflow status will stay in 'Pending' state. This is to ensure that no new workflows are created before the processing results from CDX can be consumed and interpreted by the plugin. When this situation occurs, *manual intervention will be required* to resolve the workflow before a new workflow can begin. When a resolution is made, the workflow will manually need to be set to 'Complete' or 'Failed' or the ETL will not resume and no new workflows will be created.

Result Processing

- The ICS_SUBM_RESULTS table stores accepted and rejected transaction data returned from ICIS-NPDES. This table contains columns for each of the business key fields used throughout the ICIS-NPDES XML schema. The business key fields vary depending on the submission type. The submission type is stored in the SUBM_TYPE_NAME field. Business key fields by submission type are listed in Appendix A of this document.
- The GetICISStatusAndProcessReports plugin service is responsible for checking the status of outstanding submissions. When a status returns 'Complete' from CDX, the plugin will download and parse the transactions into the ICS_SUBM_RESULTS table.
- Once the plugin finishes saving the data from the processing reports, the plugin then executes a post-processing stored procedure named ics_process_accepted_trans. This procedure is bundled with the plugin database scripts. The procedure copies data from Staging-LOCAL to Staging-ICIS for accepted transactions. For more information on this process, see the detailed plugin design specification. Some key aspects of the processing procedure are as follows:
    - Accepted transactions from previous submissions are purged each time new results files are downloaded and processed by the plugin. Therefore, the results table only shows accepted business keys from the most recent submission.
    - If a business key is accepted in the most recent submission, any previous warnings or errors for the same business key are deleted from the results table. This ensures that resolved errors are gone, allowing a data steward to focus only on outstanding issues.
    - If a business key returns a warning or error, any previous errors for the same business key are deleted from the results table. This ensures that the warnings or errors in the results table are accurate for the current state of submission data.

Setting up the Staging Database

Follow the steps below to establish the database environment for the ICIS-NPDES Full Batch data flow. The process is the same for Oracle and SQL Server. The database scripts referenced in the instructions below are included in the plugin download zip file.

These steps assume data staging approach #1 or #2 is selected. See the previous section for considerations if approach #3 is selected. Separate instructions are provided for the Oracle and SQL Server database platforms.

Oracle Database Deployment

1. 3 Oracle schema are required to securely support the ICIS NPDES data flow. Create 3 Oracle schema named ICS_FLOW_LOCAL, ICS_FLOW_LOCAL_USER and ICS_FLOW_ICIS. Grant Oracle permissions to the schema owners as outlined below:
    a. **ICS_FLOW_LOCAL**
        i. CONNECT
        ii. RESOURCE
        iii. CREATE TABLE
        iv. CREATE VIEW
    b. **ICS_FLOW_ICIS**
        i. CONNECT
        ii. RESOURCE
        iii. CREATE TABLE
    c. **ICIS_FLOW_LOCAL_USER**
        i. CONNECT
        ii. RESOURCE
2. Create a database connection to your Oracle instance as schema owner ICS_FLOW_LOCAL. Execute the DDL script within the **staging_table_ddl** folder called ICIS_5.8-ORA-DDL.sql.
3. Create a database connection to your Oracle instance as schema owner ICS_FLOW_ICIS. Execute the DDL script within the **staging_table_ddl** folder called ICIS_5.8-ORA-DDL.sql.
4. Create a database connection to your Oracle instance as schema owner SYS. Execute the DDL script within the **grants** folder called ICIS_5.8-ORA-DDL-GRANTS-SYS.sql. This will grant execute permissions on the SYS owned package DBMS_CRYPTO to both SYSTEM (with grant option) and ICIS_FLOW_LOCAL. This is the only step that should be executed with the elevated SYS connection…!
5. Create a database connection to your Oracle instance as schema owner ICS_FLOW_ICIS, or SYSTEM. Execute the SQL script within the **grants** folder called ICIS_5.8-ORA-DDL-GRANTS-ICS_FLOW_ICIS.sql. This will grant the schema owner ICS_FLOW_LOCAL select, insert, update, and delete privileges on the database objects owned by ICS_FLOW_ICIS.
6. Create a database connection to your Oracle instance as schema owner ICS_FLOW_LOCAL. Execute the following SQL scripts within the **functions** folder.

a. ICIS_5.8-ORA-GET_LIMIT_MONTHS.sql

b. ICIS_5.8-ORA-MD5_HASH.sql

7. Create a database connection to your Oracle instance as schema owner ICS_FLOW_LOCAL. Execute the following SQL scripts within the **views** folder.

   a. **ICIS_5.8-ORA-Biz Rule Error Review Views.sql**: Optional, creates views that are tied to an error review tool in the OpenNode2 Data Viewer.

   b. **ICIS_5.8-ORA-Change Detection Views.sql**: Required, creates a data change detection view for each submission type. Each view is used by ETL Change Detection processing.

   c. **ICIS_5.8-ORA-ics_v_anml_type_hib.sql** – Required. Allows proper serialization of staging table data to XML via the Hibernate framework. This view is not needed for .NET OpenNode2 implementations.

8. Create a database connection to your Oracle instance as schema owner ICS_FLOW_LOCAL. Execute the following SQL scripts within the **procedures** folder. * Ensure that ICS_DATA_PREP_MASTER script is compiled last as its dependent on the prior scripts compiling first.

   a. ICIS_5.8-ORA-ICS_CHANGE_DETECTION.sql

   b. ICIS_5.8-ORA-ICS_ETL.sql

   c. ICIS_5.8-ORA-ICS_PROCESS_ACCEPTED_TRANS.sql

   d. ICIS_5.8-ORA-ICS_FORCE_REISSUANCE.sql

   e. ICIS_5.8-ORA-ICS_SET_HASHES.sql

   f. * ICIS_5.8-ORA-ICS_DATA_PREP_MASTER.sql

9. Create a database connection to your Oracle instance as schema owner ICS_FLOW_LOCAL. Execute the DDL script within the **grants** folder called ICIS_5.8-ORA-DDL-GRANTS-ICS_FLOW_LOCAL.sql.

10. Create a database connection to your Oracle instance as schema owner SYSTEM, or an alternate schema owner that has permissions to create synonyms within the ICS_FLOW_LOCAL_USER schema. Execute the DDL script within the **synonyms** folder called ICIS_5.8-ORA-DDL-SYNONYMS-ICS_FLOW_LOCAL_USER.sql.

11. The ICS_ETL procedure is only a shell. Update the ICS_ETL procedure to include the custom data loading procedures from your agency's NPDES source database. Use the following resources to help you design and build the ETL routines:

    - The staging table block diagrams in the appendix of this document,
    - The Full Batch plugin design specification, and
    - The EPA-provided ICIS-NPDES full batch documentation available on the Exchange Network web site (www.exchangenetwork.net).

12. Open table ICS_PAYLOAD in the ICS_FLOW_LOCAL schema/database. All 48 payload types accepted by ICIS-NPDES are inserted by default. You may remove the rows for the payload types you do not plan on submitting although it is not necessary.

13. If the agency already has data in ICIS-NPDES, populate the business key fields for the existing records into the respective payload table in the ICS_FLOW_ICIS schema. This will inform the

change detection process what data is already in ICIS, therefore preventing erroneous creation of new transactions for data that already exists in ICIS.

SQL Server Database Deployment

1. Create two databases named ICS_FLOW_LOCAL and ICS_FLOW_ICIS.
2. Run the data definition language (DDL) script to create the staging tables in the ICS_FLOW_LOCAL schema/database. Run the script again to create the same objects in ICS_FLOW_ICIS schema/database.

   Note: the DDL scripts begin by dropping the staging tables and constraints. This is useful if rebuilding the staging database from scratch. Comment out or skip this portion of the script when running the first time.

   a. Optionally, drop the ICS_SUBM_RESULTS and ICS_SUBM_TRACK tables from ICS_FLOW_ICIS since they are not used.
3. Run the script in the **functions** folder to create a function used by the bundled procedures.
4. Run the scripts in the **views** folder on the ICS_FLOW_LOCAL schema/database. The scripts are described briefly below:

   a. **ICIS_5.8-SQL-Biz Rule Error Review Views.sql** – Optional. Creates views that can be tied to an error review tool such as the OpenNode2 Data Viewer.

   b. **ICIS_5.8-SQL-Change Detection Views.sql** – Required. Creates a view for each submission type. Used by the ETL Change Detection procedure.

   c. **ICIS_5.8-SQL-ics_v_anml_type_hib.sql** – Only required if using the Java OpenNode2.
5. Run the scripts in the **procedures** folder on the ICS_FLOW_LOCAL schema/database. This will create the procedures used to execute the ETL, perform the change detection process, and process accepted transactions. Run the ICS_DATA_PREP_MASTER script ***last*** since it depends on two others.
6. Create a SQL login named ICS_FLOW_LOCAL_USER. This is the account that will be used by OpenNode2 to interact with the staging database and execute stored procedures.

   a. Create a new database role named *db_execute* that allows stored procedure execution rights.

