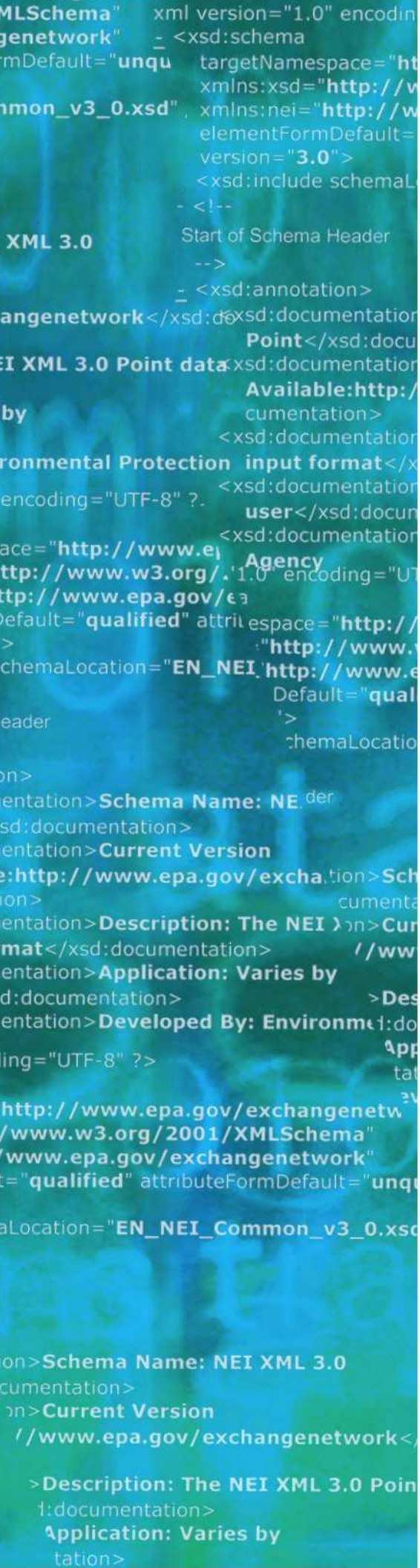


OpenNode2

ICIS-Air v5.0
Data Exchange Implementation Guide
(JAVA)

Revision Date: 10/05/2015

Prepared By:

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

Revision History

Date	Author	Changes
9/11/2014	Windsor	Initial version
9/26/2014	Windsor	Updated steps in Setting up the Staging Database section
10/15/2014	Windsor	Removed references to seeding of staging tables from ICIS-Air.
11/5/2014	Windsor	Updated as follows: - Include instructions on creating data sources prior to data service configuration. - Include references to CDX Node 2.1 Endpoints. - Clarify purpose of hash initialization database script.
12/5/2014	Windsor	Updated as follows: - Remove references to CDX Node 1.1 Endpoints. This plugin should only be configured to direct submissions to CDX Node 2.1 Endpoints. - Remove unnecessary step from Data Service configuration instructions.
2/2/2015	Windsor	Updated as follows: - Include instructions regarding running staging table patch scripts when updating an existing implementation.
8/13/2015	Windsor	Updated as follows: - Revise staging database installation instructions to reflect increment of XML schema version to 5.4.
10/05/2015	Windsor	Updated as follows: - Revise Oracle staging database installation instructions.

Table of Contents

DATA EXCHANGE OVERVIEW 1
STAGING TABLE ARCHITECTURE 3
Database 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-AIR PLUGIN 14
Create ICIS-Air Data Exchange *14*
Install ICIS-Air Full Batch Plugin *15*
Add CDX to the OpenNode2 Partner List *15*
Create Staging Data Source *16*
Create ICIS-Air Data Services *17*
Create Data Exchange Schedules *20*
ADDITIONAL ACTIVITIES 24
Contact CDX to Establish Data Exchange Settings *24*
Monitor Flow Activity *24*
APPENDIX A - ICIS-AIR XML PAYLOAD STAGING TABLES 25
APPENDIX B – ICIS-AIR XML SCHEMA/STAGING TABLE MAPPING DIAGRAMS 26

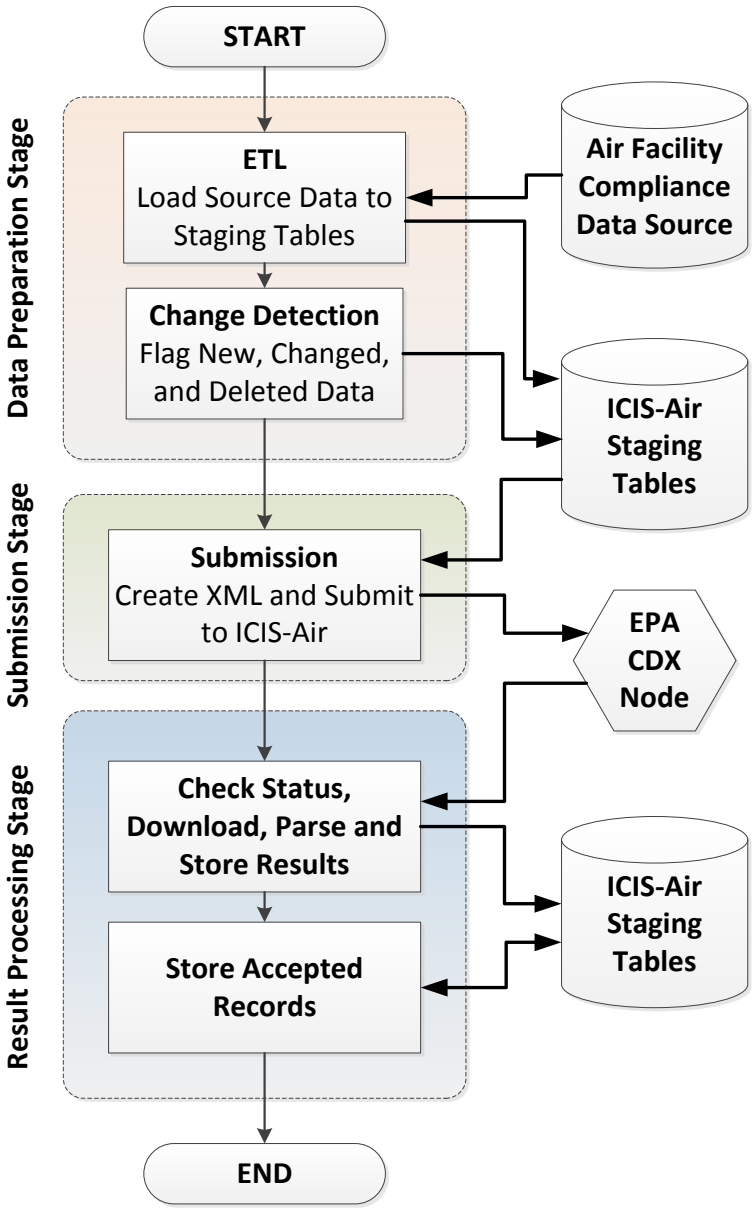
THIS PAGE INTENTIONALLY LEFT BLANK

Data Exchange Overview

The purpose of this document is to provide detailed instructions for the installation and configuration of the ICIS-Air 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-Air Flow Configuration Document* (FCD) and *ICIS XML Schema User's Guide* available on the Exchange Network website, as well as the *ICIS-Air v5.0 Plugin Design Specification* document available from the OpenNode2 SourceForge Code repository.

The ICIS-Air plugin operation includes three discrete processing stages as illustrated in the following figure:



The **Data Preparation Stage** first refreshes the staging database with the latest data from the agency's air facility compliance information system, and then second determines what data needs to be sent to ICIS-Air. Refreshing the staging tables from the source system involves the use of a custom extract, transformation, and load (ETL) procedure or procedures. Special database procedures provided with the plugin package will then automatically determine what data from the staging database needs to be sent to ICIS-Air, eliminating the need for the custom ETL logic to perform this task. This stage is executed by a scheduled task within OpenNode2.

