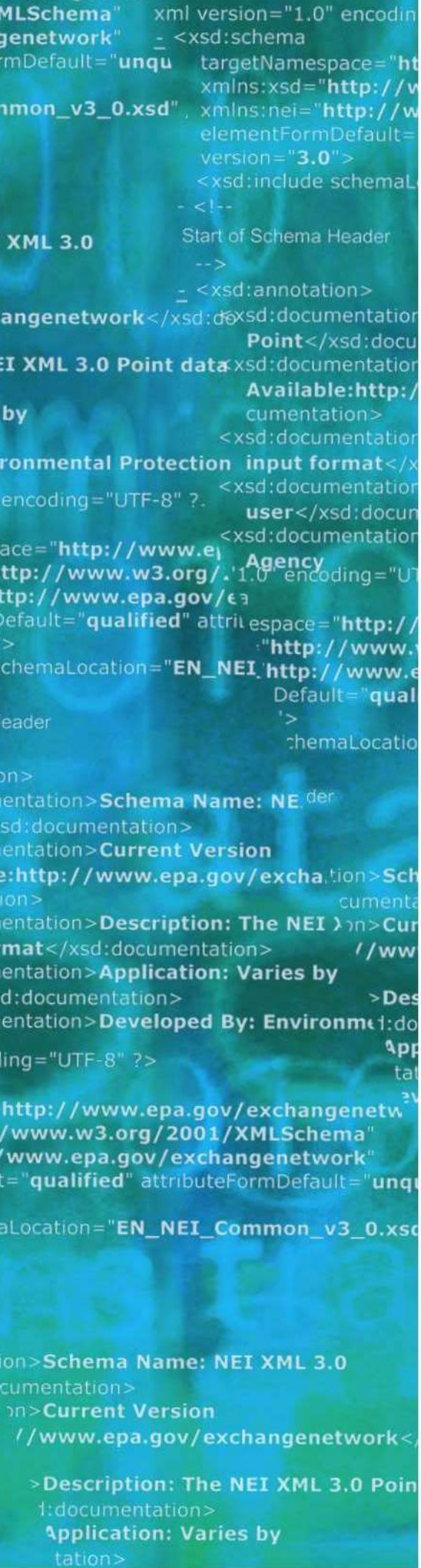


OpenNode2

ICIS-NPDES 5.14 Data Exchange Implementation Guide (.NET)

Revision Date: 08/02/2025

Prepared By:

WINDSOR

Environmental Information Exchange Network

Revision History

Date	Author	Changes
11/20/2014	Windsor	Initial version based on v4.0 implementation guide. Added Appendix C - upgrading from v4 to v5 Updated Appendix A and B to include new/changed payloads for v5
4/14/2015	Windsor	Added documentation for new ICIS Flow Name service parameter Updated screenshots
6/14/2016	Windsor	Update Compliance Monitoring business key field in Appendix A Add missing ics_geo_coord table to Appendix B block diagram
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.
10/10/2017	Windsor	Clarified SQL and Oracle schema deployment instructions.
01/05/2018	Windsor	Modified instruction of view creation to reflect the new approach of bundling all views supporting the ICIS-NPDES data flow into a single script.
4/2/2025	Windsor	Updates for plugin version 5.14 Beta
8/2/2024	Windsor	Updates for plugin version 5.14 procedures and updates to installation procedure

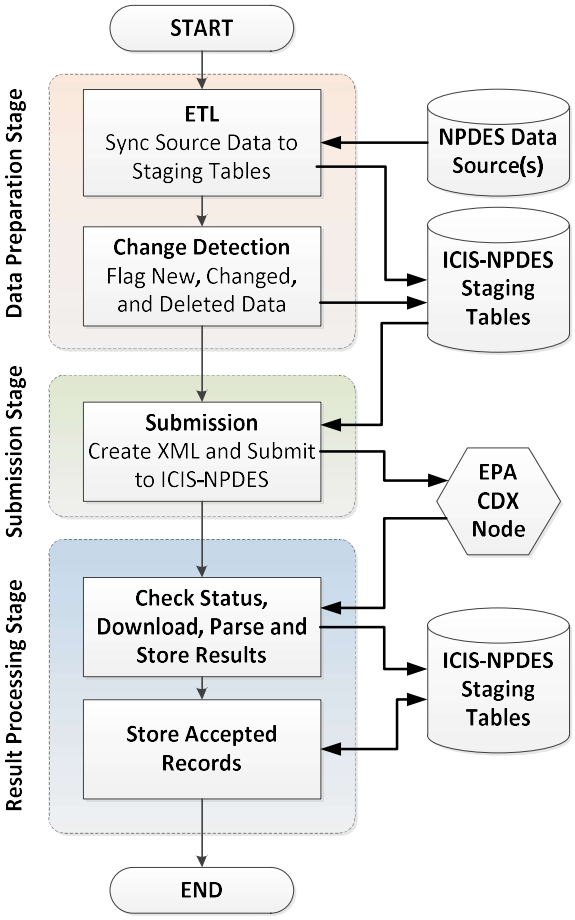
Table of Contents

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 .NET 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. Please contact opennode2@windsorsolutions.com for the full design specification document.

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 agency's NPDES information system. A change detection database stored procedure[1] 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.

[1] The Change detection stored procedure is not included in the Beta release of v5.14

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[2] 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.

[2] The accepted transaction processing procedure is not included in the v5.14 Beta plugin.

Staging Table Architecture

Like most OpenNode2 exchanges, the ICIS-NPDES exchange leverages a series of database staging tables. These staging tables serve as a "parking lot" for data that is ready to be sent via OpenNode2 to EPA. 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[3] 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 an agency-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, reflect 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.

[3] 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:

- 219 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)

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 46 child tables, each relating to a different submission (payload) 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

Custom database Extract, Transformation, and Load (ETL) routines will need to be developed by the implementing agency to load ICIS data from the source database to the OpenNode2 staging database.

The ICIS-NPDES plugin supports two 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:

- Automatic Change Tracking
- Manual Change Tracking

Change tracking is the process of setting the **Transaction Code** on each record in the Staging-Local database. The Transaction Code indicates whether a record is a New, Replace, Delete or Mass Delete operation in ICIS. Records with a blank (null) Transaction Code in the database will not be included in the XML file generated by the node.

Approach 1: Automatic Change Tracking

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. The plugin's built in change detection procedure compares what was staged with what was previously sent and automatically sets the transaction codes for the records that have been added or changed compared to what was previously accepted by ICIS-NPDES.

This approach requires the agency to set up the Staging-ICIS database/schema. This schema is used by the plugin to store copies of accepted transactions returned in the submission processing report from EPA and is used by the plugin procedures to perform the comparison between the data that has been loaded in Staging-Local and what has been previously accepted by ICIS in Staging-ICIS.

Approach #2: Manual Change Tracking

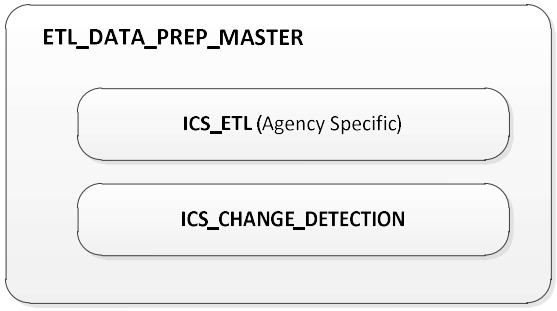
Using this approach, the ETL developer is responsible for setting the transaction codes using their own ETL logic. This approach requires that the state has developed its own mechanism to track which data needs to be added, updated, or removed from ICIS in the batch dataflow.

If this approach is used, the following components can be removed from the staging database:

- 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. **If Automatic Change Tracking is Enabled:** 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).

ETL_DATA_SEND_AS_IS is a version of the parent stored procedure that encapsulates the workflow tracking for sending a submission but does not run the ICS_ETL or ICS_CHANGE_DETECTION. It can be used to create a send a submission of data as it is currently staged in the database, usually for debugging purposes or for use with manual change tracking.

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 48 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 and is not included in the submission. 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 finishes, due to either:
    - When Submission partner is specified: The node processes results from a successful submission. This includes retrieving a "Complete" status from submission partner (CDX), downloading the processing reports, parsing the results, and successfully running the ProcessAcceptedTransactions procedure.
    - When Submission partner is blank, then workflow it set to "Complete" when the submission xml file is generated.
- A workflow is set to the "Failed" status if the plugin receives a "Failed" status from CDX, or if there is an internal error detected while the node is serializing the data. Manual intervention may be necessary to resolve internal errors.
- The workflow status will stay in 'Pending' state until the submission result is returned (Complete or Failed). This is to ensure that no new workflows are created before the processing results from CDX can be consumed and interpreted by the plugin.
- If a submission is stuck in "Pending" *manual intervention will be required* to resolve the workflow before a new workflow can begin. An example is if the submission partner (CDX) fails to return a result report for a submission. To resolve this, the "Pending" workflow needs to be set to 'Complete' or 'Failed' so that the ETL will create a new workflow on the next run. Data will likely be resent as a result.