   b. Map the ICS_FLOW_LOCAL_USER user to the *db_datareader*, *db_datawriter*, and *db_execute* roles in the ICS_FLOW_LOCAL database.

   c. Map the ICS_FLOW_LOCAL_USER user to the *db_datareader*, *db_datawriter*, and *db_execute* roles in the ICS_FLOW_ICIS database.
7. The ICS_ETL procedure is only a shell. Update the ICS_ETL procedure to include the custom data loading procedures from your agency's NPDES source database. Use the following resources to help you design and build the ETL routines:

   a. The staging table block diagrams in the appendix of this document,

   b. The Full Batch plugin design specification, and

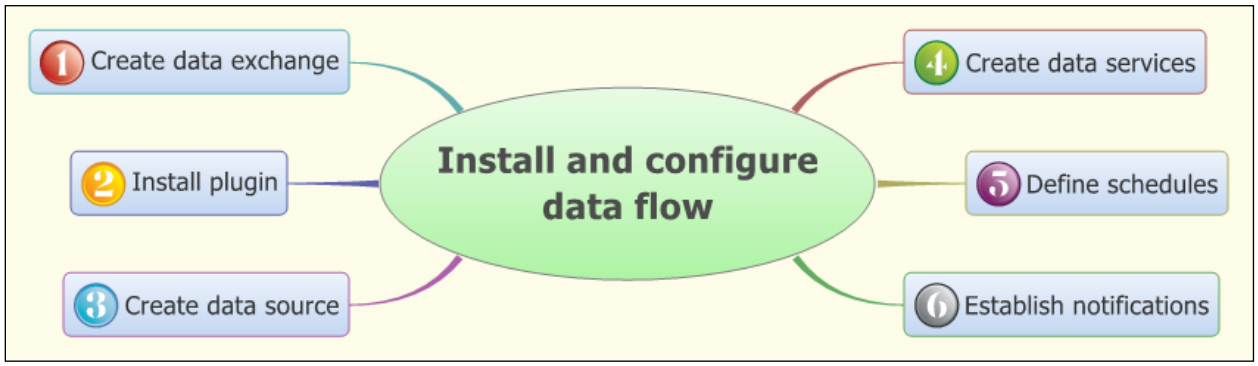
   c. The EPA-provided ICIS-NPDES full batch documentation available on the Exchange Network web site (www.exchangenetwork.net).

8. Open table ICS_PAYLOAD in the ICS_FLOW_LOCAL schema/database. All 48 payload types accepted by ICIS-NPDES are inserted by default. You may remove the rows for the payload types you do not plan on submitting although it is not necessary.
9. If the agency already has data in ICIS-NPDES, populate the business key fields for the existing records into the respective payload table in the ICS_FLOW_ICIS schema. This will inform the change detection process what data is already in ICIS, therefore preventing erroneous creation of new transactions for data that already exists in ICIS.

Install and Configure ICIS-NPDES Plugin

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

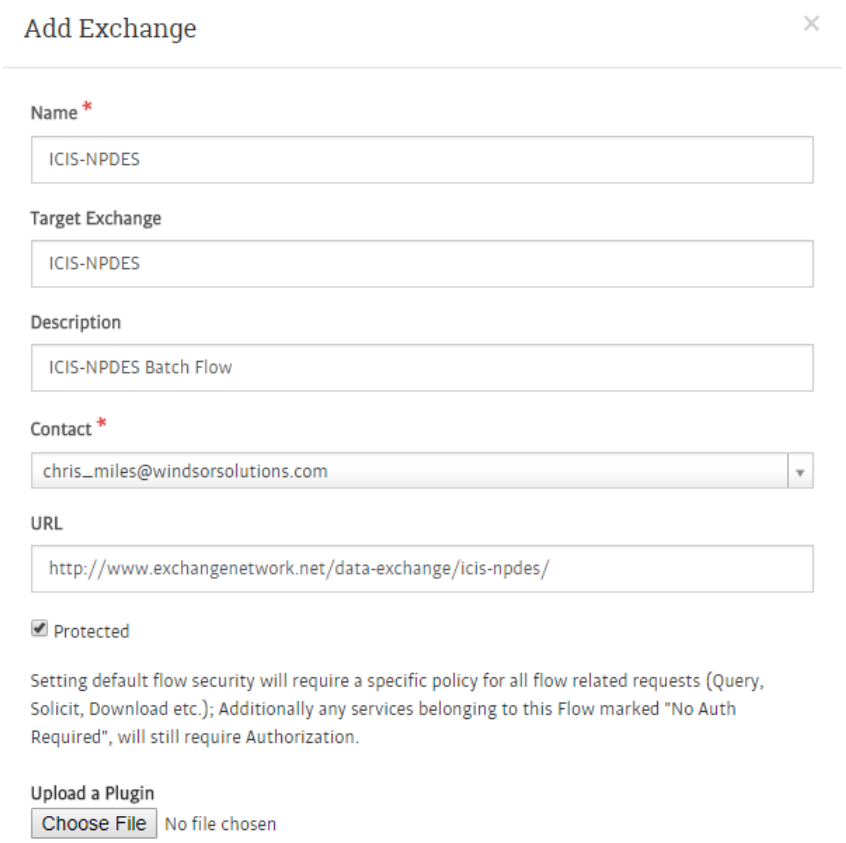
The following figure illustrates these steps:



Create ICIS-NPDES Data Exchange

The first step to implement the ICIS-NPDES 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 a new Exchange** button. The Data Exchange Manager screen will be displayed:



3. Set the **Name** to *ICIS-NPDES*.
4. Set the **Target Exchange Name** to *ICIS-NPDES*. This field can be used if EPA renames the exchange.
5. Select a user account name from the **Contact** drop down box. Contacts are populated with all accounts that have been set up on the OpenNode2. This value is not used by OpenNode2.
6. Type any valid URL in the **Web Info** field (e.g., the URL of the ICIS-NPDES webpage on the Exchange Network, *http://www.exchangenetwork.net/data-exchange/icis-npdes/*). This value is not used by OpenNode2.
7. Check the **Protected** checkbox. This will restrict access to the flow to only those external users that have an appropriate NAAS policy. This setting is technically unnecessary for this plugin since it does not offer external services such as Query or Solicit.
8. Unzip the **java-opennode2-v2.12-4-plugins-only.zip** file to a folder.
9. Click **Choose File**, then select **java_icisnpdes_plugin_v2.12.43b.zip** from the directory containing the plugin files extracted from **java-opennode2-v2.12-4-plugins-only.zip** in step 8.
10. Click **Save** to save the data exchange.

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.

Add CDX to the OpenNode2 Partner List

If necessary, add an endpoint to either CDX Test or CDX Production node, depending on the OpenNode2 deployment environment.

CDX Test: https://testngn.epacdxnode.net/cdx-enws10/services/NetworkNodePortType_V10

CDX Prod: https://cdxnodengn.epa.gov/cdx-enws10/services/NetworkNodePortType_V10

Follow the steps below to add a Node Partner to CDX Test or CDX Prod.

1. Click the **Configuration** tab and select **Network Partners** from the left navigation link.
2. Look through the list of configured Network Partners to see if the endpoint is already added.
3. If not, click the **Add Partner** button.
4. In the **Name** field, type *ICIS-CDX TEST* or *ICIS-CDX PROD*.
5. In the **Endpoint URL** field, type the URL to either the EPA CDX Test or CDX Prod node listed above.
6. Select *Node v1.1* from the Version drop-down menu.
7. Click the **Check Connection** button to verify that the node connection is successful.
8. Click the **Save** button.

Create ICIS-NPDES Data Services

The ICIS-NPDES Full Batch Plugin consists of two data services:

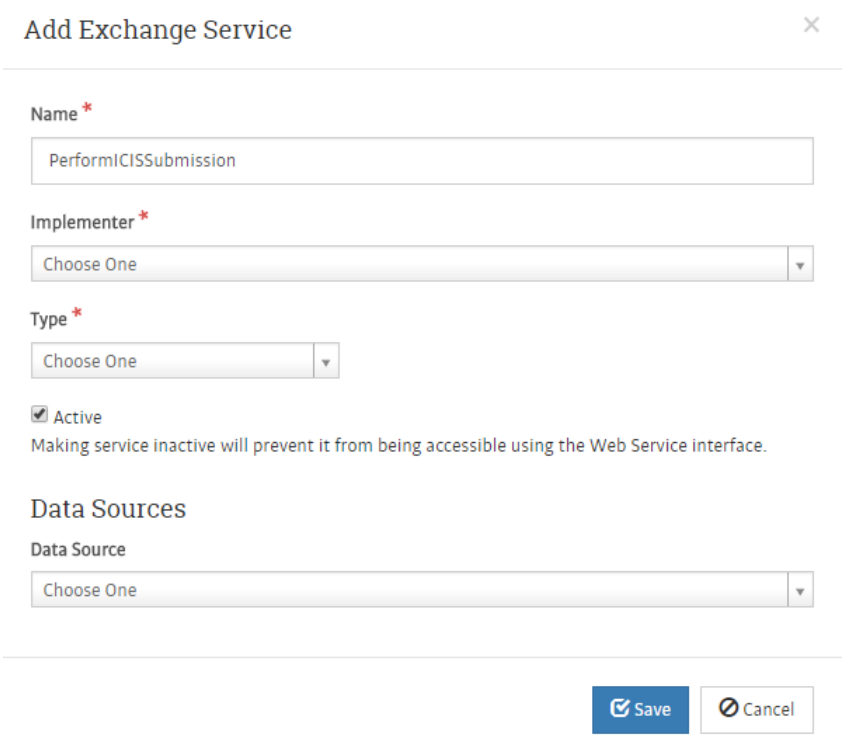
- **PerformICISSubmission** - responsible for sending data to ICIS-NPDES from the staging environment.
- **GetICISStatusAndProcessReports** - responsible for checking the status of previous submissions and, if processing is completed, downloading and parsing the Accepted and Rejected Transactions reports and parsing the results into a result tracking table.