The **Submission Stage** involves retrieving new, changed, or deleted data from the staging tables, converting the data into the ICIS-Air XML schema format, and then transmitting the XML to EPA's Central Data Exchange (CDX). This stage is executed by the same scheduled task within OpenNode2 which manages the previous Data Preparation Stage.

The **Result Processing Stage** involves retrieving ICIS-Air processing reports in XML format from EPA's CDX system and parsing 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-Air and used to determine the data that needs to be sent in subsequent submissions. This stage is executed by a second scheduled task within OpenNode2.

Each stage is executed in isolation and is only responsible for one aspect of the data flow. 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.

Note on ICIS XML Schema

The ICIS-Air data exchange will utilize the ICIS XML schema version 5.4. EPA CDX is currently supporting this schema version for submissions to the test version of ICIS-Air and will support this version for submissions to the production system beginning on August 14th, 2015.

The ICIS XML schema supports submissions to both ICIS-Air and ICIS-Air. The ICIS-Air OpenNode2 plugin will utilize only a subset of the schema elements and only the elements used by ICIS-Air are included in the supporting staging table architecture.

Staging Table Architecture

Like most data exchanges implemented on the OpenNode2 platform, the ICIS-Air data 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.

Database Schemas

The ICIS-Air plugin requires that two separate database schemas[1] be created; referred to in this document as **Staging-Local** and **Staging-ICIS**. Staging-Local (named ICA_FLOW_LOCAL) stores data that the agency wishes to flow to ICIS. Staging-ICIS (named ICA_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 (ICA_FLOW_LOCAL)

The Staging-Local schema is used to store data that the agency wishes to flow to ICIS-Air. It is populated by an agency-specific ETL routine on a regular interval.

The Staging-Local tables closely follow the structure of the ICIS-Air XML schema, and are used to store data that the agency wishes to flow to ICIS-Air. These tables are populated with data provided by a custom developed ETL routine that reads data from the agency's air facility compliance information management system, translates the data into ICIS-Air structures, formats the data appropriately (including the 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 (ICA_FLOW_ICIS)

The tables in the Staging-ICIS schema are used to store all the records that have been accepted by EPA ICIS-Air. The table structures are identical to those of the Staging-Local schema.

When an Accepted Transactions XML Report is received from EPA CDX and processed by OpenNode2, the accepted data records 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 the Staging-Local schema.

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

Staging Tables

Each schema contains the following tables:

- 56 staging tables to store ICIS-Air data
- 1 table to track accepted and rejected transactions (ICA_SUBM_RSLTS)
- 1 table to track plugin workflow execution state (ICA_SUBM_TRACK)
- 1 table to track the history of plugin submissions (ICA_SUBM_HIST)

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

ICIS-Air Staging Tables

All but three of the tables in the staging table schemas are used to store ICIS-Air data. The root table is ICA_PAYLOAD. Below this table are 13 child tables, each relating to a different submission type supported by ICIS-Air.

Results Tracking Table (ICA_SUBM_RSLTS)

The ICA_SUBM_RSLTS table is used to store the business keys for accepted and rejected transactions along with warning and error messages returned by ICIS-Air 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-Air.

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.

Submission Tracking Table (ICA_SUBM_TRACK)

The ICA_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-Air 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.

Transaction History Table (ICA_SUBM_HIST)

This table stores the counts of Change/Replace, Mass Delete, Accepted, Warning, and Rejected transactions by transaction ID and submission type. Each time a submission processing report is downloaded and processed by the plugin, the ICA_PROCESS_ACCEPTED_TRANS procedure calculates these counts and adds them to this table. This table can be used to develop a chart or graph of data and error volume over time.

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 ETL routine will need to be built by the implementing agency to load ICIS-Air data from the agency's air facility compliance database to the OpenNode2 staging database. The OpenNode2 ICIS-Air data exchange supports three different approaches to staging and submitting data. Each agency will need to decide the approach that best supports its own needs. These three approaches are:

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

Approach #1: Full Data Synchronization

Using this approach, the ETL process updates the ICIS-Air data in the Staging-Local schema to represent a reflection of *all* the current air facility compliance 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 the ICA_PAYLOAD table 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 transactions whenever data in the Staging-Local schema is missing, but data for the same business key has already been successfully sent to ICIS-Air in the past, as determined by the presence of the data in the Staging-ICIS schema.

One of the major advantages to this approach is that it eliminates the need to track what new, changed, or deleted data needs to be sent to ICIS-Air in the source air facility compliance system itself. This has historically been one of the bigger challenges for agencies implementing data flows which synchronize with EPA systems.

Approach #2: Incremental Data with Automated Change Detection

Using this alternative, the ETL process updates the ICIS-Air data in the Staging-Local schema with a specific set of data to be sent. This approach would be used where only the data added or changed in a given timeframe is to be staged.

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 ICA_SUBM_TRACK table. It also requires that the agency's air facility compliance 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.

It should be noted that this approach will not support automatic deletion of records from ICIS-Air and 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-Air that was successfully sent to ICIS-Air 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

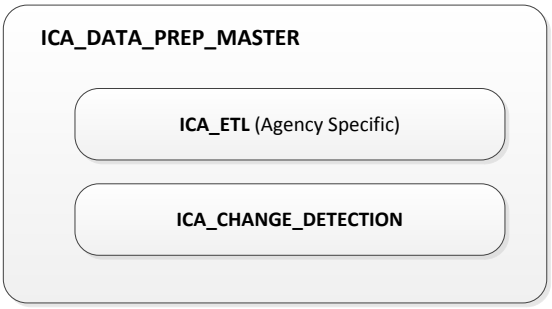
This approach is similar to #2 above, however the agency chooses to set the transaction types using their own developed ETL logic instead of leveraging the built-in change detection procedure. This approach requires that the state has developed its own mechanism to track which data needs to be added,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

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

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

Data Preparation Stored Procedures

The ICIS-Air plugin will come 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 will be three stored procedures that together comprise the data preparation stage. The diagram below illustrates the relationship between these procedures.



ICA_DATA_PREP_MASTER is the parent stored procedure that encapsulates the agency-specific ICA_ETL procedure and the included ICA_CHANGE_DETECTION procedure. The ICA_DATA_PREP_MASTER procedure executes the following workflow:

1. Check the ICA_SUBM_TRACK table to ensure that no existing workflows have a status of 'Pending'. If any exist, exit without error, returning a NULL ICA_SUBM_TRACK_ID in the procedure's output parameter.
2. Begin a new transaction.
3. Insert a new record in the ICA_SUBM_TRACK table to begin a new workflow with a status of 'Pending'.
4. Call the agency-specific ICA-ETL procedure.
5. Update the workflow record with the completion date/time of the ETL process.
6. Call the ICA_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 ICA_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 ICA_DATA_PREP_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-Air 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-Air v5.0 Plugin Design Specification* document 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 ICA_PAYLOAD. A record should exist in this table for every submission or payload type to be submitted to EPA. The only time when a record should be added or removed from this table is if the agency is adding or removing payload types from their data flow.
- The ICA_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 which are present in Staging-ICIS.
- "Root" staging tables exist for all 13 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 (air facility, informal enforcement action, 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 for each payload type.
- Each root table contains a column labeled SRC_SYSTM_IDENT. This field is not used by the plugin but provides the agency with a convenient place to store the primary key value for the relevant record in the agency's source air facility compliance database that maps to the staging record. This can be convenient for ETL design and data auditing purposes.

ETL Design Considerations

- If a submission 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 ICA_CONTACT and ICA_TELEPH. Only one foreign key should be populated at a time in such tables. 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 downloading and parsing of processing results, or failure of the workflow at some point in its execution.