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 Pending submissions. When a status returns 'Complete' from CDX, the plugin will download and parse the transactions into the ICS_SUBM_RESULTS table.
- **If Automatic Change Tracking is Enabled:** 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.

Scripts are provided for SQL Server and Oracle. SQL Server has two configuration options:

- Oracle – Local and ICIS table sets are setup as separate User Schemas, ICS_FLOW_LOCAL and ICS_FLOW_ICIS
- SQL Server – Two options for configuration:
    - Explicit Database – Local and ICIS table sets are setup in separate databases, named ICS_FLOW_LOCAL and ICS_FLOW_ICIS, using the dbo schema
    - Explicit Schema – Local and ICIS table sets are setup in the same database, in the separate ICS_FLOW_LOCAL and ICS_FLOW_ICIS Schemas.

The database scripts referenced in the instructions below are included in the plugin download zip file.

Oracle Database Deployment

1. Two Oracle schemas are required to support the ICIS NPDES data flow. A third schema can be optionally used if desired to isolate the procedures and data for extra security. Create Oracle schemas named ICS_FLOW_LOCAL, and ICS_FLOW_ICIS and (Optional) ICS_FLOW_LOCAL_USER. 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 (optional)**
        i. CONNECT
        ii. RESOURCE
2. Create a database connection to your Oracle instance as schema owner SYS. Execute the DDL script within the **grants** folder called ICIS_5.14-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 and ICS_FLOW_ICIS. This is the only step that needs to be executed with the elevated SYS connection.
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_5.14-ORA-DDL.sql.
4. 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_5.14-ORA-DDL.sql.

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_5.14-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_5.14-ORA-GET_LIMIT_MONTHS.sql
    b. ICIS_5.14-ORA-MD5_HASH.sql
    c. Optionally, run both of these on ICS_FLOW_ICIS too. This is not necessary for basic operation, but can be handy for debugging and reporting on data sent.
7. Create a database connection to your Oracle instance as schema owner ICS_FLOW_LOCAL. Execute the following SQL script within the **views** folder.
    a. ICIS_5.14-ORA-VIEWS_ICS_FLOW_LOCAL.sql
        i. Creates data change detection views for each submission type. Each view created is utilized by ETL Change Detection processing.
        ii. Creates views that are tied to an error review tool in the OpenNode2 Data Viewer.
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. If this is an update, make sure any custom logic in this procedure is preserved.

    a. ICIS_5.14-ORA-ICS_SET_DATA_HASH.sql
    b. ICIS_5.14-ORA-ICS_CHANGE_DETECTION.sql
    c. ICIS_5.14-ORA-ICS_ETL.sql
    d. ICIS_5.14-ORA-ICS_PROCESS_ACCEPTED_TRANS.sql
    e. * ICIS_5.14-ORA-ICS_DATA_PREP_MASTER.sql
9. (Optional if using ICS_FLOW_LOCAL_USER) Create a database connection to your Oracle instance as schema owner ICS_FLOW_LOCAL. Execute the DDL script within the **grants** folder called ICIS_5.14-ORA-DDL-GRANTS-ICS_FLOW_LOCAL.sql.
10. (Optional if using ICS_FLOW_LOCAL_USER) 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_5.14-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. For Upgrades, update ETL procedures as necessary to work with new 5.14 data:
    a. **ICIS_5.14-ORA-ICS_ETL_payload_stubs.zip**: This archive contains a basic template for each ICIS payload supported in the 5.14 data flow. Locate the stub procedure

provided for the appropriate module and use that as your guide through the data staging process.

b. The staging table path list in the appendix of this document,

c. The Full Batch plugin design specification, and

d. 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 payload types accepted by ICIS-NPDES are inserted by default. To enable or disable payloads, set the ENABLED column to 'Y' or 'N'.

Oracle Upgrade Instructions (From version 5.8 or 5.8.1)

Due to the extensive changes with the 5.14 schema, we have modified the method of upgrading from 5.8 or 5.8.1 for Oracle.

Instead of a script that alters existing tables, the update method consists of 3 steps:

1. Run ICIS_5.14.StashAndDrop5.8Tables.sql : This script will save all 5.8 tables into ones prefixed with I58, and remove the ICS tables from the schema.
2. Run ICIS_5.14-ORA-DDL.sql : Run ICIS_5.14.ICIS_5.14-ORA-DDL.sql to create the ICS 5.14 objects in the schema.
3. Run ICIS_5.14.RestoreData5.8To5.14.sql: Run this to copy data, where it fits, from 5.8 to 5.14.

Only data that fits between 5.8 and 5.14 will be present after upgrade. Specifically, only columns on table paths that are included exactly in both schemas.

Continue with Installation instructions from step 3 (Grants) in order to ensure the latest db objects are present to support the ICIS 5.14 schema.

SQL Server – Explicit Schema Deployment

This section describes staging table setup using a single SQL Server database with two separate schemas setup to hod the Local and ICIS tables. This method is usually preferred for new installations since it allows the database to have any name, but requires the node user login to have a default schema of ICS_FLOW_LOCAL.

1. Create a new database (we will use ICIS_NPDES for example)
2. Create schemas within the database named ICS_FLOW_LOCAL and ICS_FLOW_ICIS.
3. Create a SQL login named (for example) icis_user, with at least Read, Write and Execute permission on the ICIS database. **Set the default schema for this account to ICS_FLOW_LOCAL** in order for the node to access objects in the correct schema by default.
    a. If the db_execute role does not exist on the server, create one with CREATE ROLE db_executor; and GRANT EXECUTE to db_executor; statements.
4. Run the scripts in the **functions** folder to create functions used by the bundled procedures.

5. Run the script in the **views** folder in the ICS_FLOW_LOCAL schema/database. This script is described briefly below:
6. ICIS_5.14-SQLES-VIEWS_ICS_FLOW_LOCAL.sql
    i. Creates data change detection views for each submission type. Each view created is utilized by ETL Change Detection processing.
    ii. Creates views that are tied to an error review tool in the OpenNode2 Data Viewer.
    iii. Views are only used in the ICS_FLOW_LOCAL schema
7. 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. Compile the ICS_DATA_PREP_MASTER script *last* since its dependent on others.
8. The provided ICS_ETL stored procedure is only a shell. Update ICS_ETL to include custom data loading procedures from your agency's NPDES source database(s). Use the following resources to help you design and build the ETL routines (For upgrades, update ETLs as necessary):
9. A list of the table paths is provided in the appendix of this document,
10. Open table ICS_PAYLOAD in the ICS_FLOW_LOCAL schema/database. All payload types accepted by ICIS-NPDES are inserted by default. To enable or disable payloads, set the ENABLED column to 'Y' or 'N'.

SQL Explicit Schema Upgrade Instructions (from version 5.8.1)

Run the table upgrade script for each schema against the appropriate schema to update the tables from 5.8.1 to 5.14

Then, follow instructions for setup starting with step 5,Views to make sure you have the latest version of all database objects.

SQL Server – Explicit Database Deployment

This section describes staging table setup using two separate SQL Server databases, one each for the Local and ICIS table sets. This was the original configuration supported by the plugin. This method requires exact names of ICS_FLOW_LOCAL and ICS_FLOW_ICIS for the databases.

1. Create two new databases named ICS_FLOW_LOCAL and ICS_FLOW_ICIS.
2. Run the data definition language (DDL) script in the **staging_table_ddl** folder to create a set of data staging tables in the ICS_FLOW_LOCAL schema/database. Run the same script again to create the same staging tables in the ICS_FLOW_ICIS schema/database.
    a. Optionally, drop the ICS_SUBM_RESULTS and ICS_SUBM_TRACK tables from the ICS_FLOW_ICIS schema/database since they are not used in that context.