Create the PerformICISSubmission Data Service

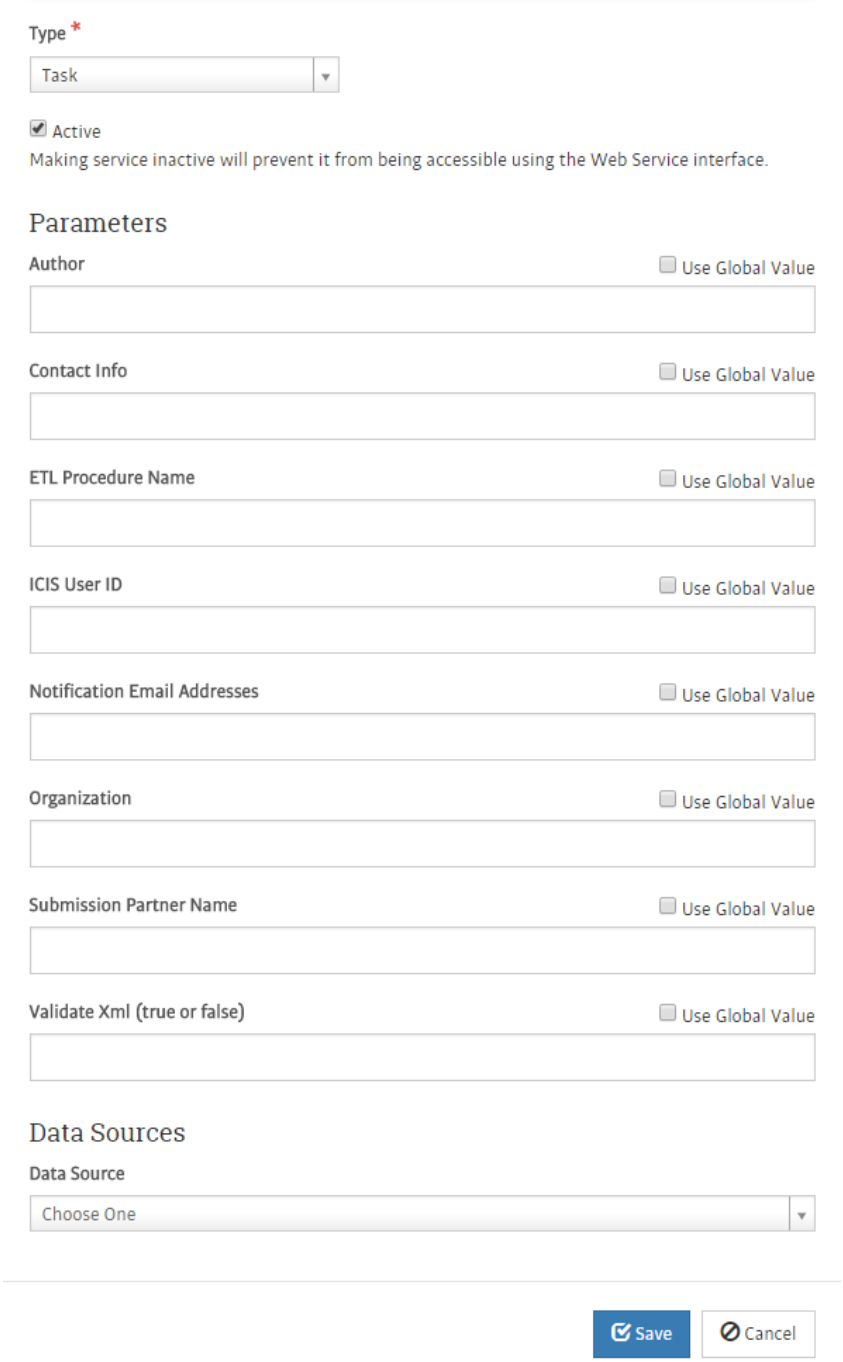
The **PerformICISSubmission** data service will be called by the schedule to read data from the ICIS-NPDES staging database, convert the data in XML format for delivery. It can be configured to submit the XML to an exchange network partner.

From the **Exchange** tab, locate the ICIS-NPDES data exchange in the list of available exchanges.

1. Click the **Add (+)** button located to the right of the ICIS-NPDES data exchange entry. The following page will be displayed to allow a new data service to be added.



2. In the **Name** field, enter "PerformICISSubmission".
3. Select the "PerformICISSubmission" entry from the **Implementer** drop-down menu.
4. The rest of the service arguments will appear:



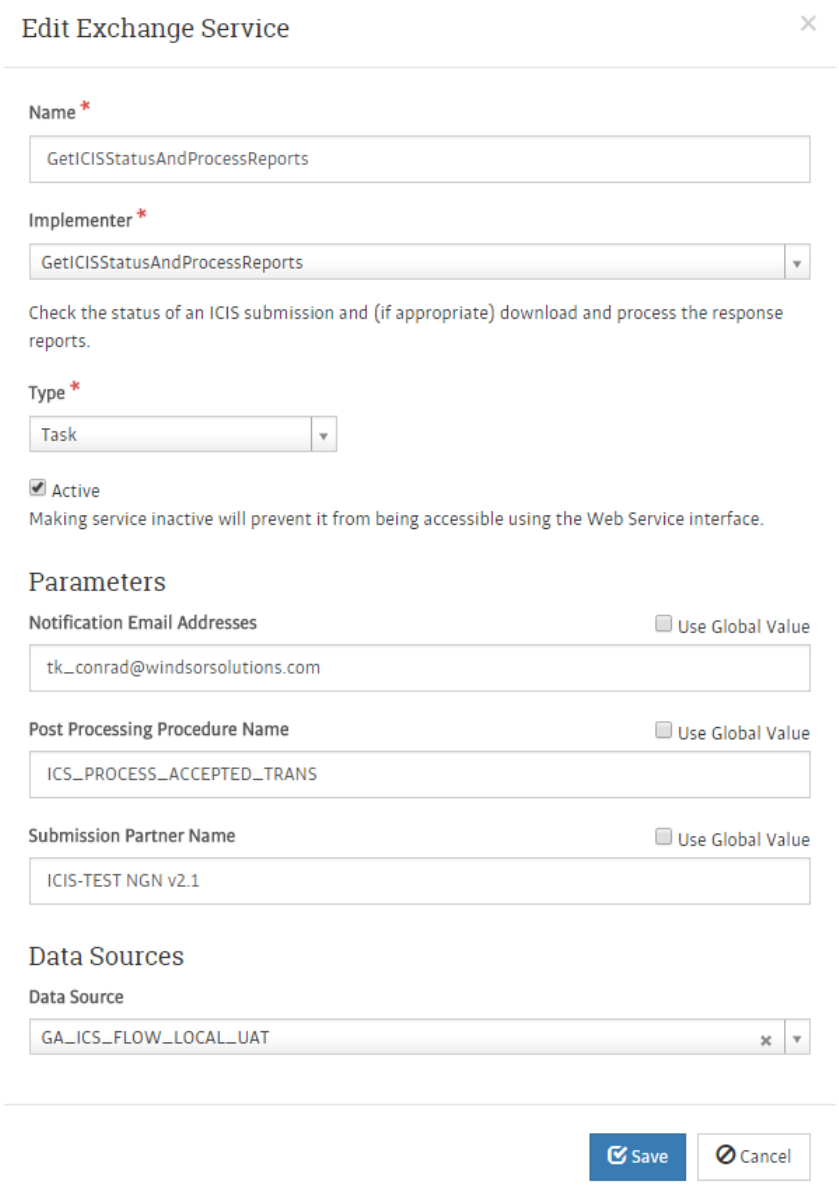
5. From the **Type** drop-down menu, select "Task". The Scheduler will use this service to compose XML for submission to EPA.
6. Enable the service by checking the **Active** checkbox.
7. Optionally, supply text for the **Author**, **Contact Info**, and **Organization** fields. These values are inserted into the XML header block for every submission that is created by the service.