- Overall workflow is managed in the ICA_SUBM_TRACK table. The ICA_DATA_PREP_MASTER stored 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 WORKFLOW_STAT_MESSAGE column of the ICA_SUBM_TRACK table and in the OpenNode2 Activity Log.
- 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 ICA_SUBM_RSLTS table stores accepted and rejected transaction data returned from ICIS-Air. This table contains columns for each of the business key fields used throughout the ICIS-Air 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 ICA_SUBM_RSLTS table.
- Once the plugin finishes saving the data from the processing reports, the plugin then executes a post-processing stored procedure named ICA_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 *ICIS-Air Plugin Design Specification* document. 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 removed, 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

The following steps will establish the database environment for the ICIS-Air data flow. The process is essentially the same for Oracle and SQL Server with minor technical variances. The database scripts referenced in the instructions below are included in the OpenNode2 software installation zip file in the **Sql\ICIS-AIR** directory.

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

There are two deployment scenarios for the ICIS-Air staging tables:

a) New installation
b) Updating an existing installation

The following steps should be followed for each of these scenarios.

New Installation

1. Create three schemas named ICA_FLOW_LOCAL, ICA_FLOW_LOCAL_USER, and ICA_FLOW_ICIS.
    a. ICA_FLOW_LOCAL will need CONNECT, RESOURCE, and CREATE VIEW privileges.
    b. ICA_FLOW_LOCAL_USER will need CONNECT, RESOURCE, and CREATE SYNONYM privileges.
    c. ICA_FLOW_ICIS will need CONNECT and RESOURCE privileges.
2. Run the *ICIS_AIR_5.4-ORA-DDL-GRANTS-SYS.sql* script, found in the **grants** folder, as the SYS user. This will provide access to an internal Oracle package DBMS_CRYPTO that is vital to the ICIS-Air data processing.
3. Run the *ICIS_AIR_5.4-ORA-DDL.sql* script, found in the **staging_table_ddl** folder, as the ICA_FLOW_LOCAL user. This will create the ICIS-Air staging tables in the ICA_FLOW_LOCAL database schema. Run this same script again as the ICA_FLOW_ICIS user to create those identical staging tables in ICA_FLOW_ICIS schema.

    Note: The **staging_table_ddl** folder also contains a "drop all tables" script. This is useful if rebuilding an existing ICIS-Air staging database. Note that the script will not drop tables that have been renamed between updated releases of the ICIS-Air staging tables. If this is the case, manually delete any tables that remain after the script is run.
    a. Optionally, drop the ICA_SUBM_RSLTS and ICA_SUBM_TRACK tables from the ICA_FLOW_ICIS schema since they are not used.
4. Run the *ICIS_AIR_5.4-ORA-DDL-GRANTS-ICA_FLOW_ICIS.sql* script, found in the **grants** folder, as user ICA_FLOW_ICIS. This will grant ICA_FLOW_LOCAL the access needed for change comparison processing.
5. Run the following scripts found in the **views** folder as the ICA_FLOW_LOCAL user:
    a. ICIS_AIR_5.4-ORA-Change Detection Views.sql

   b. *ICIS_AIR_5.4-ORA-ICA_V_MODULE_COUNT.sql*
   c. *ICIS_AIR_5.4-ORA-VW_ICA_CASE_FILE_LNK.sql*
   d. *ICIS_AIR_5.4-ORA-VW_ICA_CMPL_MON_LNK.sql*
   e. *ICIS_AIR_5.4-ORA-VW_ICA_DA_ENFRC_ACTN_LNK.sql*
   f. *ICIS_AIR_5.4-ORA-VW_ICA_FAC_GEO_COORD.sql*
   g. *ICIS_AIR_5.4-ORA-VW_ICA_POLUTS_CLASS.sql*

6. Run the *ICIS_AIR_5.4-ORA-MD5_HASH.sql* script, found in the **functions** folder as the ICA_FLOW_LOCAL user to create a user defined function used by the change comparison processing.
7. Run the following scripts, found in the **procedures** folder, as the ICA_FLOW_LOCAL user in the order below. These will create the procedures that execute the ETL, perform the change detection process, and process accepted transactions. Run the ICA_DATA_PREP_MASTER script last since it depends on two others.

   a. *ICIS_AIR_5.4-ORA-ICA_CHANGE_DETECTION.sql*
   b. *ICIS_AIR_5.4-ORA-ICA_ETL.sql*
   c. *ICIS_AIR_5.4-ORA-ICA_PROCESS_ACCEPTED_TRANS.sql*
   d. *ICIS_AIR_5.4-ORA-ICA_DATA_PREP_MASTER.sql*

8. Run the script *ICIS_AIR_5.4-ORA-DDL-GRANTS-ICA_FLOW_LOCAL.sql*, found in the **grants** folder, as the user ICA_FLOW_LOCAL.
9. In the synonyms folder, run the script *ICIS_AIR_5.4-ORA-DDL-SYNONYMS-ICA_FLOW_LOCAL_USER.sql* as the ICA_FLOW_LOCAL_USER. This will alias database objects from ICA_FLOW_LOCAL accessed by ICA_FLOW_LOCAL_USER.
10. The ICA_ETL procedure is only a shell. Update the ICA_ETL procedure to include the custom data loading procedures from your agency's air facility compliance database. Use the following resources to help you design and build the ETL routines:

    a. The staging table block diagrams in **Appendix B** of this document,

    b. The *ICIS-Air v5.0 Plugin Design Specification* document, and

    c. The EPA-provided ICIS-Air data flow documentation available on the Exchange Network web site (www.exchangenetwork.net).

11. Open table ICA_PAYLOAD in the ICA_FLOW_LOCAL schema/database. All 13 payload types accepted by ICIS-Air are inserted by default. You may remove the rows for the payload types you do not plan on submitting although it is not necessary.
12. An optional utility script is also included in the deployment package called *ICIS_AIR_5.4-ORA-ICA_ICIS_HASH_INITIALIZATION.sql*. This script will create a new stored procedure called *ICA_ICIS_HASH_INITIALIZATION* which can be used to establish the necessary hash values on data records in the ICA_FLOW_ICIS schema if the agency chooses to pre-load the ICA_FLOW_ICIS tables with data migrated to ICIS-Air from the legacy AFS system and provided to the agency by EPA. Establishing the hash values will allow the change detection and error processing procedures to function correctly. Take the following steps to execute this utility procedure if needed:

    a. Run the *ICIS_AIR_5.4-ORA-MD5_HASH.sql* script, found in the **functions** folder as the ICA_FLOW_ICIS user to create a user defined function used by the initialization utility.

b. Execute the *ICIS_AIR_5.4-ORA-ICA_ICIS_HASH_INITIALIZATION.sql* database script as the ICA_FLOW_ICIS schema owner to create the *ICA_ICIS_HASH_INITIALIZATION* stored procedure.
c. Execute the stored procedure *ICA_ICIS_HASH_INITIALIZATION* to initialize the key and data hash columns.

Updating Existing Installation

1. A patch script is included in the deployment package which may be used to update an existing staging table installation to bring the staging databases in line with the latest available structures and objects that would be deployed with a brand new implementation as described above. This script will be updated with future versions of the plugin deployment package to include any necessary updates. Take the following steps to execute this utility procedure if needed:

    a. Execute the *ICIS_AIR_5.4-ORA-DDL-Patch.sql* stored procedure as the SYSTEM user to apply the included updates.

SQL Server Database Deployment

There are two deployment scenarios for the ICIS-Air staging tables:

a) New installation
b) Updating an existing installation

The following steps should be followed for each of these scenarios.

New Installation

1. Create two databases named ICA_FLOW_LOCAL and ICA_FLOW_ICIS.
2. Run the *ICIS_AIR_5.4-SQL-DDL.sql* script in the **ddl** folder to create the staging tables in the ICA_FLOW_LOCAL schema/database. Run the script again to create the same objects in ICA_FLOW_ICIS database.
3. **Note**: The **staging_table_ddl** folder also contains a "drop all tables" script. This is useful if rebuilding an existing ICIS-Air staging database. Note that the script will not drop tables that have been renamed between updated releases of the ICIS-Air staging tables. If this is the case, manually delete any tables that remain after the script is run.

    a. Optionally, drop the ICA_SUBM_RSLTS and ICA_SUBM_TRACK tables from the ICA_FLOW_ICIS database since they are not used.