3. Create a SQL login named (for example) icis_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 for icis_user named *db_execute.* Grant execute to this new role allowing for stored procedure execution rights.
    b. Map the icis_user user to the *db_datareader*, *db_datawriter,* and *db_execute* roles in the ICS_FLOW_LOCAL database.
    c. Map the icis_user user to the *db_datareader* and *db_datawriter* roles in the ICS_FLOW_ICIS database. Note, the *db_execute* role is not needed in ICS_FLOW_ICIS.
4. Run the scripts in the **functions** folder to create functions used by the bundled procedures. Functions must be created on ICS_FLOW_LOCAL. Creation on ICS_FLOW_ICIS is optional but useful during debugging.
5. Run the script in the **views** folder in the ICS_FLOW_LOCAL database. This script is described briefly below:
    a. ICIS_5.14-SQLES-VIEWS_ICS_FLOW_LOCAL.sql
        i. Creates data change detection views for each submission type. Each view created is utilized by ETL Change Detection processing.
        ii. Creates views that are tied to an error review tool in the OpenNode2 Data Viewer.
6. Run the scripts in the **procedures** folder on the ICS_FLOW_LOCAL schema/database.
    a. **For New Install ONLY**, run ICS_ETL (For updates, keep and modify existing ETL procedures as necessary)
    b. Run the following to update 'stock' procedures for new installs AND updates. (If the existing procedures contain custom code, then this should be preserved and added to the new procedures as needed)

    - ICS_SET_HASHES
    - ICS_PROCESS_ACCEPTED_TRANS
    - ICS_CHANGE_DETECTION
    - ICS_DATA_PREP_MASTER (preserve any custom logic first in upgrades)
    - ICS_DATA_SEND_AS_IS

- **ICS_CLEAR_NEW_ELEMENTS** is provided to remove elements in the ICIS-NPDES 5.14 schema that are causing error when included as of release in July 2025. This procedure will remove this data. It is a temporary patch until ICIS is fixed to accept these new fields. Until ICIS is fixed, call this procedure at the end of the ICS_ETL procedure.

7. The provided ICS_ETL stored procedure is only a shell. Update ICS_ETL to include custom data loading procedures from your agency's NPDES source database(s). Use the following resources to help you design and build the ETL routines (For upgrades, update ETLs as necessary):
    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, and enable payloads for which data is loaded. All 48 payload types accepted by ICIS-NPDES are inserted by default. To enable or disable payloads, set the ENABLED column to 'Y' or 'N'.

SQL Explicit Database Upgrade Instructions (from version 5.8.1)

Run the table upgrade script for against each database to update the tables from 5.8.1 to 5.14

Then, follow instructions for setup starting with step 5,Views to make sure you have the latest version of all database objects.

Permit Date Seeding for New Installations

If the agency already has data in ICIS-NPDES, then data will get overwritten with data from the flow on initial sends. For most payloads, this is harmless and a side effect change detection.

For Permit flows (BasicPermit, GeneralPermit and MasterGeneralPermit) and for Permit Terminations, this can cause errors since a permit version (As defined by a unique effective date) can only be sent once. Therefore, it may be necessary to 'seed' the ICS_FLOW_ICIS base tables for these payloads directly by inserting records with the Permit Number, Issuance, Effective and Expiration dates. The Permit Termination payload requires the Termination date of terminated permits to prevent resend errors.

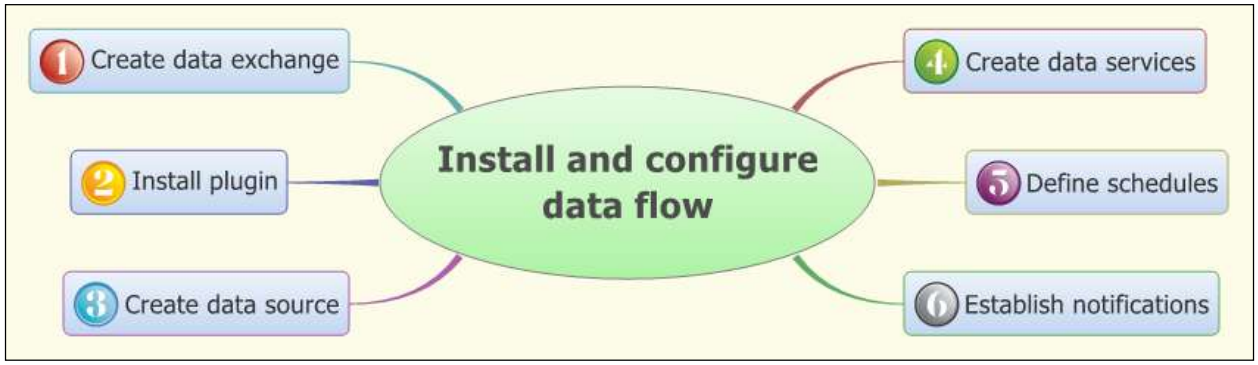
Reports for permit dates are usually sourced from the ICIS BO reporting site for which the node is submitting data (Prod, Stage or Test). A report should include the Permit Number, Issue, Effective and Expiration Dates, Permit Type (for determining which payload to load it to) and Termination Date to seed the ICS_PRMT_TERM table for terminated permits.

Only the Base tables need to be seeded, and only the essential permit identifying information with dates need to be filled out. The permit will then flow along with other data, and data will be filled out completely upon acceptance by 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 Exchange** button. The Manage Data Exchange screen will be displayed:

3. Type *ICIS-NPDES* in the **Name** field.
4. 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.
5. Optionally, type any valid URL in the **Web Info** field. This value is not used by OpenNode2.
6. 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.
7. Click **Save** to save the data exchange.

Install ICIS-NPDES Full Batch Plugin

Once the ICIS-NPDES data exchange has been created, the next step is to upload the ICIS-NPDES plugin 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 ICIS-NPDES exchange is located.
2. Click the **Add Service** button located just beneath the ICIS-NPDES 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 navigation block.

4. Click the **Browse** button which is located to the right of the **Plugin** field.
5. Locate and select the compressed (zipped) file containing the code component for the ICIS-NPDES Full Batch plugin you created in step 2 above.
6. Select *ICIS-NPDES* from the **Exchange** drop-down menu. If *ICIS-NPDES* is not available, ensure that the previous step was completed (*Create ICIS-NPDES Data Exchange*).
7. Click the **Upload** button to install the ICIS-NPDES 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.

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/ngn-enws20/services/NetworkNode2ServiceConditionalMTOM

CDX Prod: https://cdxnodengn.epa.gov/ngn-enws20/services/NetworkNode2Service

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 v2.0* 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:

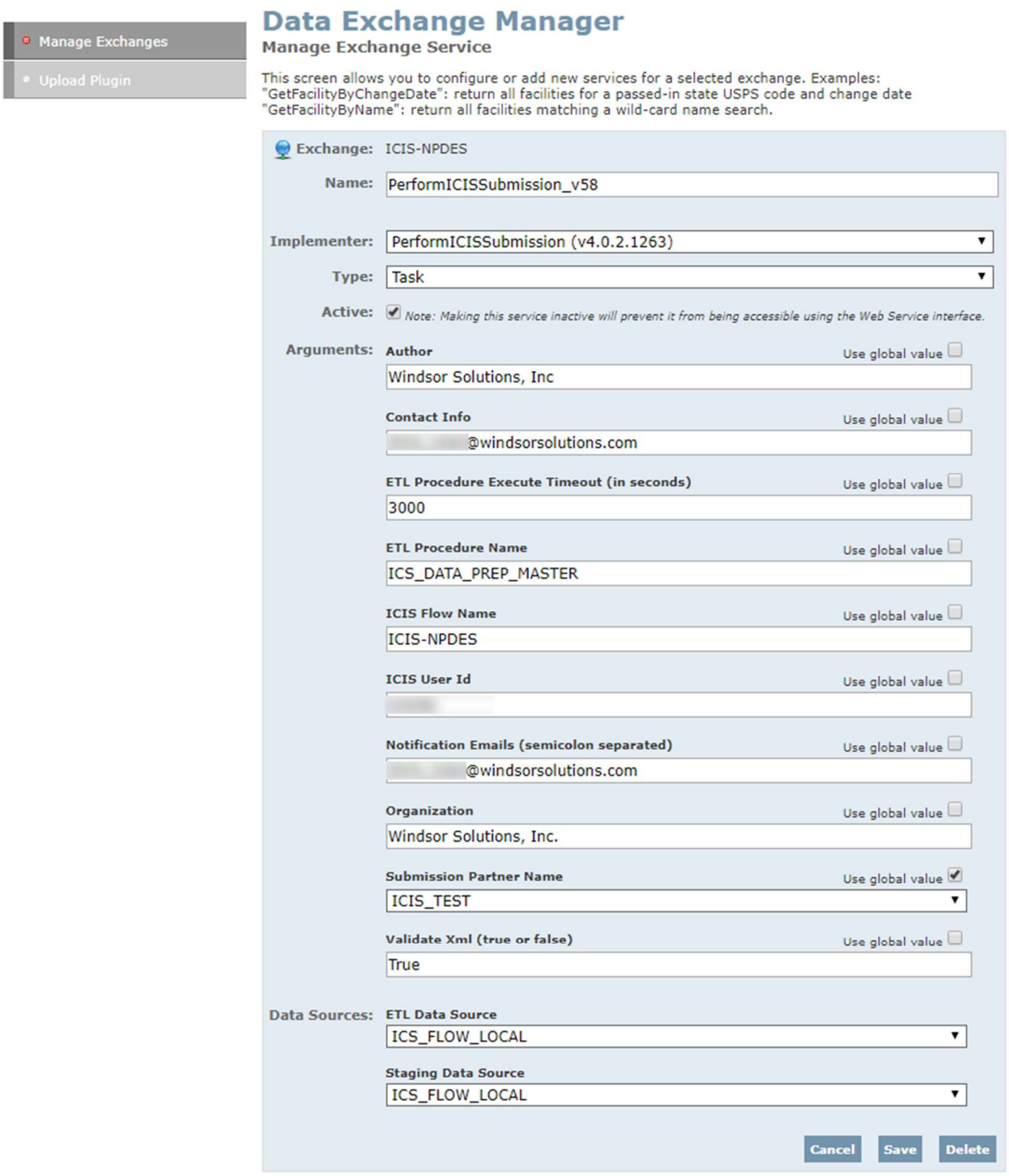
- **PerformICISSubmission** - responsible for sending data to ICIS-NPDES from the staging environment.
- **GetICISStatusAndProcessReports** - responsible for checking the status of previous submissions and, if processing is completed, downloading and parsing the Accepted and Rejected Transactions reports 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 Service** button located just beneath the ICIS-NPDES data exchange entry. The following page will be displayed to allow a new data service to be added.