8. If the plugin is to be responsible for executing the Data Preparation database procedures, enter the name of the main data preparation stored procedure in the **ETL Procedure Name** field. The procedure supplied with the plugin for this purpose is "ETL_DATA_PREP_MASTER". Alternatively, the data preparation can be invoked by some external process, in which case this field should be left blank.
9. Set the **ICIS User ID** field, set the name of the relevant ICIS WAM account to be inserted in the XML file as part of the payload Header.
10. If desired, add one or more semicolon-delimited email addresses in the **Notification Email Addresses** field. Each address will be added to the XML header submission, instructing CDX to send email notifications when submissions are received by CDX and when Processing finished (either completed or failed).
11. In the **Submission Partner Name** field, type the name of the network partner configured in OpenNode2 for either the CDX Test or Prod environment. (See *Add CDX to the OpenNode2 Partner List* section above). This field can be left blank. If not set, the submission file will be built and stored in the node's transaction log without sending. This can be useful for testing purposes.
12. In the **Validate Xml** field, type either "true" or "false". It is recommended that this be set to "true".
13. For the argument labeled **Source Data Provider**, choose the OpenNode2 data source that hosts the ICIS-NPDES Staging-Local schema/database. The account must have read/write access to the staging tables. If necessary, please see the OpenNode2 Administration Guide for information on setting up and testing data sources.
14. If the plugin will be responsible for executing the data preparation stored procedures, also configure the data source for the **ETL Data Source** field. This will most often be the same data source as configured in the previous step.
15. Click the **Save** button to save the service.

Create the GetICISStatusAndProcessReports Data Service

The GetICISStatusAndProcessReports service is responsible for checking the status of previous submissions and, if processing is completed, downloading and parsing the Accepted and Rejected Transactions reports and parsing the results into a result tracking table.

1. From the **Exchange** tab, locate the ICIS-NPDES data exchange in the list of available exchanges.
2. Click the **Add (+)** button located to the right of the ICIS-NPDES data exchange entry. The following page will be displayed to allow a new data service to be added.



3. In the **Service Name** field, enter "GetICISStatusAndProcessReports".
4. Select the "GetICISStatusAndProcessReports" entry from the **Implementer** drop-down menu.
5. Click the **Next** button. The rest of the service arguments will appear.
6. From the **Type** drop-down menu, select "Task". This makes the service visible to the Scheduler component.
7. Enable the service by checking the **Active** checkbox.
8. If desired, add one or more semicolon-delimited email addresses in the **Notification Email Addresses** field. OpenNode2 will send PDF processing reports downloaded from CDX for completed submissions to all addresses in this list. Note that PDFs are not distributed in the event of failed processing.
9. Set the **Post Processing Procedure Name** to "ics_process_accepted_trans". This is the name of the stored procedure supplied with the plugin that is responsible for copying accepted transactions

from the Staging-Local to Staging-ICIS schema/database. The accepted transactions are used to drive the automatic change detection process.

10. Set the **Submission Partner Name** to the same network partner configured in step 11 in the service configuration described on the previous page.
11. For the **Source Data Provider**, choose the OpenNode2 data source that hosts the ICIS Staging-Local staging tables. If necessary, please see the OpenNode2 Administration Guide for information on setting up and testing data sources.
12. Click the **Save** button to save the service.

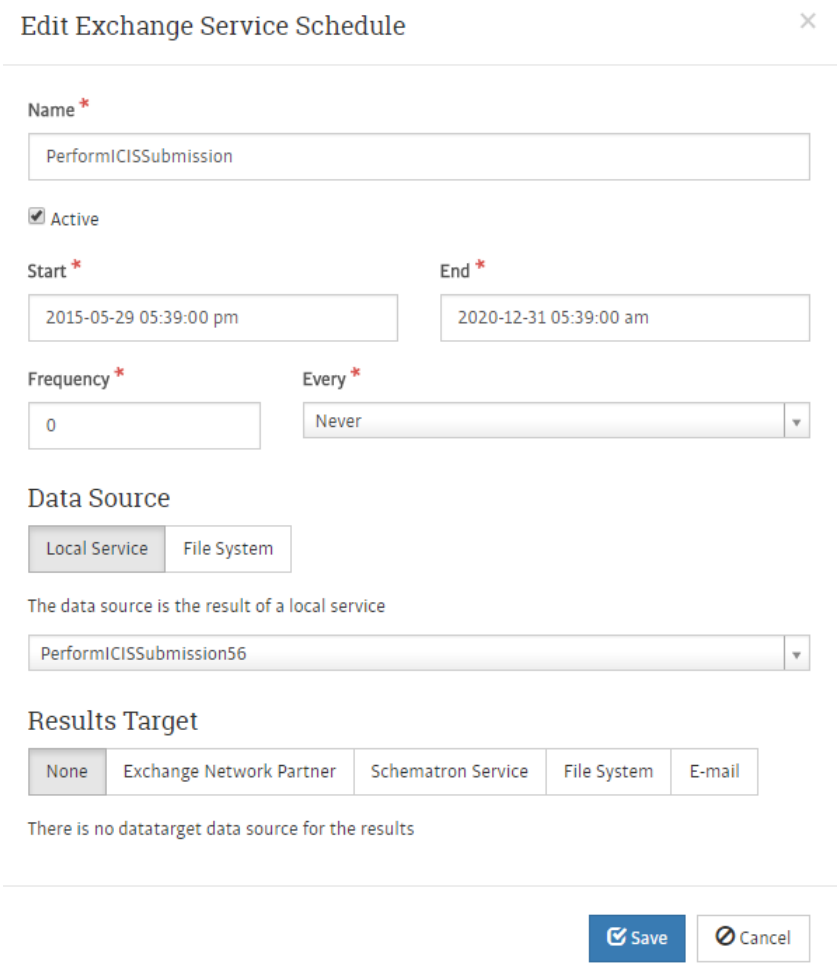
Create Data Exchange Schedules

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.

The ICIS-NPDES Full Batch data exchange should have two schedules set up; one to submit data to ICIS-NPDES and one to check the status of submissions and download/parse reports from ICIS for completed submissions.

Create "Perform ICIS Submission" Schedule

1. From the **Schedules** tab, click the **Add** button.



2. Type "Perform ICIS Submission" in the **Name** field.
3. Enable the schedule by clicking the **Active** checkbox if not already checked.
4. Select "ICIS-NPDES" from the **Exchange** dropdown list.
5. Set the **Start** date to the date on which you wish the schedule to run, typically todays date. Set the **End** date to some date in the distant future. Note that the time of day to execute is also set using this control.
6. Set the **Frequency** to the desired interval. Daily submissions are typical.

7. In the **Data Source** area, check the radio button labeled **Results of local service execution**.
8. In the **Service** dropdown box, select the value "PerformICISSubmission". This informs the schedule to use the selected ICIS-NPDES service as the data source for the submission.
9. No changes are needed to the **Additional Parameters** area.
10. In the **Result Process** area, check the radio button labeled **None.**

> **Important Note:** The plugin will perform the submission to ICIS, so the schedule does not need to perform this task. **Do not** set the schedule to submit to an Exchange Network Partner.

11. Click the **Save** button to save the schedule.

Create the "Get ICIS Status and Download Reports" Schedule

1. From the **Schedules** tab, click the **Add Schedule** button.
2. Type "Get ICIS Status and Download Reports" in the **Name** field.
3. Enable the schedule by clicking the **Active** checkbox if not already checked.
4. Select "ICIS-NPDES" from the **Exchange** dropdown list.
5. In the **Availability** area, set the **Starts On** date to the date on which you wish the schedule to run, typically todays date. Set the **Ends On** date to some date in the distant future. The run time should take in to consideration the run time of the submission process and the run time of the processing at CDX, as mentioned in step 5 in the previous schedule.
6. Set the **Frequency** to the desired interval. The frequency should be the same as the submission schedule or more frequent.
7. In the **Data Source** area, check the radio button labeled **Results of local service execution**.
8. In the **Service** dropdown box, select the value "GetICISStatusAndDownloadReports". This informs the schedule to use the selected ICIS-NPDES service as the processor for the task.
9. In the **Result Process** area, check the radio button labeled **None**.
10. Click the **Save** button to save the schedule.

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

Additional Activities

Contact CDX to Establish Data Exchange Settings

Once the ICIS-NPDES DMR data exchange is installed and configured, contact the EPA CDX Node helpdesk and ask them to authorize the OpenNode2 runtime (operator) NAAS account to submit to the ICIS-NPDES data exchange on the EPA systems for both the Test and Production CDX Node environments.

Monitor Flow Activity

The OpenNode2 will track all ICIS-NPDES 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.

The best way to audit data errors is to develop a report off the **ics_subm_results** table. This table lists all the errors encountered by ICIS for each submission module. This should be used to guide data cleanup efforts in the agency source data system. Once data is corrected and it flows successfully to ICIS-NPDES, the error will automatically clear from the table.

Appendix A: Payload Staging Tables

The following table lists the submission types supported by ICIS-NPDES v5.0, the corresponding staging table, and business key fields.