4. Run the *ICIS_AIR_5.4-SQL-Change Detection Views.sql* script in the **views** folder on the ICA_FLOW_LOCAL schema/database.
5. Run the *ICIS_AIR_5.4-SQL-ICA_V_MODULE_COUNT.sql* script in the **views** folder on the ICA_FLOW_LOCAL schema/database.
6. Run the *ICIS_AIR_5.4-SQL-MD5_HASH.sql* script in the **functions** folder to create a function used by the bundled procedures.
7. Run the following scripts in the **procedures** folder on the ICA_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 ICA_DATA_PREP_MASTER script last since it depends on two others.

    a. ICIS_AIR_5.4-SQL-ICA_CHANGE_DETECTION.sql

   b. *ICIS_AIR_5.4-SQL-ICA_ETL.sql*
      i. *Ignore warning related to missing objects. These warnings are related to the specific ICIS Air extract procedures that will need to be develop to extract, transform and then populate the ICIS Air staging schema.*
   c. *ICIS_AIR_5.4-SQL-ICA_PROCESS_ACCEPTED_TRANS.sql*
   d. *ICIS_AIR_5.4-SQL-ICA_DATA_PREP_MASTER.sql*

8. Create a SQL login named ICA_FLOW_LOCAL_USER and map the to the newly created ICIS Air databases. This login will be used by OpenNode2 to interact with the staging database and execute stored procedures.
   a. Create a new database role in the ICA_FLOW_LOCAL database named *db_execute.*
      i. Command: CREATE ROLE db_execute AUTHORIZATION dbo
   b. Grant EXECUTE to the database role db_execute.
      ii. Command: GRANT EXECUTE TO db_execute
   c. Map the ICA_FLOW_LOCAL_USER user to the *db_datareader*, *db_datawriter*, and the newly created *db_execute* roles in the ICA_FLOW_LOCAL database.
   d. Map the ICA_FLOW_LOCAL_USER user to the *db_datareader*, and *db_datawriter*, in the ICA_FLOW_ICIS database.
   e. **Note**: You will most likely also need to map the ICA_FLOW_LOCAL_USER login to your ICIS Air source database. This will allow the ICIS Air data extract to run properly. If at all possible only give ICA_FLOW_LOCAL_USER *db_datareader* on the source database to eliminate the possibility of data changes getting pushed into your production environment.
9. The ICA_ETL procedure is only a shell. Update the ICA_ETL procedure to include the custom data loading procedures from your agency's air facility compliance database. Use the following resources to help you design and build the ETL routines:
   a. The staging table block diagrams in **Appendix B** of this document,
   b. The *ICIS-Air Plugin Design Specification* document, and
   c. The EPA-provided ICIS-Air data flow documentation available on the Exchange Network web site (www.exchangenetwork.net).
10. Open table ICA_PAYLOAD in the ICA_FLOW_LOCAL schema/database. All 13 payload types accepted by ICIS-Air are inserted by default. You may remove the rows for the payload types you do not plan on submitting although it is not necessary.
11. An optional utility script is also included in the deployment package called *ICIS_AIR_5.4-SQL-ICA_ICIS_HASH_INITIALIZATION.sql*. This script will create a new stored procedure called *ICA_ICIS_HASH_INITIALIZATION* which can be used to establish the necessary hash values on data records in the ICA_FLOW_ICIS schema if the agency chooses to pre-load the ICA_FLOW_ICIS tables with data migrated to ICIS-Air from the legacy AFS system and provided to the agency by EPA. Establishing the hash values will allow the change detection and error processing procedures to function correctly. Take the following steps to execute this utility procedure if needed:
   a. Execute the *ICIS_AIR_5.4-ORA-ICA_ICIS_HASH_INITIALIZATION.sql* stored procedure in the ICA_FLOW_ICIS schema to create the *ICA_ICIS_HASH_INITIALIZATION* stored procedure.

b. Execute the stored procedure *ICA_ICIS_HASH_INITIALIZATION* to initialize the key and data hash columns.

Updating Existing Installation

1. A patch script is included in the deployment package which may be used to update an existing staging table installation to bring the staging databases in line with the latest available structures and objects that would be deployed with a brand new implementation as described above. This script will be updated with future versions of the plugin deployment package to include any necessary updates. Take the following steps to execute this utility procedure if needed:

    a. Execute the *ICIS_AIR_5.4-SQL-DDL-Patch.sql* stored procedure as a user with **sysadmin** privileges to apply the included updates.

Install and Configure ICIS-Air Plugin

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

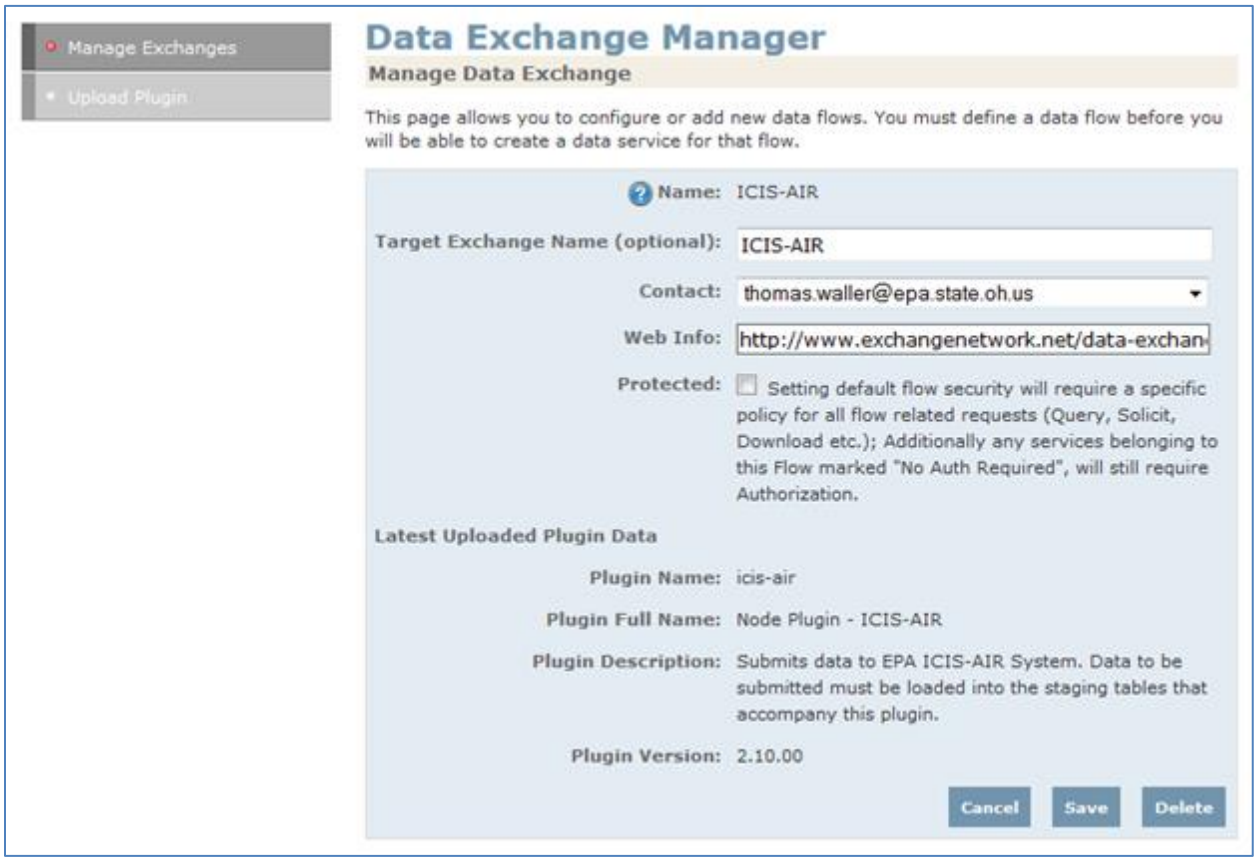
The following steps are required:

1) Create ICIS-Air Data Exchange
2) Install ICIS-Air Full Batch Plugin
3) Add CDX to the OpenNode2 Partner List
4) Create Staging Data Source
5) Create ICIS-Air Data Services
6) Create Data Exchange Schedules

Create ICIS-Air Data Exchange

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

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



3. Type *ICIS-AIR* in the **Name** field.
4. Type the appropriate data exchange name in the **Target Exchange Name** field.

a. When submitting data to test EPA CDX and ICIS-Air, this value should be set to *ICIS-Air*.