2. In the **Service Name** field, enter "PerformICISSubmission".
3. Select the "PerformICISSubmission" entry from the **Implementer** drop-down menu.

   Note: When the implementer is selected, several arguments will appear. The Node Admin will obtain these properties directly from the ICIS-NPDES plugin.

4. From the **Type** drop-down menu, select "Task". The Scheduler will use this service to compose XML for submission to EPA.

5. Enable the service by checking the **Active** checkbox.
6. 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.
7. 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. Options supplied with the plugin:
    - ICS_DATA_PREP_MASTER – this procedure runs the full flow sequence, including tracking flow, calling ETL (procedure ICS_ETL), and Change detection.
    - ICS_DATA_SEND_AS_IS – this procedure tracks the flow but does not run ETL or change detection, and is therefore useful to flow manually staged data.
    - (blank) – The plugin will serialize data in the tables, but no tracking, ETL or change detection (not often used)

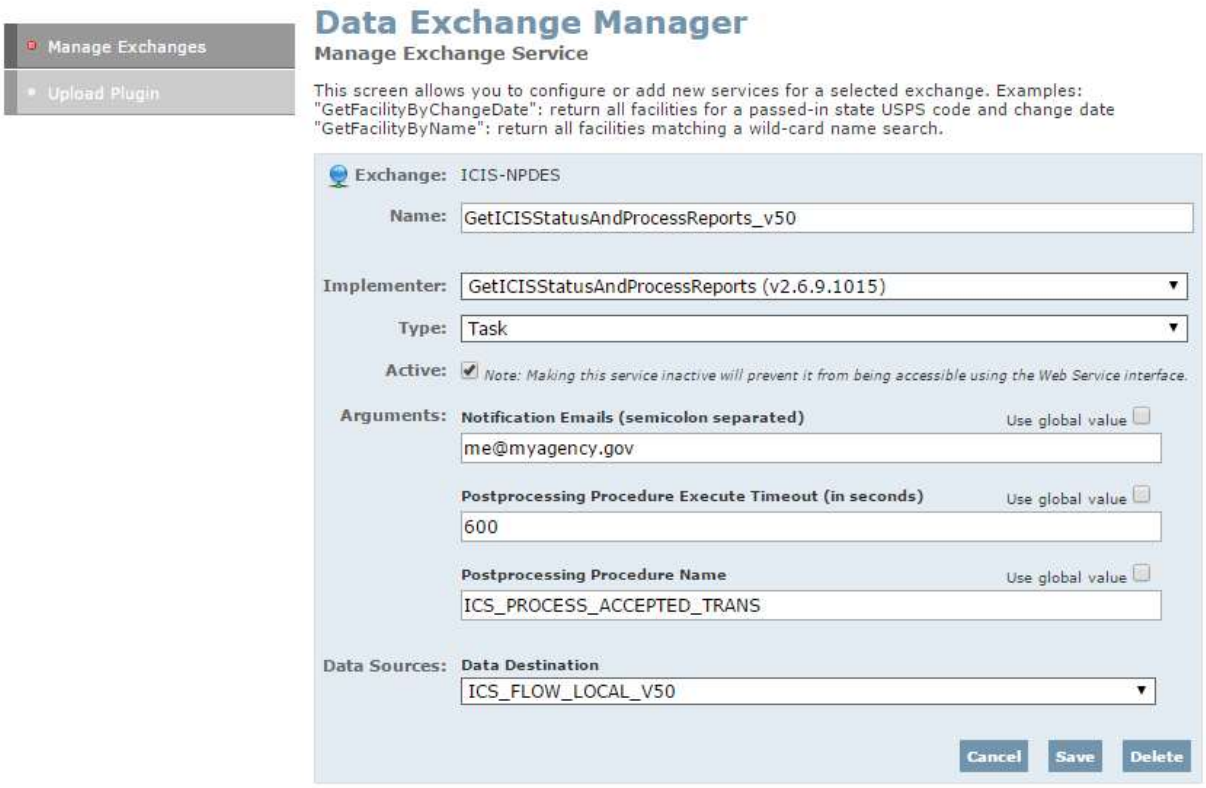
    When procedure is configured, the **ETL Procedure Execute Timeout** should also be set to a time that includes execution of the ETL and change detection (usually 3600).
8. Set the **ICIS Flow Name** to *ICIS-NPDES* . for flow to Production or Stage. Set the **ICIS Flow Name** to *ICIS-AIR* if sending to Test. Correct capitalization is important!
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 Emails** 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 **Staging Data Source**, 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 Service** button located just beneath 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. From the **Type** drop-down menu, select "Task". This makes the service visible to the Scheduler component.
6. Enable the service by checking the **Active** checkbox.
7. If desired, add one or more semicolon-delimited email addresses in the **Notification Emails** 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.
8. Set the **Postprocessing Procedure Execute Timeout** long enough to ensure completion of ICS_PROCESS_ACCEPTED_TRANS (usually 3600).
9. **If Enabling Automatic Change Tracking:** Set the **Postprocessing 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. For the **Data Destination**, 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.
11. 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 Schedule** 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. 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.
6. Set the **Frequency** to the desired interval. Daily submissions are typical.
7. The **Run Time** should be set for the desired time of day to submit the file.
8. In the **Data Source** area, check the radio button labeled **Results of local service execution**.
9. In the **From** dropdown box, select the value "PerformICISSubmission". This informs the schedule to use the selected ICIS-NPDES service as the data source for the submission.
10. No changes are needed to the **Additional Parameters** area.
11. 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 or two submissions will occur and the transaction ID tracking will become out of sync between the plugin and the node's transaction tracking.

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

Create the "Get ICIS Status and Process 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.
6. Set the **Frequency** to the desired interval. The frequency should be the same as the submission schedule or more frequent.
7. The **Run Time** should be set for the desired time of day. 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 7 in the previous schedule.

8. In the **Data Source** area, check the radio button labeled **Results of local service execution**.
9. In the **From** dropdown box, select the value "GetICISStatusAndProcessReports". This informs the schedule to use the selected ICIS-NPDES service as the processor for the task.
10. In the **Result Process** area, check the radio button labeled **None**.
11. 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: v5.8 vs. v5.14 Payload Comparison

The following table provides a comparison between the payloads available in the previous v5.8 plugin versus the new v5.14 plugin.

Root Staging Table	Payload v5.8	Payload v5.14	Status	Comments
ICS_CAFO_ANNUL_PROG_REP		CAFOAnnualProgramReportSubmission	Added	Replaces CAFOAnnualReportSubmission
ICS_CSO_LONG_TERM_CONTROL_PLAN		CSOLongTermControlPlanSubmission	Added	
ICS_CWA_316B_PROG_REP		CWA316bProgramReportSubmission	Added	
ICS_COLL_SYSTM_PRMT		CollectionSystemPermitSubmission	Added	Replaces CSOPermitSubmission
ICS_COPY_MGPMS_4_REQ		CopyMGPMS4RequirementSubmission	Added	
ICS_NPDES_VARIANCE_PRMT		NPDESVariancePermitSubmission	Added	
ICS_POTW_TRTMNT_TECHNOLOGY_PRMT		POTWTreatmentTechnologyPermitSubmission	Added	
ICS_PRETR_PROG_REP		PretreatmentProgramReportSubmission	Added	Replaces LocalLimitsProgramReportSubmission and PretreatmentPerformanceSummarySubmission
ICS_SWMS_4_ANNUL_PROG_REP		SWMS4AnnualProgramReportSubmission	Added	Replaces SWMS4ProgramReportSubmission
ICS_SWMS_4_PRMT		SWMS4PermitSubmission	Added	Replaces SWMS4LargePermitSubmission and SWMS4SmallPermitSubmission
ICS_SEWER_OVRFLW_BYPASS_EVT_REP		SewerOverflowBypassEventReportSubmission	Added	Replaces 4 old CSO/SSO submissions
ICS_BS_ANNUL_PROG_REP		BiosolidsAnnualProgramReportSubmission	Added	Replaces BiosolidsProgramReportSubmission BASE TABLE NAME is REUSED!
ICS_BASIC_PRMT	BasicPermitSubmission	BasicPermitSubmission	Modified	
ICS_BS_PRMT	BiosolidsPermitSubmission	BiosolidsPermitSubmission	Modified	
ICS_CAFO_PRMT	CAFOPermitSubmission	CAFOPermitSubmission	Modified	
ICS_CMPL_MON_LNK	ComplianceMonitoringLinkageSubmission	ComplianceMonitoringLinkageSubmission	Modified	
ICS_CMPL_MON	ComplianceMonitoringSubmission	ComplianceMonitoringSubmission	Modified	
ICS_GNRL_PRMT	GeneralPermitSubmission	GeneralPermitSubmission	Modified	
ICS_MASTER_GNRL_PRMT	MasterGeneralPermitSubmission	MasterGeneralPermitSubmission	Modified	