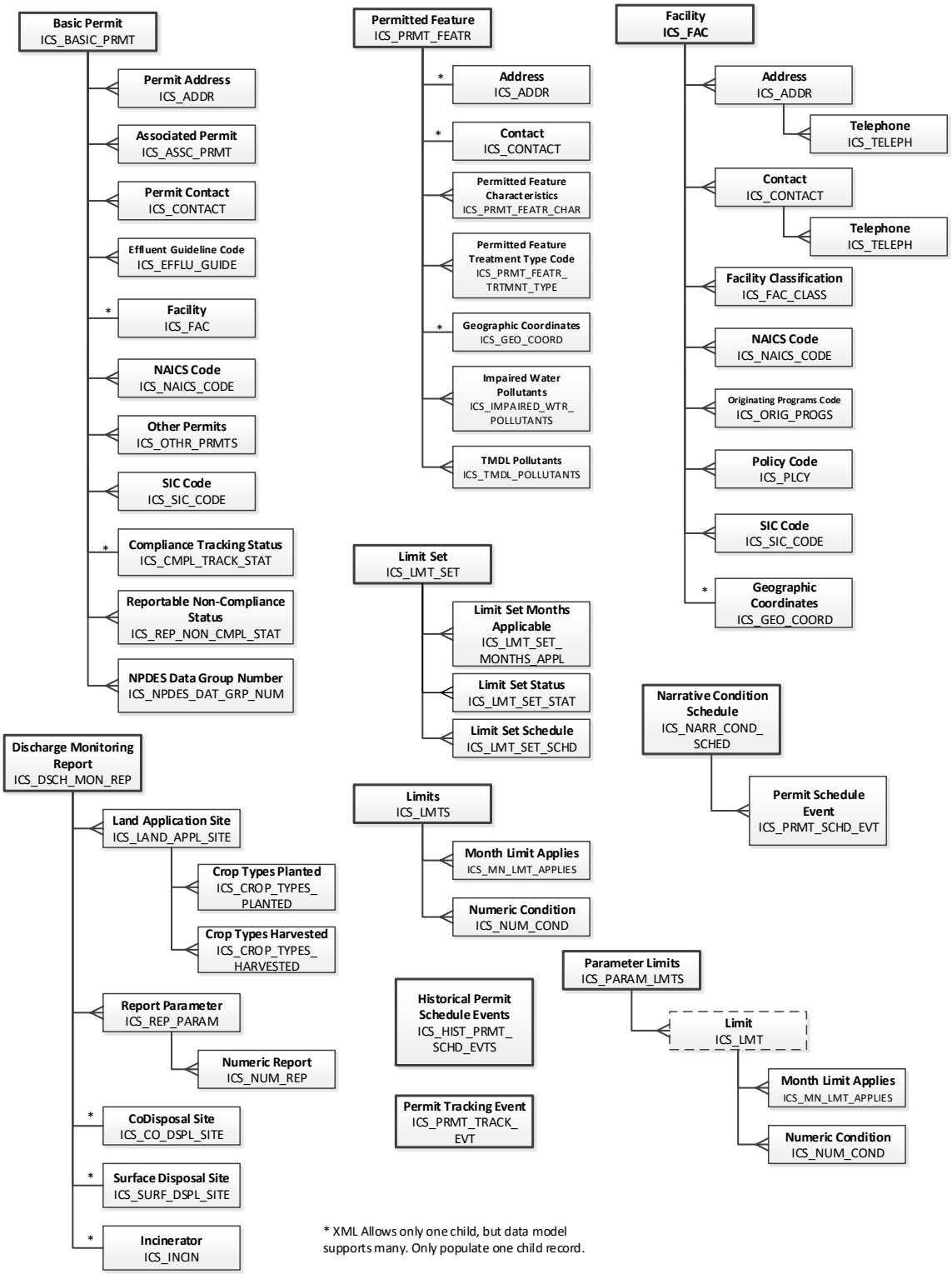
Submission Type	Staging Table Name	Business Key Fields
Basic Permit	ICS_BASIC_PRMT	PRMT_IDENT
Biosolids Annual Program Report	ICS_BS_ANNUL_PROG_REP	PRMT_IDENT, BS_ANNUL_REP_RCVD_DATE
Biosolids Permit	ICS_BS_PRMT	PRMT_IDENT
Biosolids Program Report	ICS_BS_PROG_REP	PRMT_IDENT, REP_COVERAGE_END_DATE
CAFO Annual Report	ICS_CAFO_ANNUL_REP	PRMT_IDENT, PRMT_AUTH_REP_RCVD_DATE
CAFO Permit	ICS_CAFO_PRMT	PRMT_IDENT
Compliance Monitoring	ICS_CMPL_MON	PRMT_IDENT, CMPL_MON_CATG_CODE, CMPL_MON_DATE
Compliance Monitoring Linkage	ICS_CMPL_MON_LNK	PRMT_IDENT, CMPL_MON_CATG_CODE, CMPL_MON_DATE, PRMT_IDENT_2, SNGL_EVT_VIOL_CODE, SNGL_EVT_VIOL_DATE, ENFRC_ACTN_IDENT, REP_COVERAGE_END_DATE, PRMT_AUTH_REP_RCVD_DATE, CSO_EVT_DATE, PRETR_PERF_SUMM_END_DATE, SSO_ANNUL_REP_RCVD_DATE, SSO_EVT_DATE, SSO_MONTHLY_REP_RCVD_DATE, DATE_STRM_EVT_SMPL, SW_MS_4_REP_RCVD_DATE, CMPL_MON_CATG_CODE_2, CMPL_MON_DATE_2
Compliance Schedule	ICS_CMPL_SCHD	ENFRC_ACTN_IDENT, FINAL_ORDER_IDENT, PRMT_IDENT, CMPL_SCHD_NUM
CSO Event Report	ICS_CSO_EVT_REP	PRMT_IDENT, CSO_EVT_DATE
CSO Permit	ICS_CSO_PRMT	PRMT_IDENT
Discharge Monitoring Report	ICS_DSCH_MON_REP	PRMT_IDENT, PRMT_FEATR_IDENT, LMT_SET_DESIGNATOR, MON_PERIOD_END_DATE
DMR Program Report Linkage	ICS_DMR_PROG_REP_LNK	PRMT_IDENT, PRMT_FEATR_IDENT, LMT_SET_DESIGNATOR, MON_PERIOD_END_DATE, PRMT_IDENT_2, REP_COVERAGE_END_DATE, DATE_STRM_EVT_SMPL
DMR Violation	ICS_DMR_VIOL	PRMT_IDENT, PRMT_FEATR_IDENT, LMT_SET_DESIGNATOR, MON_PERIOD_END_DATE, PARAM_CODE, MON_SITE_DESC_CODE, LMT_SEASON_NUM, NUM_REP_CODE, NUM_REP_VIOL_CODE

Effluent Trade Partner	ICS_EFFLU_TRADE_PRTNER	PRMT_IDENT, PRMT_FEATR_IDENT, LMT_SET_DESIGNATOR, PARAM_CODE, MON_SITE_DESC_CODE, LMT_SEASON_NUM, LMT_START_DATE, LMT_END_DATE, LMT_MOD_EFFECTIVE_DATE, TRADE_ID
Enforcement Action Milestone	ICS_ENFRC_ACTN_MILESTONE	ENFRC_ACTN_IDENT, MILESTONE_TYPE_CODE
Enforcement Action Violation Linkage	ICS_ENFRC_ACTN_VIOL_LNK	ENFRC_ACTN_IDENT, PRMT_IDENT, NARR_COND_NUM, SCHD_EVT_CODE, SCHD_DATE, ENFRC_ACTN_IDENT_2, FINAL_ORDER_IDENT, PRMT_IDENT_2, CMPL_SCHD_NUM, PRMT_FEATR_IDENT, LMT_SET_DESIGNATOR, PARAM_CODE, MON_SITE_DESC_CODE, LMT_SEASON_NUM, MON_PERIOD_END_DATE, SNGL_EVT_VIOL_CODE, SNGL_EVT_VIOL_DATE
Final Order Violation Linkage	ICS_FINAL_ORDER_VIOL_LNK	ENFRC_ACTN_IDENT, FINAL_ORDER_IDENT, PRMT_IDENT, NARR_COND_NUM, SCHD_EVT_CODE, SCHD_DATE, ENFRC_ACTN_IDENT_2, FINAL_ORDER_IDENT_2, CMPL_SCHD_NUM, …
Formal Enforcement Action	ICS_FRML_ENFRC_ACTN	ENFRC_ACTN_IDENT
General Permit	ICS_GNRL_PRMT	PRMT_IDENT
Historical Permit Schedule Events	ICS_HIST_PRMT_SCHD_EVTS	PRMT_IDENT, PRMT_EFFECTIVE_DATE, NARR_COND_NUM, SCHD_EVT_CODE, SCHD_DATE, NARR_COND_NUM, SCHD_EVT_CODE, SCHD_DATE
Informal Enforcement Action	ICS_INFRML_ENFRC_ACTN	ENFRC_ACTN_IDENT
Limit Set	ICS_LMT_SET	PRMT_IDENT, PRMT_FEATR_IDENT, LMT_SET_DESIGNATOR
Limits	ICS_LMTS	PRMT_IDENT, PRMT_FEATR_IDENT, LMT_SET_DESIGNATOR, PARAM_CODE, MON_SITE_DESC_CODE, LMT_SEASON_NUM, LMT_START_DATE, LMT_END_DATE
Local Limits Program Report	ICS_LOC_LMTS_PROG_REP	PRMT_IDENT, PRMT_AUTH_REP_RCVD_DATE
Master General Permit	ICS_MASTER_GNRL_PRMT	PRMT_IDENT
Narrative Condition Schedule	ICS_NARR_COND_SCHD	PRMT_IDENT, NARR_COND_NUM
Parameter Limits	ICS_PARAM_LMTS	PRMT_IDENT, PRMT_FEATR_IDENT, LMT_SET_DESIGNATOR, PARAM_CODE, MON_SITE_DESC_CODE, LMT_SEASON_NUM
Permit Reissuance	ICS_PRMT_REISSU	PRMT_IDENT, PRMT_ISSUE_DATE
Permit Termination	ICS_PRMT_TERM	PRMT_IDENT
Permit Tracking Event	ICS_PRMT_TRACK_EVT	PRMT_IDENT, PRMT_TRACK_EVT_CODE, PRMT_TRACK_EVT_DATE
Permitted Feature	ICS_PRMT_FEATR	PRMT_IDENT, PRMT_FEATR_IDENT