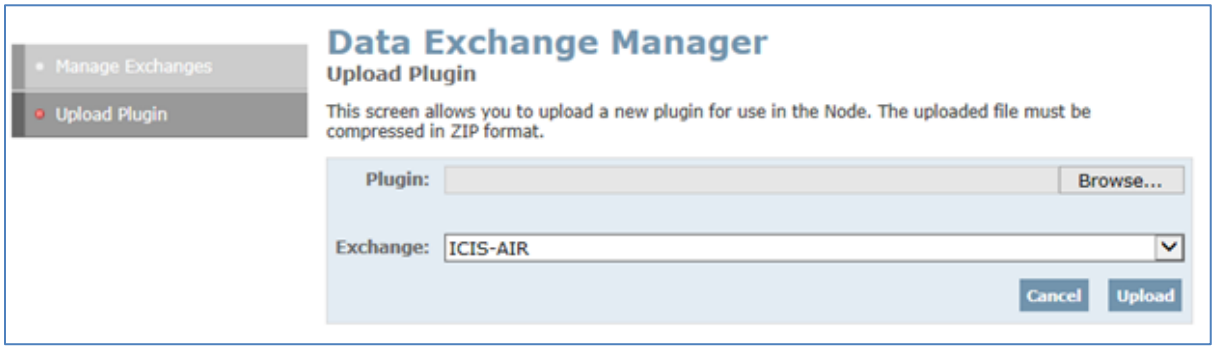
b. When submitting data to production EPA CDX and ICIS-Air, this value should be set to *ICIS-NPDES*.

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. Optionally, type any valid URL in the **Web Info** field. 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. Click **Save** to save the data exchange.

Install ICIS-Air Full Batch Plugin

Once the ICIS-Air data exchange has been created, the next step is to upload the ICIS-Air plugin into the OpenNode2 plugin repository.

1. From the **Exchange** tab, click the **Upload Plugin** button on the left navigation block.



2. Click the **Browse** button which is located to the right of the **Plugin** field.
1. Locate and select the compressed (zipped) file containing the code component for the ICIS-Air plugin obtained from the SourceForge OpenNode2 plugin repository (icisair50.zip).
2. Select *ICIS-Air* from the **Exchange** drop-down menu. If *ICIS-Air* is not available, ensure that the previous step was completed (*Create ICIS-Air Data Exchange*).
3. Click the **Upload** button to install the ICIS-Air 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 when establishing data services as described later in this section.

Add CDX to the OpenNode2 Partner List

If necessary, add an endpoint for either or both of the CDX Test or CDX Production Nodes, depending on the OpenNode2 deployment environment. Only 2.1 Endpoints are recommended for the ICIS-Air data flow.

CDX Test

Node 2.1 Endpoint

https://testngn.epacdxnode.net/ngn-enws20/services/NetworkNode2ServiceConditionalMTOM

CDX Production

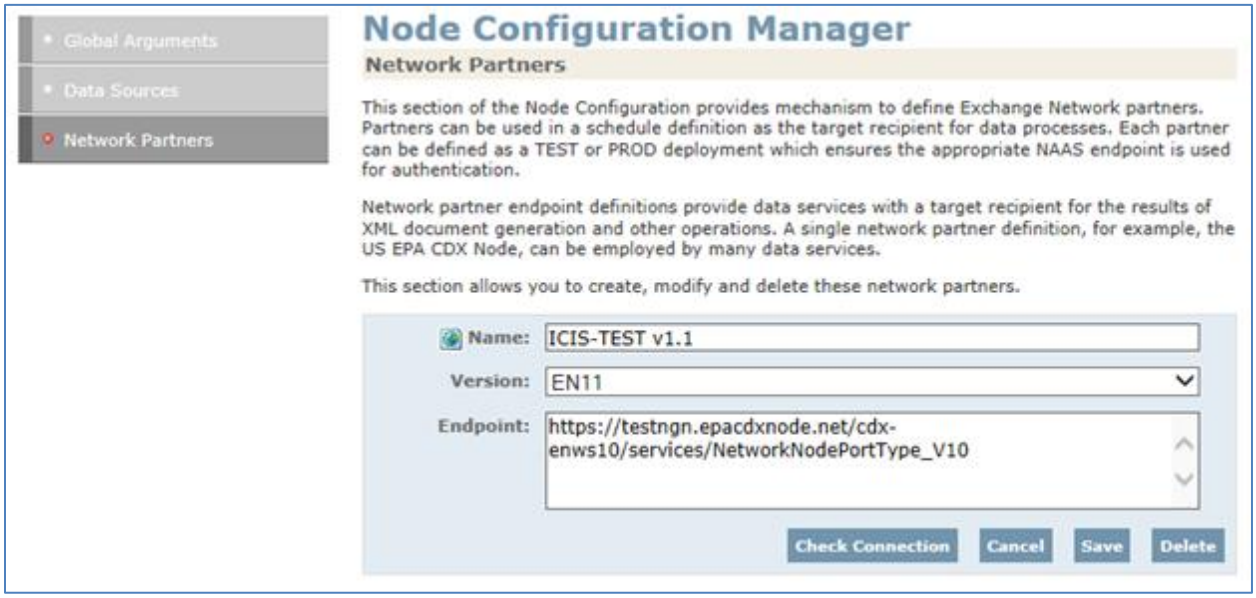
Node 2.1 Endpoint

https://cdxnodengn.epa.gov/ngn-enws20/services/NetworkNode2ServiceConditionalMTOM

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. Review 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 a name that will distinguish between the Endpoint version and test or production status of the CDX Node.
5. Select either *EN11* or *EN21* from the **Version** drop-down menu depending on the Endpoint version of the CDX Node being created.
6. In the **Endpoint** field, type the URL to either the CDX Test or CDX Prod Node listed above.
7. Click the **Check Connection** button to verify that the node connection is successful.
8. Click the **Save** button.

The following screen shot illustrates the proper configuration for the CDX Production Node 1.1 Network Partner definition.



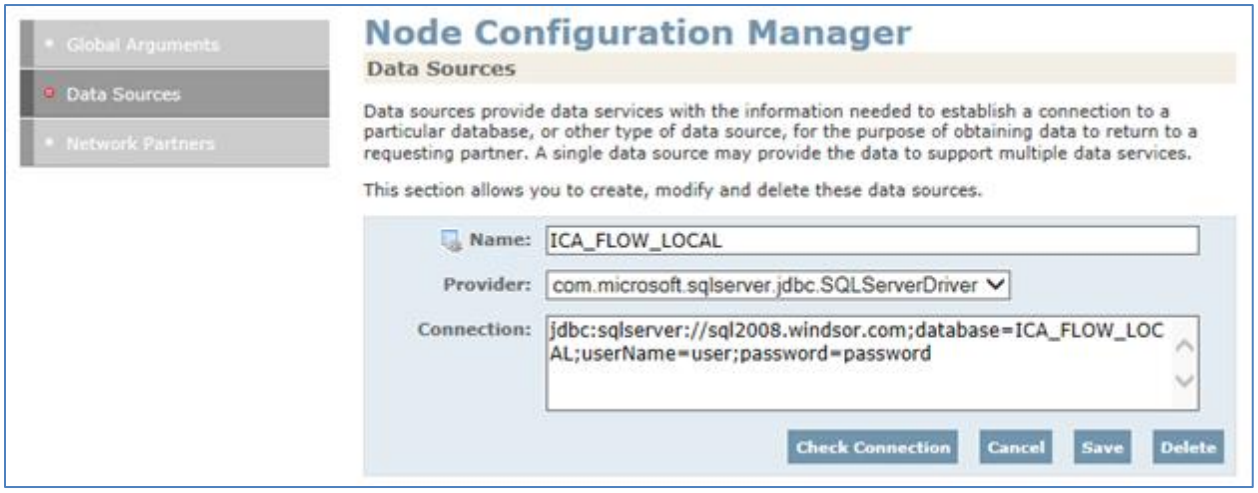
Create Staging Data Source

If necessary, create a data source to define the connection to the ICIS-Air staging databases. Follow the steps below to add a new data source for the ICIS-Air staging database.