ICS_PRMT_FEATR	PermittedFeatureSubmission	PermittedFeatureSubmission	Modified	
ICS_PRETR_PRMT	PretreatmentPermitSubmission	PretreatmentPermitSubmission	Modified	
ICS_SW_CNST_PRMT	SWConstructionPermitSubmission	SWConstructionPermitSubmission	Modified	
ICS_SW_INDST_PRMT	SWIndustrialPermitSubmission	SWIndustrialPermitSubmission	Modified	
ICS_BS_PROG_REP	BiosolidsProgramReportSubmission		Removed	
ICS_CAFO_ANNUL_REP	CAFOAnnualReportSubmission		Removed	
ICS_CSO_EVT_REP	CSOEventReportSubmission		Removed	
ICS_CSO_PRMT	CSOPermitSubmission		Removed	
ICS_DMR_PROG_REP_LNK	DMRProgramReportLinkageSubmission		Removed	(no longer relavant)
ICS_LOC_LMTS_PROG_REP	LocalLimitsProgramReportSubmission		Removed	
ICS_PRETR_PERF_SUMM	PretreatmentPerformanceSummarySubmission		Removed	
ICS_SSO_ANNUL_REP	SSOAnnualReportSubmission		Removed	
ICS_SSO_EVT_REP	SSOEventReportSubmission		Removed	
ICS_SSO_MONTHLY_EVT_REP	SSOMonthlyEventReportSubmission		Removed	
ICS_SW_EVT_REP	SWEventReportSubmission		Removed	
ICS_SWMS_4_LARGE_PRMT	SWMS4LargePermitSubmission		Removed	
ICS_SWMS_4_PROG_REP	SWMS4ProgramReportSubmission		Removed	
ICS_SWMS_4_SMALL_PRMT	SWMS4SmallPermitSubmission		Removed	
ICS_FRML_ENFRC_ACTN	FormalEnforcementActionSubmission	FormalEnforcementActionSubmission	Unchanged	
ICS_CMPL_SCHD	CopyMGPLimitSetSubmission	CopyMGPLimitSetSubmission	Unchanged	
ICS_DMR_VIOL	DMRViolationSubmission	DMRViolationSubmission	Unchanged	
ICS_DSCH_MON_REP	DischargeMonitoringReportSubmission	DischargeMonitoringReportSubmission	Unchanged	
ICS_EFFLU_TRADE_PRTNER	EffluentTradePartnerSubmission	EffluentTradePartnerSubmission	Unchanged	
ICS_FINAL_ORDER_VIOL_LNK	FinalOrderViolationLinkageSubmission	FinalOrderViolationLinkageSubmission	Unchanged	
ICS_HIST_PRMT_SCHD_EVTS	HistoricalPermitScheduleEventsSubmission	HistoricalPermitScheduleEventsSubmission	Unchanged	
ICS_INFRML_ENFRC_ACTN	InformalEnforcementActionSubmission	InformalEnforcementActionSubmission	Unchanged	
ICS_LMT_SET	LimitSetSubmission	LimitSetSubmission	Unchanged	
ICS_LMTS	LimitsSubmission	LimitsSubmission	Unchanged	
ICS_NARR_COND_SCHD	NarrativeConditionScheduleSubmission	NarrativeConditionScheduleSubmission	Unchanged	
ICS_POTW_PRMT	POTWPermitSubmission	POTWPermitSubmission	Unchanged	

ICS_PARAM_LMTS	ParameterLimitsSubmission	ParameterLimitsSubmission	Unchanged	
ICS_PRMT_REISSU	PermitReissuanceSubmission	PermitReissuanceSubmission	Unchanged	
ICS_PRMT_TERM	PermitTerminationSubmission	PermitTerminationSubmission	Unchanged	
ICS_PRMT_TRACK_EVT	PermitTrackingEventSubmission	PermitTrackingEventSubmission	Unchanged	
ICS_SW_INDST_ANNUL_REP	SWIndustrialAnnualReportSubmission	SWIndustrialAnnualReportSubmission	Unchanged	
ICS_SCHD_EVT_VIOL	ScheduleEventViolationSubmission	ScheduleEventViolationSubmission	Unchanged	
ICS_SNGL_EVT_VIOL	SingleEventViolationSubmission	SingleEventViolationSubmission	Unchanged	
ICS_UNPRMT_FAC	UnpermittedFacilitySubmission	UnpermittedFacilitySubmission	Unchanged	
ICS_CMPL_SCHD	ComplianceScheduleSubmission	ComplianceScheduleSubmission	Unchanged	
ICS_ENFRC_ACTN_MILESTONE	EnforcementActionMilestoneSubmission	EnforcementActionMilestoneSubmission	Unchanged	
ICS_ENFRC_ACTN_VIOL_LNK	EnforcementActionViolationLinkageSubmission	EnforcementActionViolationLinkageSubmission	Unchanged	

Appendix B - Table/Module Diagrams

The following sections include diagrams showing the data hierarchy and mapping between the OpenNode2 ICIS-NPDES staging tables and the corresponding XML element name.

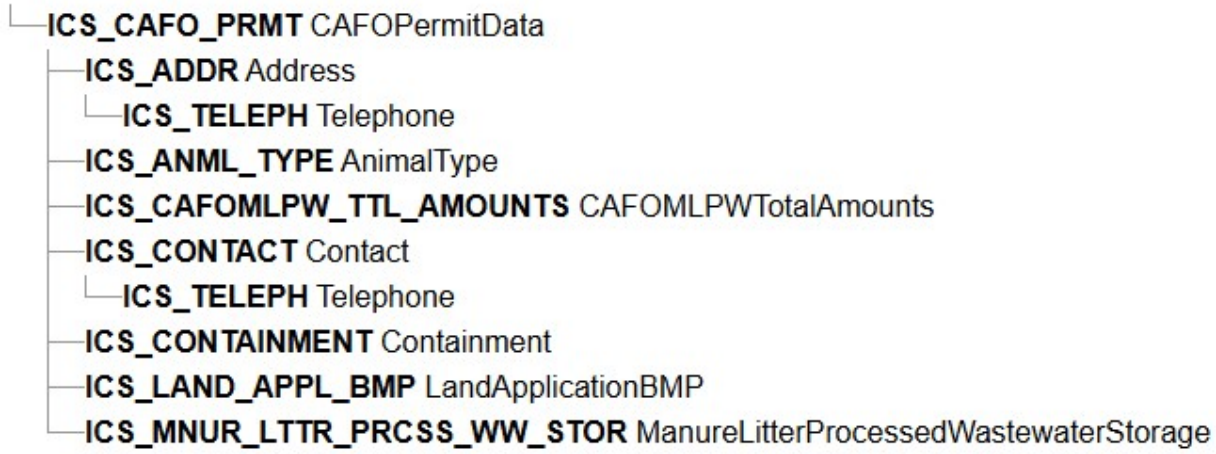
BasicPermitSubmission

- **ICS_BASIC_PRMT** BasicPermitData
  - **ICS_ADDR** Address
    - **ICS_TELEPH** Telephone
  - **ICS_ASSC_PRMT** AssociatedPermit
  - **ICS_CMPL_TRACK_STAT** ComplianceTrackingStatus
  - **ICS_CONTACT** Contact
    - **ICS_TELEPH** Telephone
  - **ICS_EFFLU_GUIDE** EffluentGuidelineCode
  - **ICS_FAC** Facility
    - **ICS_ADDR** Address
      - **ICS_TELEPH** Telephone
    - **ICS_CONTACT** Contact
      - **ICS_TELEPH** Telephone
    - **ICS_FAC_CLASS** FacilityClassification
    - **ICS_GEO_COORD** GeographicCoordinates
    - **ICS_NAICS_CODE** NAICSCodeDetails
    - **ICS_ORIG_PROGS** OriginatingProgramsCode
    - **ICS_PLCY** PolicyCode
    - **ICS_SIC_CODE** SICCodeDetails
  - **ICS_NAICS_CODE** NAICSCodeDetails
  - **ICS_NPDES_DAT_GRP_NUM** NPDESDataGroupNumberCode
  - **ICS_OTHR_PRMTS** OtherPermits
  - **ICS_REP_NON_CMPL_STAT** ReportableNonComplianceStatus
  - **ICS_RESIDUAL_DESGN_DTRMN** ResidualDesignationDetermination
  - **ICS_SIC_CODE** SICCodeDetails
  - **ICS_SIU_DESGN_TYPE** SIUDesignationType