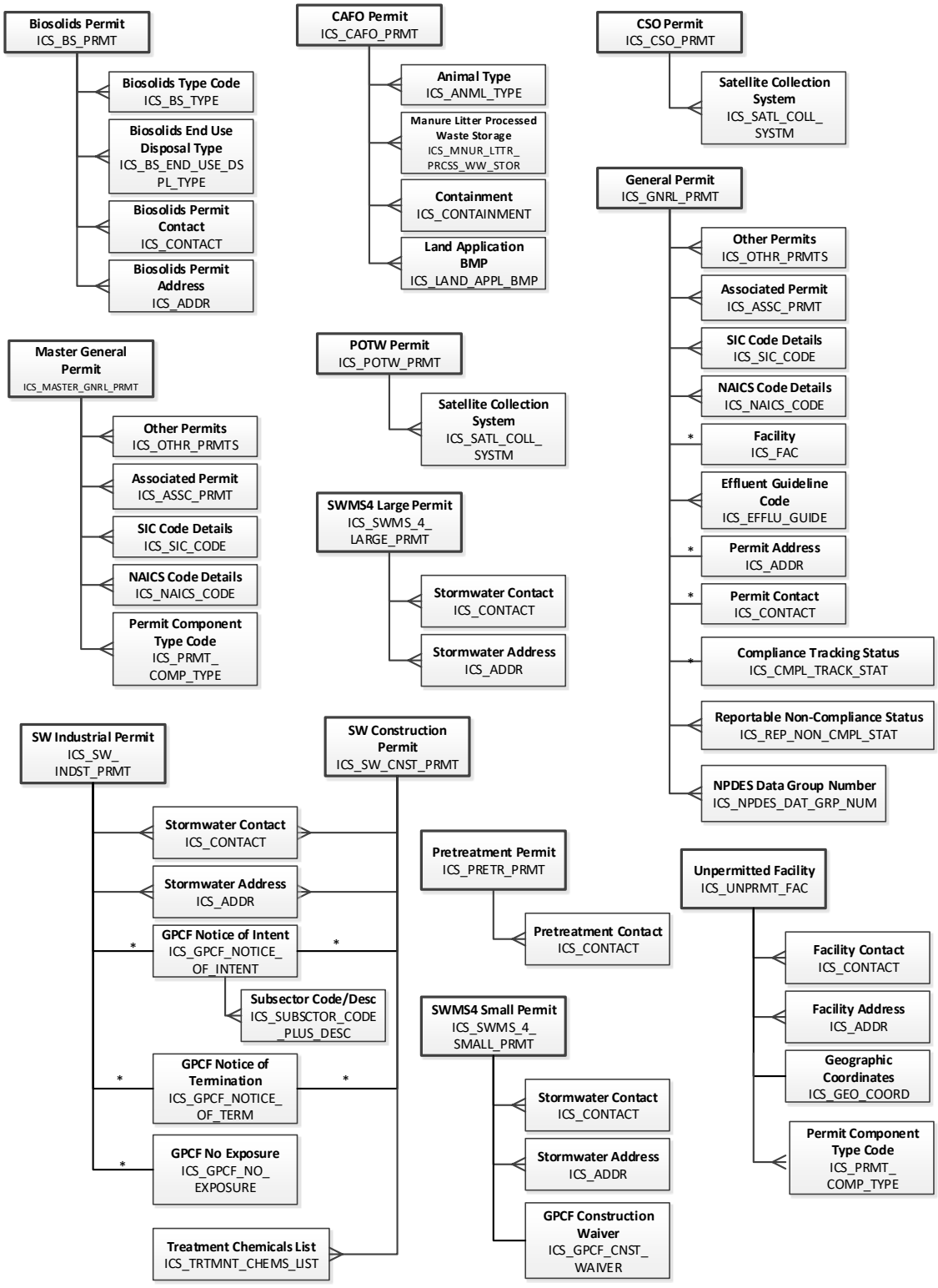
POTW Permit	ICS_POTW_PRMT	PRMT_IDENT
Pretreatment Performance Summary	ICS_PRETR_PERF_SUMM	PRMT_IDENT, PRETR_PERF_SUMM_END_DATE
Pretreatment Permit	ICS_PRETR_PRMT	PRMT_IDENT
Schedule Event Violation	ICS_SCHD_EVT_VIOL	PRMT_IDENT, NARR_COND_NUM, SCHD_EVT_CODE, SCHD_DATE, SCHD_VIOL_CODE, ENFRC_ACTN_IDENT, FINAL_ORDER_IDENT, CMPL_SCHD_NUM
Single Event Violation	ICS_SNGL_EVT_VIOL	PRMT_IDENT, SNGL_EVT_VIOL_CODE, SNGL_EVT_VIOL_DATE
SSO Annual Report	ICS_SSO_ANNUL_REP	PRMT_IDENT, SSO_ANNUL_REP_RCVD_DATE
SSO Event Report	ICS_SSO_EVT_REP	PRMT_IDENT, SSO_EVT_DATE
SSO Monthly Event Report	ICS_SSO_MONTHLY_EVT_REP	PRMT_IDENT, SSO_MONTHLY_REP_RCVD_DATE
SW Construction Permit	ICS_SW_CNST_PRMT	PRMT_IDENT
SW Event Report	ICS_SW_EVT_REP	PRMT_IDENT, DATE_STRM_EVT_SMPL
SW Industrial Annual Report	ICS_SW_INDST_ANNUL_REP	PRMT_IDENT, INDST_SW_ANNUL_REP_RCVD_DATE
SW Industrial Permit	ICS_SW_INDST_PRMT	PRMT_IDENT
SW MS4 Large Permit	ICS_SWMS_4_LARGE_PRMT	PRMT_IDENT
SW MS4 Program Report	ICS_SWMS_4_PROG_REP	PRMT_IDENT, SW_MS_4_REP_RCVD_DATE
SW MS4 Small Permit	ICS_SWMS_4_SMALL_PRMT	PRMT_IDENT
Unpermitted Facility	ICS_UNPRMT_FAC	PRMT_IDENT