1. Click the **Configuration** tab and select **Data Sources** from the left navigation link.

2. Click the **Add Data Source** button on the right-hand side of the page.
3. Enter a name for the data source in the **Name** field. This name will be displayed as an available data source in dropdown menus throughout the Node Admin.
4. Select a database provider from the dropdown list for the data source in the **Provider** field.
5. Enter the connection string for the data source in the **Connection** field. This is the actual connection string value that will be utilized when this data source is selected.
6. Click the **Check Connection** button to verify that the database connection is successful.
7. Click the **Save** button to add the data source to the database and return to the list of available data sources.

The following screen shot illustrates the proper configuration for a typical SQL Server staging database definition.



Create ICIS-Air Data Services

The ICIS-Air plugin includes two data services:

- **PerformICISSubmission** – this service is responsible for sending data to ICIS-Air from the staging environment.
- **GetICISStatusAndProcessReports** - this 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.

Create the PerformICISSubmission Data Service

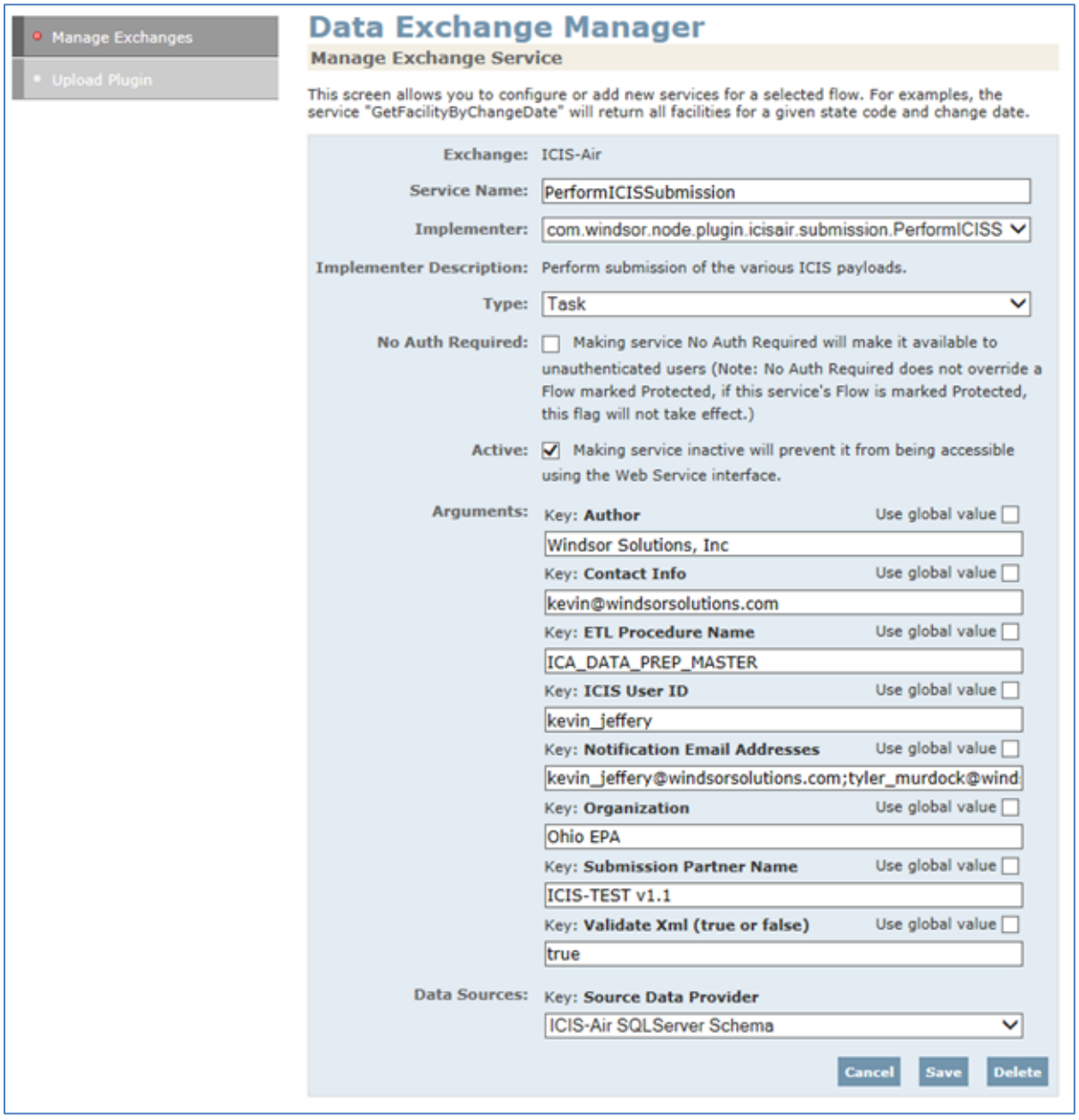
The **PerformICISSubmission** data service will be called by an OpenNode2 schedule to read data from the ICIS-Air staging database, convert the data in XML format for delivery, and submit the resulting XML files to the EPA CDX exchange network partner.

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

1. Click the **Add Service** button located just beneath the ICIS-Air 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-Air 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 but are not critical for processing by ICIS-Air.
7. If the plugin is to be responsible for executing the staging data preparation database procedures, enter the name of the main data preparation stored procedure in the **ETL Procedure Name** field.