BiosolidsPermitSubmission

- **ICS_BS_PRMT** BiosolidsPermitData
  - **ICS_BS_FAC_TRTMNT** BiosolidsFacilityTreatment
  - **ICS_BS_FAC_TYPE** BiosolidsFacilityType
  - **ICS_PRMT_BS_MGMT_PRACTICE** PermitBiosolidsManagementPracticeData
    - **ICS_BS_OFF_SITE_HNDLR_APPLIER_PREPR** BiosolidsOffSiteHandlerApplierPreparer
      - **ICS_ADDR** Address
        - **ICS_TELEPH** Telephone
      - **ICS_CONTACT** Contact
        - **ICS_TELEPH** Telephone
    - **ICS_PATHOGEN_REDUCTION_TYPE** PathogenReductionTypeCode
    - **ICS_VECTOR_A_REDUCTION_TYPE** VectorAttractionReductionTypeCode

CAFOPermitSubmission



GeneralPermitSubmission

- **ICS_GNRL_PRMT** GeneralPermitData
  - **ICS_ADDR** Address
    - **ICS_TELEPH** Telephone
  - **ICS_ASSC_PRMT** AssociatedPermit
  - **ICS_CMPL_TRACK_STAT** ComplianceTrackingStatus
  - **ICS_CONTACT** Contact
    - **ICS_TELEPH** Telephone
  - **ICS_EFFLU_GUIDE** EffluentGuidelineCode
  - **ICS_FAC** Facility
    - **ICS_ADDR** Address
      - **ICS_TELEPH** Telephone
    - **ICS_CONTACT** Contact
      - **ICS_TELEPH** Telephone
    - **ICS_FAC_CLASS** FacilityClassification
    - **ICS_GEO_COORD** GeographicCoordinates
    - **ICS_NAICS_CODE** NAICSCodeDetails
    - **ICS_ORIG_PROGS** OriginatingProgramsCode
    - **ICS_PLCY** PolicyCode
    - **ICS_SIC_CODE** SICCodeDetails
  - **ICS_NAICS_CODE** NAICSCodeDetails
  - **ICS_NPDES_DAT_GRP_NUM** NPDESDataGroupNumberCode
  - **ICS_OTHR_PRMTS** OtherPermits
  - **ICS_REP_NON_CMPL_STAT** ReportableNonComplianceStatus
  - **ICS_RESIDUAL_DESGN_DTRMN** ResidualDesignationDetermination
  - **ICS_SIC_CODE** SICCodeDetails

MasterGeneralPermitSubmission

- **ICS_MASTER_GNRL_PRMT** MasterGeneralPermitData
  - **ICS_ASSC_PRMT** AssociatedPermit
  - **ICS_CONTACT** Contact
    - **ICS_TELEPH** Telephone
  - **ICS_NAICS_CODE** NAICSCodeDetails
  - **ICS_OTHR_PRMTS** OtherPermits
  - **ICS_PRMT_COMP_TYPE** PermitComponentTypeCode
  - **ICS_RESIDUAL_DESGN_DTRMN** ResidualDesignationDetermination
  - **ICS_SIC_CODE** SICCodeDetails

PermitReissuanceSubmission

- **ICS_PRMT_REISSU** PermitReissuanceData

PermitTerminationSubmission

- **ICS_PRMT_TERM** PermitTerminationData

PermitTrackingEventSubmission

- **ICS_PRMT_TRACK_EVT** PermitTrackingEventData

POTWPermitSubmission

- **ICS_POTW_PRMT** POTWPermitData
  - **ICS_SATL_COLL_SYSTM** SatelliteCollectionSystem

PretreatmentPermitSubmission

- **ICS_PRETR_PRMT** PretreatmentPermitData
  - **ICS_CONTACT** Contact
    - **ICS_TELEPH** Telephone
  - **ICS_PRETR_PROG_MOD** PretreatmentProgramModification

SWConstructionPermitSubmission

- **ICS_SW_CNST_PRMT** SWConstructionPermitData
  - **ICS_ADDR** Address
    - **ICS_TELEPH** Telephone
  - **ICS_CNST_SITE_LIST** ConstructionSiteList
    - **ICS_CNST_SITE** ConstructionSiteCode
  - **ICS_CONTACT** Contact
    - **ICS_TELEPH** Telephone
  - **ICS_GPCF_NOTICE_OF_INTENT** GPCFNoticeOfIntent
    - **ICS_SUBSECTOR_CODE_PLUS_DESC** SubsectorCodePlusDescription
  - **ICS_PROPOSED_CNST_SW_BM_PS** ProposedConstructionStormwaterBMPs
  - **ICS_PROPOSED_POST_CNST_SW_BM_PS** ProposedPostConstructionStormwaterBMPs
  - **ICS_TRTMNT_CHEMS_LIST** TreatmentChemicalsList

SWIndustrialPermitSubmission

- **ICS_SW_INDST_PRMT** SWIndustrialPermitData
  - **ICS_ADDR** Address
    - **ICS_TELEPH** Telephone
  - **ICS_CONTACT** Contact
    - **ICS_TELEPH** Telephone
  - **ICS_GPCF_NO_EXPOSURE** GPCFNoExposure
  - **ICS_GPCF_NOTICE_OF_INTENT** GPCFNoticeOfIntent
    - **ICS_SUBSECTOR_CODE_PLUS_DESC** SubsectorCodePlusDescription
  - **ICS_GPCF_NOTICE_OF_TERM** GPCFNoticeOfTermination
  - **ICS_PROPOSED_INDST_SW_BM_PS** ProposedIndustrialStormwaterBMPs

UnpermittedFacilitySubmission

- **ICS_UNPRMT_FAC** UnpermittedFacilityData
  - **ICS_ADDR** Address
    - **ICS_TELEPH** Telephone
  - **ICS_CONTACT** Contact
    - **ICS_TELEPH** Telephone
  - **ICS_FAC_CLASS** FacilityClassification
  - **ICS_GEO_COORD** GeographicCoordinates
  - **ICS_NAICS_CODE** NAICSCodeDetails
  - **ICS_ORIG_PROGS** OriginatingProgramsCode
  - **ICS_PLCY** PolicyCode
  - **ICS_PRMT_COMP_TYPE** PermitComponentTypeCode
  - **ICS_SIC_CODE** SICCodeDetails

HistoricalPermitScheduleEventsSubmission

- **ICS_HIST_PRMT_SCHD_EVTS** HistoricalPermitScheduleEventsData

NarrativeConditionScheduleSubmission

- **ICS_NARR_COND_SCHD** NarrativeConditionScheduleData
  - **ICS_PRMT_SCHD_EVT** PermitScheduleEvent

PermittedFeatureSubmission

- **ICS_PRMT_FEATR** PermittedFeatureData
  - **ICS_ADDR** Address
    - **ICS_TELEPH** Telephone
  - **ICS_CONTACT** Contact
    - **ICS_TELEPH** Telephone
  - **ICS_COOLING_WTR_INTAKE_STRCT_INFO** CoolingWaterIntakeStructureInformation
    - **ICS_COOLING_WTR_INTAKE_STRCT_CMPL_METHOD** CoolingWaterIntakeStructureComplianceMethod
  - **ICS_GEO_COORD** GeographicCoordinates
  - **ICS_POLUT_LIST** PollutantList
    - **ICS_IMPAIRED_WTR_POLLUTANTS** ImpairedWaterPollutants
    - **ICS_TMDL_POLLUTANTS** TMDLPollutants
      - **ICS_TMDL_POLUT** TMDLPollutantCode
  - **ICS_PRMT_FEATR_CHAR** PermittedFeatureCharacteristics
  - **ICS_PRMT_FEATR_TRTMNT_TYPE** PermittedFeatureTreatmentTypeCode

LimitSetSubmission

- **ICS_LMT_SET** LimitSetData
  - **ICS_LMT_SET_MONTHS_APPL** MonthTextType
  - **ICS_LMT_SET_SCHD** LimitSetSchedule
  - **ICS_LMT_SET_STAT** LimitSetStatus

LimitsSubmission

- **ICS_LMTS** LimitsData
  - **ICS_MN_LMT_APPLIES** MonthTextType
  - **ICS_NUM_COND** NumericCondition

ParameterLimitsSubmission

- **ICS_PARAM_LMTS** ParameterLimitsData
  - **ICS_LMT** Limit
    - **ICS_MN_LMT_APPLIES** MonthTextType
    - **ICS_NUM_COND** NumericCondition