Appendix B - Table/Module Diagrams

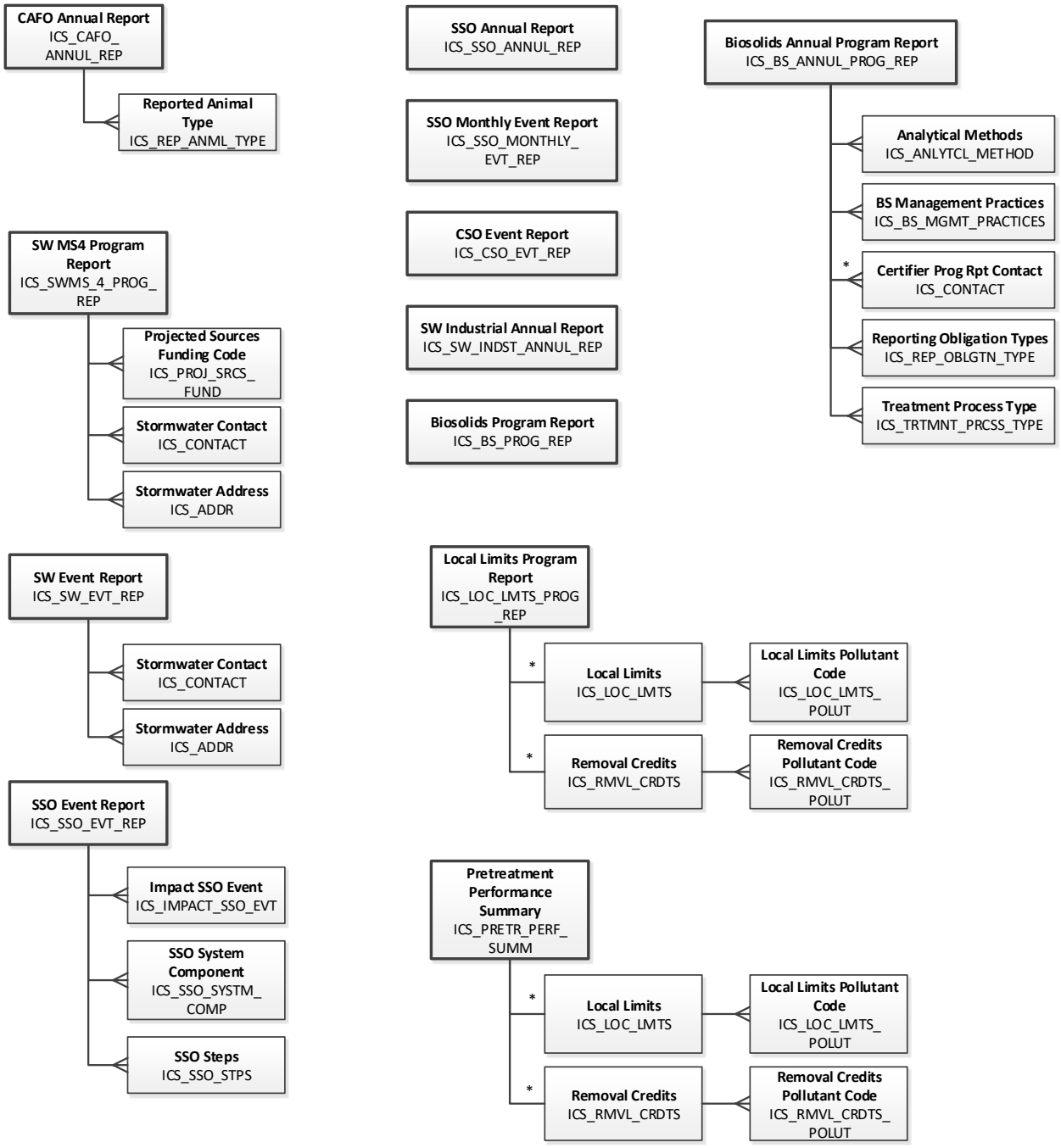
Permit, Limit, DMR Module



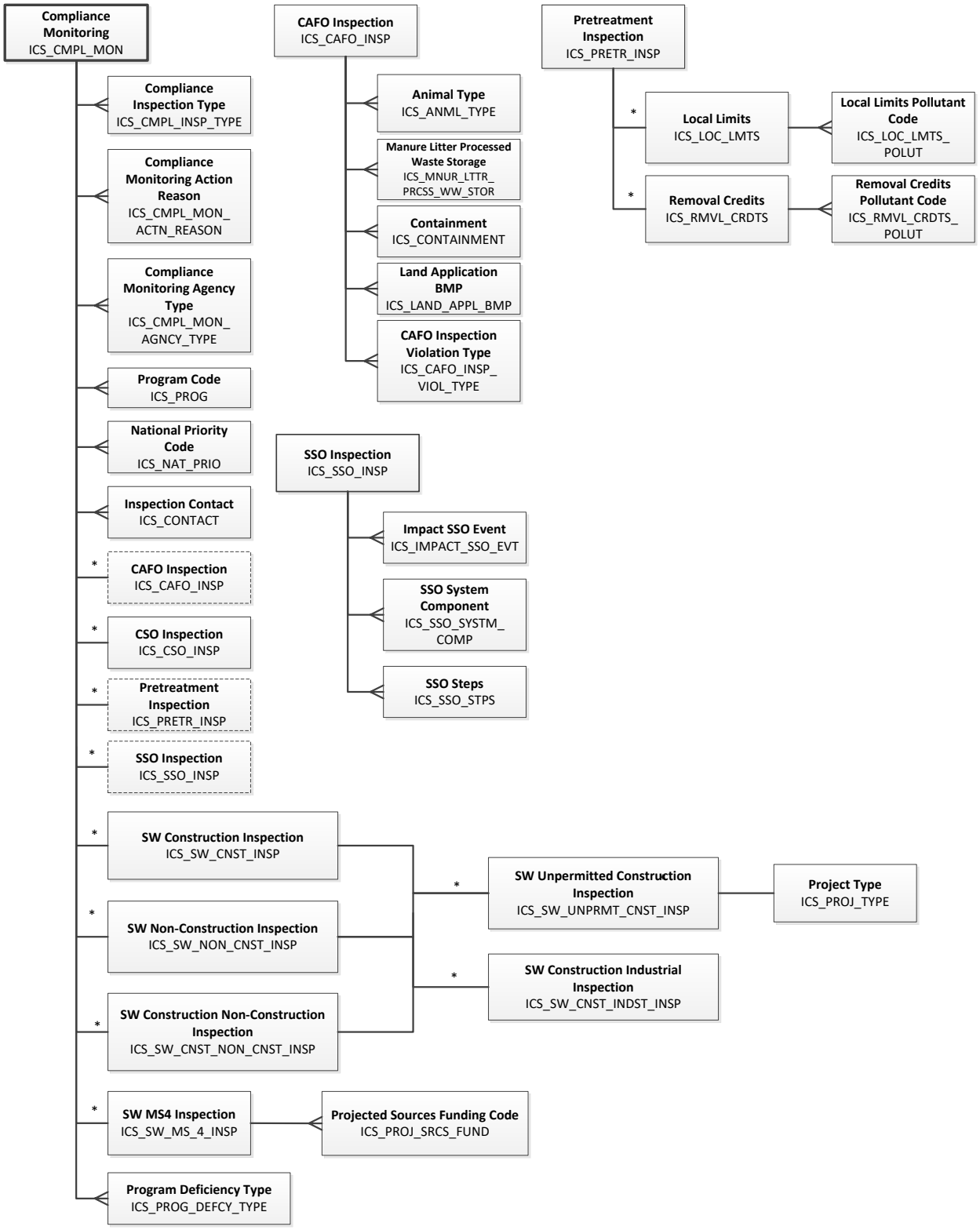
Permit Components



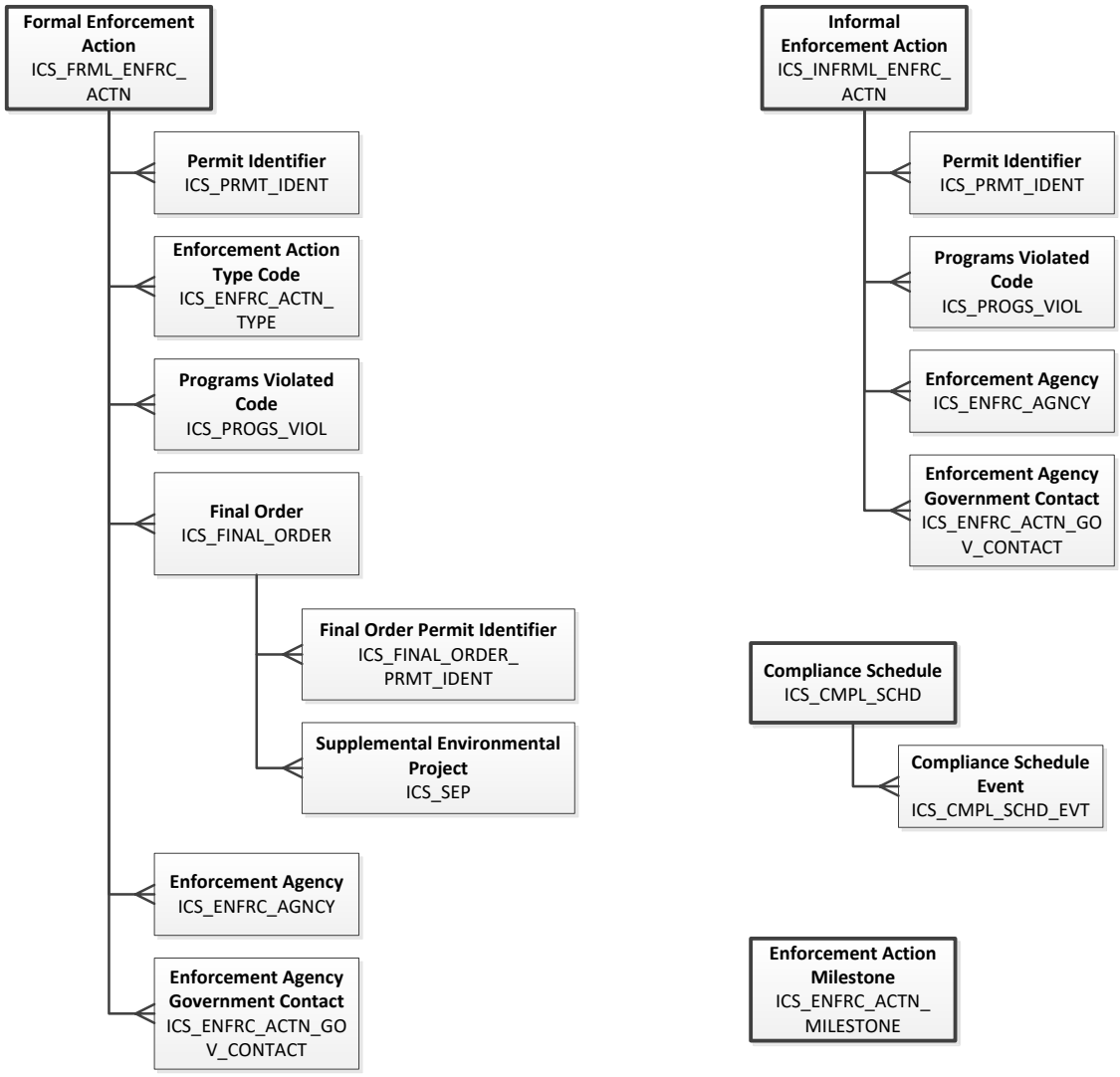
Program Reports



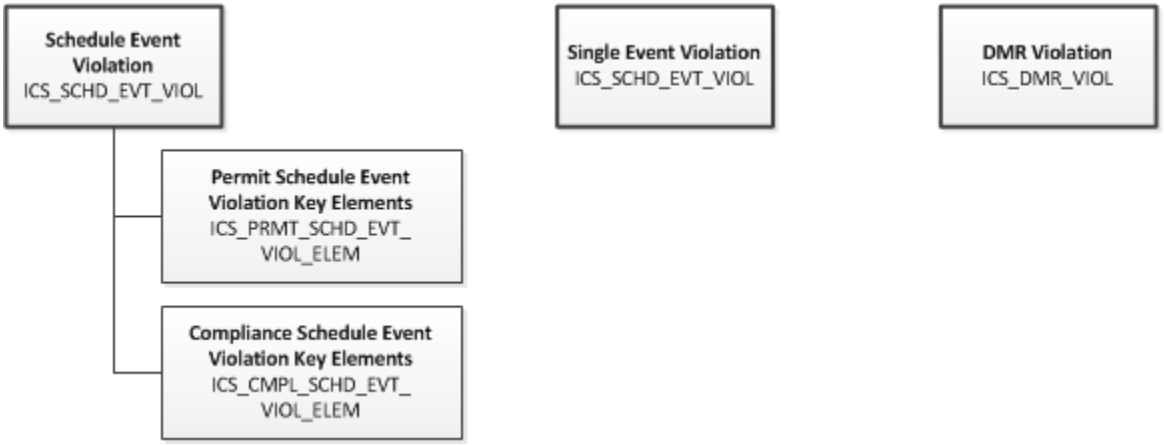
Compliance Monitoring (Inspections)



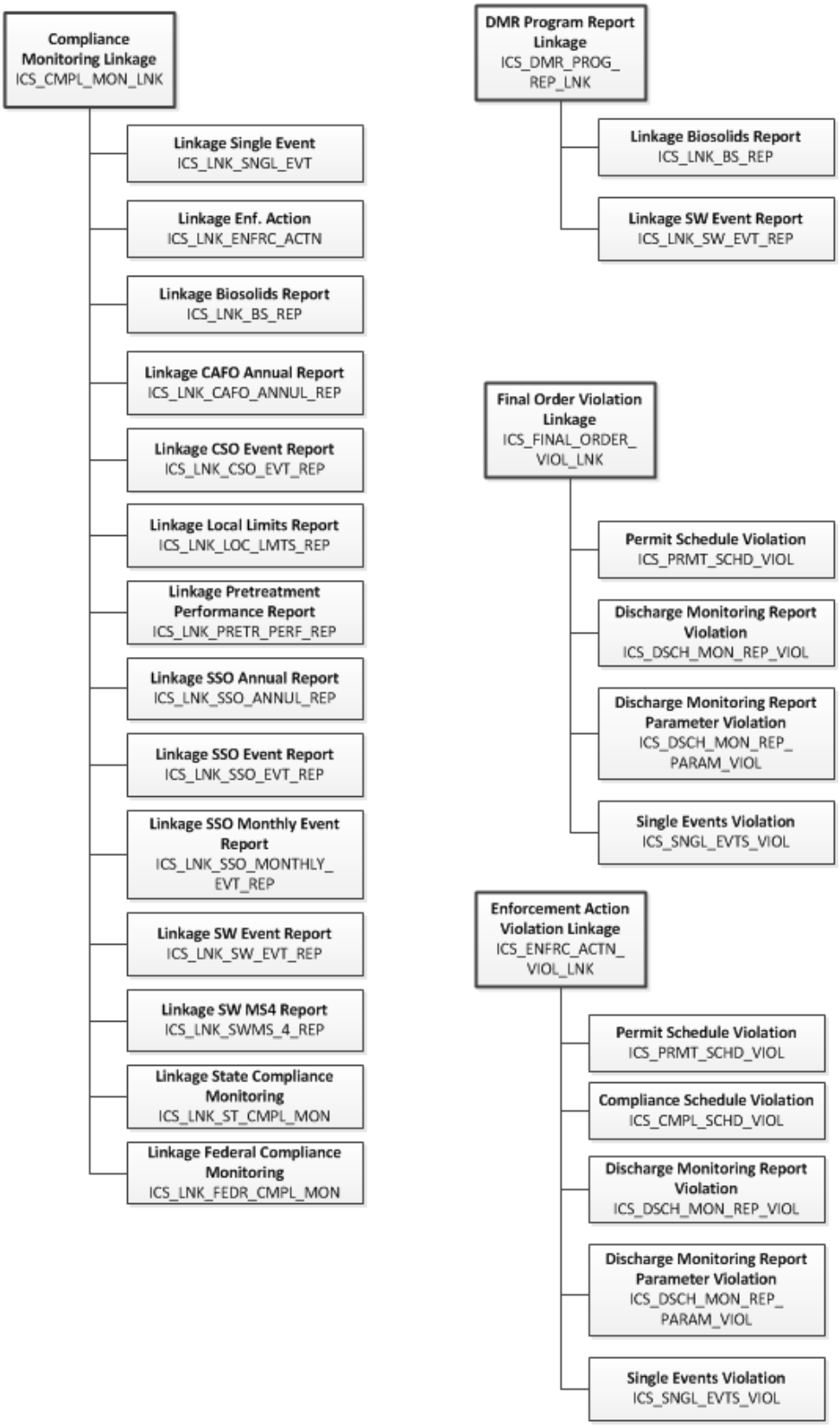
Enforcement



Violations



Linkages


Compliance Monitoring Linkage
ICS_CMPL_MON_LNK
Linkage Single Event
ICS_LNK_SNGL_EVT
Linkage Enf. Action
ICS_LNK_ENFRC_ACTN
Linkage Biosolids Report
ICS_LNK_BS_REP
Linkage CAFO Annual Report
ICS_LNK_CAFO_ANNUL_REP
Linkage CSO Event Report
ICS_LNK_CSO_EVT_REP
Linkage Local Limits Report
ICS_LNK_LOC_LMTS_REP
Linkage Pretreatment Performance Report
ICS_LNK_PRETR_PERF_REP
Linkage SSO Annual Report
ICS_LNK_SSO_ANNUL_REP
Linkage SSO Event Report
ICS_LNK_SSO_EVT_REP
Linkage SSO Monthly Event Report
ICS_LNK_SSO_MONTHLY_EVT_REP
Linkage SW Event Report