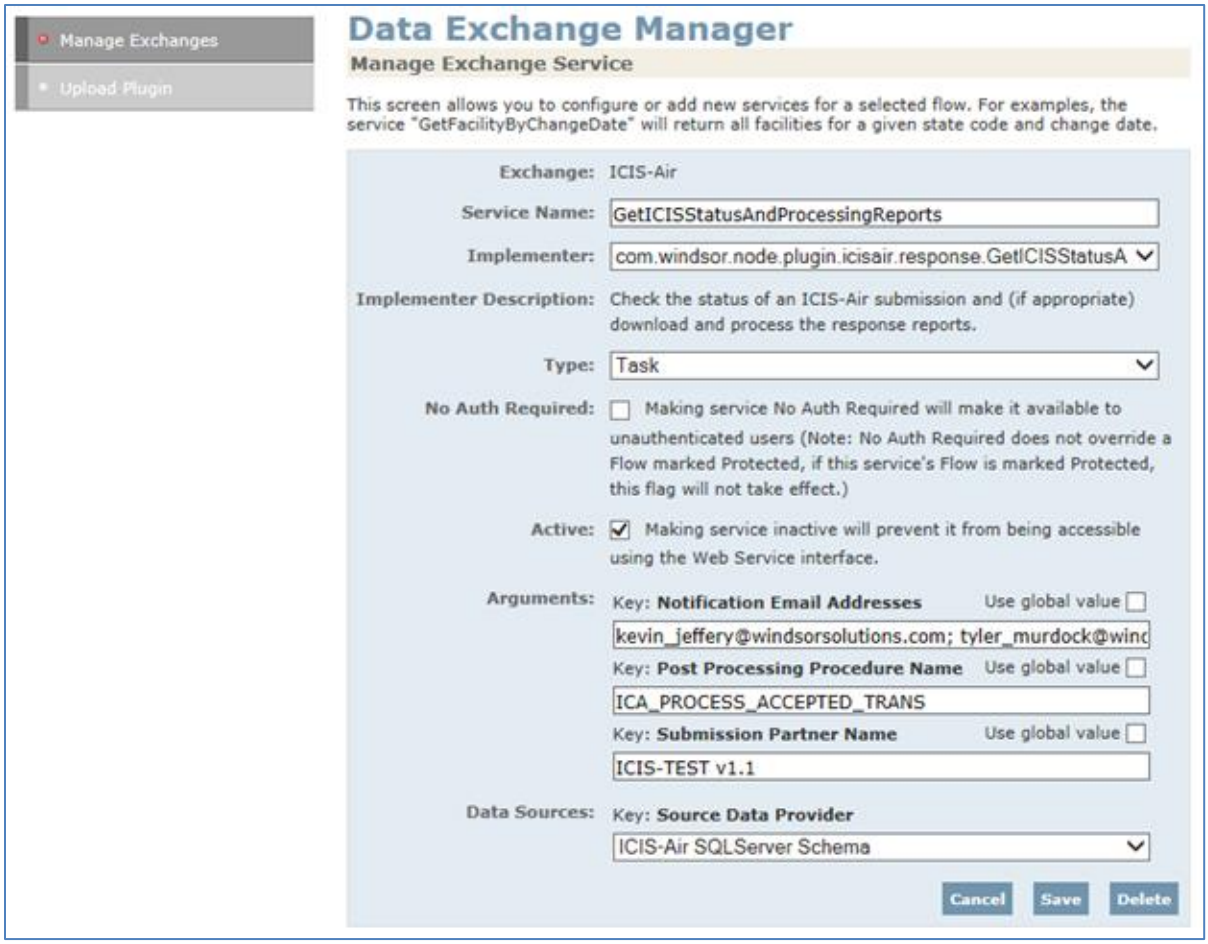
The procedure supplied with the plugin for this purpose is "ICA_DATA_PREP_MASTER". Alternatively, the data preparation can be invoked by some external process in which case this field should be left blank.

8. In 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.
9. 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).
10. 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.
11. In the **Validate Xml** field, type either "true" or "false". It is recommended that this be set to "true".
12. For the argument labeled **Data Sources**, choose the OpenNode2 data source that hosts the ICIS-Air Staging-Local schema/database. The account specified for the data source 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.
13. 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 at EPA CDX, 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-Air data exchange in the list of available exchanges.
2. Click the **Add Service** button located just beneath the ICIS-Air 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 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.
8. Set the **Postprocessing Procedure Name** to "ICA_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.



9. In the **Submission Partner Name** field, type the name of the network partner configured in OpenNode2 for either the CDX Test or Prod environment.
10. For the **Data Sources** argument, choose the OpenNode2 data source that hosts the ICIS-Air 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

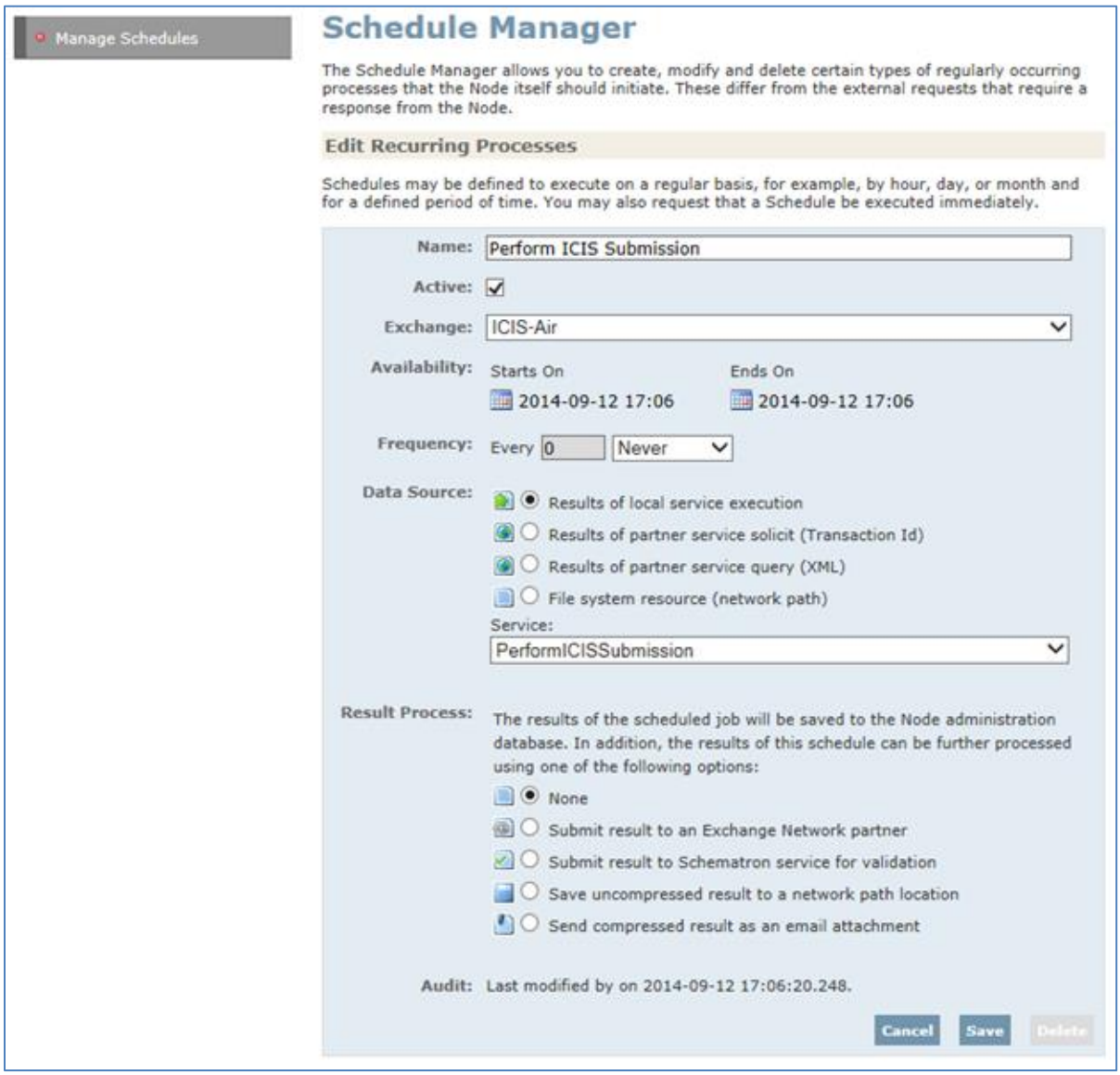
Scheduled jobs can be configured in the OpenNode2 to perform automated tasks, for example, submitting data to external Exchange Network partners or processing received files.