DischargeMonitoringReportSubmission

- **ICS_DSCH_MON_REP** DischargeMonitoringReportData
  - **ICS_CO_DSPL_SITE** CoDisposalSite
  - **ICS_INCIN** Incinerator
  - **ICS_LAND_APPL_SITE** LandApplicationSite
    - **ICS_CROP_TYPES_HARVESTED** CropTypesHarvested
    - **ICS_CROP_TYPES_PLANTED** CropTypesPlanted
  - **ICS_REP_PARAM** ReportParameter
    - **ICS_NUM_REP** NumericReport
  - **ICS_SURF_DSPL_SITE** SurfaceDisposalSite

ComplianceMonitoringSubmission

- **ICS_CMPL_MON** ComplianceMonitoringData
  - **ICS_CAFO_INSP** CAFOInspection
    - **ICS_ANML_TYPE** AnimalType
    - **ICS_CAFO_INSP_VIOL_TYPE** CAFOInspectionViolationTypeCode
    - **ICS_CONTAINMENT** Containment
    - **ICS_LAND_APPL_BMP** LandApplicationBMP
    - **ICS_MNUR_LTTR_PRCSS_WW_STOR** ManureLitterProcessedWastewaterStorage
  - **ICS_CMPL_INSP_TYPE** ComplianceInspectionTypeCode
  - **ICS_CMPL_MON_ACTN_REASON** ComplianceMonitoringActionReasonCode
  - **ICS_CMPL_MON_AGNCY_TYPE** ComplianceMonitoringAgencyTypeCode
  - **ICS_CONTACT** Contact
    - **ICS_TELEPH** Telephone
  - **ICS_CSO_INSP** CSOInspection
  - **ICS_INSP_CMNT_TXT** InspectionCommentText
  - **ICS_INSP_GOV_CONTACT** InspectionGovernmentContact
  - **ICS_NAT_PRIO** NationalPrioritiesCode
  - **ICS_PRETR_INSP** PretreatmentInspection
  - **ICS_PROG** ProgramCode
  - **ICS_PROG_DEFCY_TYPE** ProgramDeficiencyTypeCode
  - **ICS_SSO_INSP** SSOInspection
    - **ICS_IMPACT_SSO_EVT** ImpactSSOEvent
    - **ICS_SSO_STPS** SSOSteps
    - **ICS_SSO_SYSTM_COMP** SSOSystemComponent
  - **ICS_SW_CNST_INSP** StormWaterConstructionInspection
    - **ICS_SW_CNST_INDST_INSP** StormWaterConstructionIndustrialInspection
    - **ICS_SW_UNPRMT_CNST_INSP** StormWaterUnpermittedConstructionInspection
      - **ICS_PROJ_TYPE** ProjectType
  - **ICS_SW_CNST_NON_CNST_INSP** StormWaterConstructionNonConstructionInspections
    - **ICS_SW_CNST_INDST_INSP** StormWaterConstructionIndustrialInspection
    - **ICS_SW_UNPRMT_CNST_INSP** StormWaterUnpermittedConstructionInspection
      - **ICS_PROJ_TYPE** ProjectType
  - **ICS_SW_MS_4_INSP** StormWaterMS4Inspection
    - **ICS_PROJ_SRCS_FUND** ProjectedSourcesFundingCode
  - **ICS_SW_NON_CNST_INSP** StormWaterNonConstructionInspection
    - **ICS_SW_CNST_INDST_INSP** StormWaterConstructionIndustrialInspection
    - **ICS_SW_UNPRMT_CNST_INSP** StormWaterUnpermittedConstructionInspection
      - **ICS_PROJ_TYPE** ProjectType

EffluentTradePartnerSubmission

- **ICS_EFFLU_TRADE_PRTNER** EffluentTradePartnerData
  - **ICS_EFFLU_TRADE_PRTNER_ADDR** EffluentTradePartnerAddress
    - **ICS_TELEPH** Telephone

FormalEnforcementActionSubmission

- **ICS_FRML_ENFRC_ACTN** FormalEnforcementActionData
  - **ICS_ENFRC_ACTN_GOV_CONTACT** EnforcementActionGovernmentContact
  - **ICS_ENFRC_ACTN_TYPE** EnforcementActionTypeCode
  - **ICS_ENFRC_AGNCY** EnforcementAgency
  - **ICS_FINAL_ORDER** FinalOrder
    - **ICS_FINAL_ORDER_PRMT_IDENT** FinalOrderPermitIdentifier
    - **ICS_SEP** SupplementalEnvironmentalProject
  - **ICS_PRMT_IDENT** PermitIdentifier
  - **ICS_PROGS_VIOL** ProgramsViolatedCode

InformalEnforcementActionSubmission

- **ICS_INFRML_ENFRC_ACTN** InformalEnforcementActionData
  - **ICS_ENFRC_ACTN_GOV_CONTACT** EnforcementActionGovernmentContact
  - **ICS_ENFRC_AGNCY** EnforcementAgency
  - **ICS_PRMT_IDENT** PermitIdentifier
  - **ICS_PROGS_VIOL** ProgramsViolatedCode

ComplianceScheduleSubmission

- **ICS_CMPL_SCHD** ComplianceScheduleData
  - **ICS_CMPL_SCHD_EVT** ComplianceScheduleEvent

EnforcementActionMilestoneSubmission

- **ICS_ENFRC_ACTN_MILESTONE** EnforcementActionMilestoneData

DMRViolationSubmission

- **ICS_DMR_VIOL** DMRViolationData

SingleEventViolationSubmission

- **ICS_SNGL_EVT_VIOL** SingleEventViolationData

CopyMGPLimitSetSubmission

- **ICS_COPY_MGP_LMT_SET** CopyMGPLimitSetData
  - **ICS_GEO_COORD** GeographicCoordinates
  - **ICS_LMT_SET_SCHD** LimitSetSchedule
  - **ICS_LMT_SET_STAT** LimitSetStatus

SWIndustrialAnnualReportSubmission

- **ICS_SW_INDST_ANNUL_REP** SWIndustrialAnnualReportData

BiosolidsAnnualProgramReportSubmission

- **ICS_BS_ANNUL_PROG_REP** BiosolidsAnnualProgramReportData
  - **ICS_ANLYTCL_METHOD** AnalyticalMethodData
  - **ICS_BS_FAC_TRTMNT** BiosolidsFacilityTreatment
  - **ICS_BS_FAC_TYPE** BiosolidsFacilityType
  - **ICS_BS_MGMT_PRACTICE** BiosolidsManagementPracticeData
    - **ICS_BS_INCINERATION** BiosolidsIncinerationData
      - **ICS_BS_INCIN_EMISSIONS_CONTROL_TYPE** BiosolidsIncineratorEmissionsControlType
      - **ICS_BS_SEWAGE_SLDG_PARAM** BiosolidsSewageSludgeParameter
    - **ICS_BS_OFF_SITE_HNDLR_APPLIER_PREPR** BiosolidsOffSiteHandlerApplierPreparer
      - **ICS_ADDR** Address
        - **ICS_TELEPH** Telephone
      - **ICS_CONTACT** Contact
        - **ICS_TELEPH** Telephone
    - **ICS_BS_SEWAGE_SLDG_PARAM** BiosolidsSewageSludgeParameter
    - **ICS_CMPL_MON_EVT** ComplianceMonitoringEventData
      - **ICS_BS_SEWAGE_SLDG_PARAM** BiosolidsSewageSludgeParameter
    - **ICS_PATHOGEN_REDUCTION_TYPE** PathogenReductionTypeCode
    - **ICS_VECTOR_A_REDUCTION_TYPE** VectorAttractionReductionTypeCode

ComplianceMonitoringLinkageSubmission

- **ICS_CMPL_MON_LNK** ComplianceMonitoringLinkageData
  - **ICS_LNK_CMPL_MON** LinkageComplianceMonitoring
  - **ICS_LNK_ENFRC_ACTN** LinkageEnforcementAction
  - **ICS_LNK_SNGL_EVT** LinkageSingleEvent

EnforcementActionViolationLinkageSubmission

- **ICS_ENFRC_ACTN_VIOL_LNK** EnforcementActionViolationLinkageData
  - **ICS_CMPL_SCHD_VIOL** ComplianceScheduleViolation
  - **ICS_DSCH_MON_REP_PARAM_VIOL** DischargeMonitoringReportParameterViolation
  - **ICS_DSCH_MON_REP_VIOL** DischargeMonitoringReportViolation
  - **ICS_PRMT_SCHD_VIOL** PermitScheduleViolation
  - **ICS_SNGL_EVTS_VIOL** SingleEventsViolation

ScheduleEventViolationSubmission

- **ICS_SCHD_EVT_VIOL** ScheduleEventViolationData
  - **ICS_CMPL_SCHD_EVT_VIOL_ELEM** ComplianceScheduleEventViolationKeyElements
  - **ICS_PRMT_SCHD_EVT_VIOL_ELEM** PermitScheduleEventViolationKeyElements