ICS_LNK_SW_EVT_REP
Linkage SW MS4 Report
ICS_LNK_SWMS_4_REP
Linkage State Compliance Monitoring
ICS_LNK_ST_CMPL_MON
Linkage Federal Compliance Monitoring
ICS_LNK_FEDR_CMPL_MON
DMR Program Report Linkage
ICS_DMR_PROG_REP_LNK
Linkage Biosolids Report
ICS_LNK_BS_REP
Linkage SW Event Report
ICS_LNK_SW_EVT_REP
Final Order Violation Linkage
ICS_FINAL_ORDER_VIOL_LNK
Permit Schedule Violation
ICS_PRMT_SCHD_VIOL
Discharge Monitoring Report Violation
ICS_DSCH_MON_REP_VIOL
Discharge Monitoring Report Parameter Violation
ICS_DSCH_MON_REP_PARAM_VIOL
Single Events Violation
ICS_SNGL_EVTS_VIOL
Enforcement Action Violation Linkage
ICS_ENFRC_ACTN_VIOL_LNK
Permit Schedule Violation
ICS_PRMT_SCHD_VIOL
Compliance Schedule Violation
ICS_CMPL_SCHD_VIOL
Discharge Monitoring Report Violation
ICS_DSCH_MON_REP_VIOL
Discharge Monitoring Report Parameter Violation
ICS_DSCH_MON_REP_PARAM_VIOL
Single Events Violation
ICS_SNGL_EVTS_VIOL