FinalOrderViolationLinkageSubmission

- **ICS_FINAL_ORDER_VIOL_LNK** FinalOrderViolationLinkageData
  - **ICS_CMPL_SCHD_VIOL** ComplianceScheduleViolation
  - **ICS_DSCH_MON_REP_PARAM_VIOL** DischargeMonitoringReportParameterViolation
  - **ICS_DSCH_MON_REP_VIOL** DischargeMonitoringReportViolation
  - **ICS_PRMT_SCHD_VIOL** PermitScheduleViolation
  - **ICS_SNGL_EVTS_VIOL** SingleEventsViolation

CSOLongTermControlPlanSubmission

- **ICS_CSO_LONG_TERM_CONTROL_PLAN** CSOLongTermControlPlanData
  - **ICS_CSO_CONTROL_MEAS_DETAIL** CSOControlMeasureDetail
  - **ICS_LTCP_ENFORCEABLE_MECH_DETAIL** LTCPEnforceableMechanismDetail
  - **ICS_LTCP_MOST_RECENT_REVISION_DETAIL** LTCPMostRecentRevisionDetail
  - **ICS_LTCP_SUMM** LTCPSummary

CWA316bProgramReportSubmission

- **ICS_CWA_316B_PROG_REP** CWA316bProgramReportData
  - **ICS_CWA_316B_TAKE_INFO** CWA316bTakeInformation

CollectionSystemPermitSubmission

- **ICS_COLL_SYSTM_PRMT** CollectionSystemPermitData

CopyMGPMS4RequirementSubmission

- **ICS_COPY_MGPMS_4_REQ** CopyMGPMS4RequirementData
  - **ICS_GNRL_PRMT_COVERAGE_MS_4_REQ** GeneralPermitCoverageMS4Requirement
    - **ICS_MS_4_REGULATED_ENTITY_IDENT** MS4RegulatedEntityIdentifier
  - **ICS_MS_4_ACTY_IDENT** MS4ActivityIdentifier

NPDESVariancePermitSubmission

- **ICS_NPDES_VARIANCE_PRMT** NPDESVariancePermitData

POTWTreatmentTechnologyPermitSubmission

- **ICS_POTW_TRTMNT_TECHNOLOGY_PRMT** POTWTreatmentTechnologyPermitData

PretreatmentProgramReportSubmission

- **ICS_PRETR_PROG_REP** PretreatmentProgramReportData
  - **ICS_CONTROL_AUTH_PROG_INFO** ControlAuthorityProgramInformation
    - **ICS_LOC_LMTS_PARAMETERS** LocalLimitsParameters
  - **ICS_INDST_USR_INVENTORY** IndustrialUserInventory
    - **ICS_INDST_USR_INFO** IndustrialUserInformation
      - **ICS_IU_ENFRC_ACTN_INFO** IUEnforcementActionInformation
        - **ICS_IU_ENF_ACTN_TYPES** IUEnfActionTypes
      - **ICS_IU_VIOL_INFO** IUViolationInformation
        - **ICS_SNC_LISTING_MONTHS** SNCListingMonths
        - **ICS_SNC_PRETR_STND_LMTS_PARAMETERS** SNCPretrStndLimitsParameters

SWMS4AnnualProgramReportSubmission

- **ICS_SWMS_4_ANNUL_PROG_REP** SWMS4AnnualProgramReportData
  - **ICS_MS_4_ILLICIT_DETECT_REGULATED_ENTITY_INFO_PROG_REP** MS4IllicitDetectionRegulatedEntityInformationProgramReport
  - **ICS_MS_4_PROG_REP_ANALYSIS** MS4ProgramReportAnalysis
    - **ICS_MS_4_REGULATED_ENTITY_IDENT** MS4RegulatedEntityIdentifier
  - **ICS_MS_4_PROG_REP_REQS_REGULATED_ENTITY** MS4ProgramReportRequirementsRegulatedEntity
    - **ICS_MS_4_REGULATED_ENTITY_IDENT** MS4RegulatedEntityIdentifier
  - **ICS_MS_4_REGULATED_ENTITY_ENFRC** MS4RegulatedEntityEnforcement
    - **ICS_MS_4_REGULATED_ENTITY_ENFRC_ACTIONS** MS4RegulatedEntityEnforcementActions
  - **ICS_MS_4_SWMP_CHANGES** MS4SWMPChanges

SWMS4PermitSubmission

- **ICS_SWMS_4_PRMT** SWMS4PermitData
  - **ICS_MS_4_CNST_SW_REQS** MS4ConstructionStormwaterRequirements
    - **ICS_MS_4_CNST_SW_PROCEDURES** MS4ConstructionStormwaterProcedures
      - **ICS_MS_4_REGULATED_ENTITY_IDENT** MS4RegulatedEntityIdentifier
    - **ICS_MS_4_CNST_SW_REGULATED_ENTITY_INFO** MS4ConstructionStormwaterRegulatedEntityInformation
  - **ICS_MS_4_ILLICIT_DETECT_REQS** MS4IllicitDetectionRequirements
    - **ICS_MS_4_ILLICIT_DETECT_PROCEDURES** MS4IllicitDetectionProcedures
      - **ICS_MS_4_REGULATED_ENTITY_IDENT** MS4RegulatedEntityIdentifier
    - **ICS_MS_4_ILLICIT_DETECT_REGULATED_ENTITY_INFO** MS4IllicitDetectionRegulatedEntityInformation
  - **ICS_MS_4_INDST_SW_REQS** MS4IndustrialStormwaterRequirements
    - **ICS_MS_4_INDST_SW_PROCEDURES** MS4IndustrialStormwaterProcedures
      - **ICS_MS_4_REGULATED_ENTITY_IDENT** MS4RegulatedEntityIdentifier
    - **ICS_MS_4_INDST_SW_REGULATED_ENTITY_INFO** MS4IndustrialStormwaterRegulatedEntityInformation
  - **ICS_MS_4_OTHR_APPL_REQS** MS4OtherApplicableRequirements
    - **ICS_MS_4_REGULATED_ENTITY_IDENT** MS4RegulatedEntityIdentifier
  - **ICS_MS_4_PBLC_EDUCATION_REQS** MS4PublicEducationRequirements
    - **ICS_MS_4_REGULATED_ENTITY_IDENT** MS4RegulatedEntityIdentifier
  - **ICS_MS_4_PBLC_INVOLVEMENT_REQS** MS4PublicInvolvementRequirements
    - **ICS_MS_4_REGULATED_ENTITY_IDENT** MS4RegulatedEntityIdentifier
  - **ICS_MS_4_POLLUTION_PREVENTION_REQS** MS4PollutionPreventionRequirements
    - **ICS_MS_4_REGULATED_ENTITY_IDENT** MS4RegulatedEntityIdentifier
  - **ICS_MS_4_POST_CNST_SW_REQS** MS4PostConstructionStormwaterRequirements
    - **ICS_MS_4_POST_CNST_SW_PROCEDURES** MS4PostConstructionStormwaterProcedures
      - **ICS_MS_4_REGULATED_ENTITY_IDENT** MS4RegulatedEntityIdentifier
    - **ICS_MS_4_POST_CNST_SW_REGULATED_ENTITY_INFO** MS4PostConstructionStormwaterRegulatedEntityInformation
  - **ICS_MS_4_REGULATED_ENTITY** MS4RegulatedEntity
  - **ICS_MS_4_REGULATED_ENTITY_AREA** MS4RegulatedEntityArea
    - **ICS_MS_4_REGULATED_ENTITY_AREA_COORD** MS4RegulatedEntityAreaCoordinates

SewerOverflowBypassEventReportSubmission

- **ICS_SEWER_OVRFLW_BYPASS_EVT_REP** SewerOverflowBypassEventReportData
  - **ICS_SEWER_OVRFLW_BYPASS_REP_EVT** SewerOverflowBypassReportEvent
    - **ICS_BYPASS_TRTMNT_PLANT_EQUIPMENT** BypassTreatmentPlantEquipment
    - **ICS_SEWER_OVRFLW_BYPASS_CAUSE** SewerOverflowBypassCause
    - **ICS_SEWER_OVRFLW_BYPASS_CORR_ACTN** SewerOverflowBypassCorrectiveAction
    - **ICS_SEWER_OVRFLW_BYPASS_IMPACT** SewerOverflowBypassImpact
    - **ICS_SEWER_OVRFLW_BYPASS_RCVG_WTR** SewerOverflowBypassReceivingWater
    - **ICS_SEWER_OVRFLW_BYPASS_TYPE** SewerOverflowBypassTypeCode
    - **ICS_SEWER_OVRFLW_TRTMNT** SewerOverflowTreatmentCode

CAFOAnnualProgramReportSubmission