The ICIS-Air data exchange requires two schedules to be established; one to submit data to ICIS-Air and one to check the status of submissions and download/parse reports from ICIS-Air 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-Air" 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. 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-Air service as the data source for the submission.
9. In the **Result Process** area, check the radio button labeled **None.**

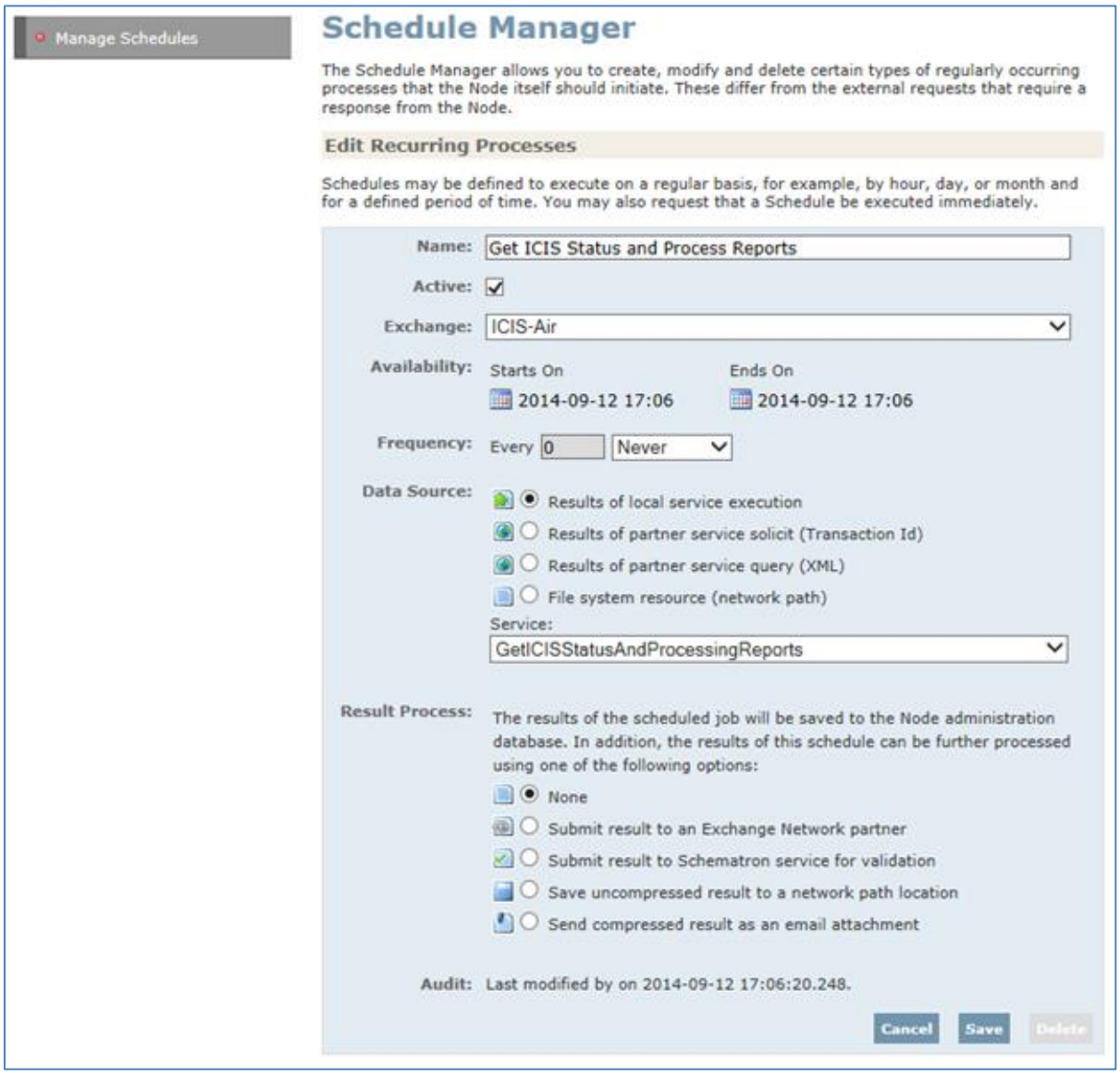
IMPORTANT NOTE:

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.

10. 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 Process Reports" in the **Name** field.
3. Enable the schedule by clicking the **Active** checkbox if not already checked.
4. Select "ICIS-Air" 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. In the **Data Source** area, check the radio button labeled **Results of local service execution**.

8. In the **Service** dropdown box, select the value "GetICISStatusAndProcessReports". This informs the schedule to use the selected ICIS-Air 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-Air 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-Air data exchange on the EPA systems for both the Test and Production CDX Node environments.

Monitor Flow Activity

The OpenNode2 will track all ICIS-Air 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.

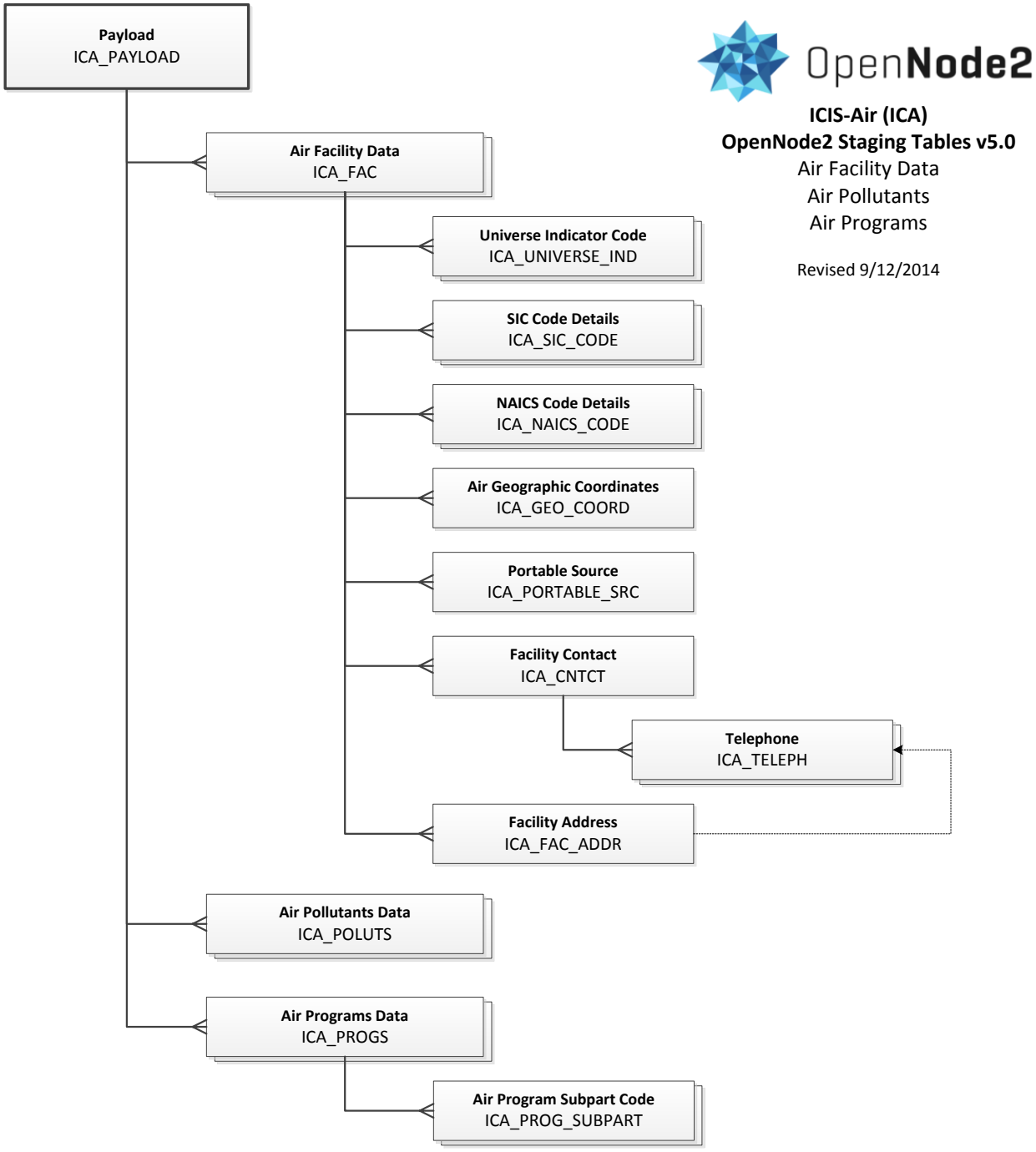
The **ICA_SUBM_RSLTS** table included in the staging database can also be used to view the errors encountered by ICIS-Air for each submission module. This should be used to guide data cleanup efforts in the agency source data system. Once data is corrected and flows successfully to ICIS-Air, the error will automatically clear from the table.

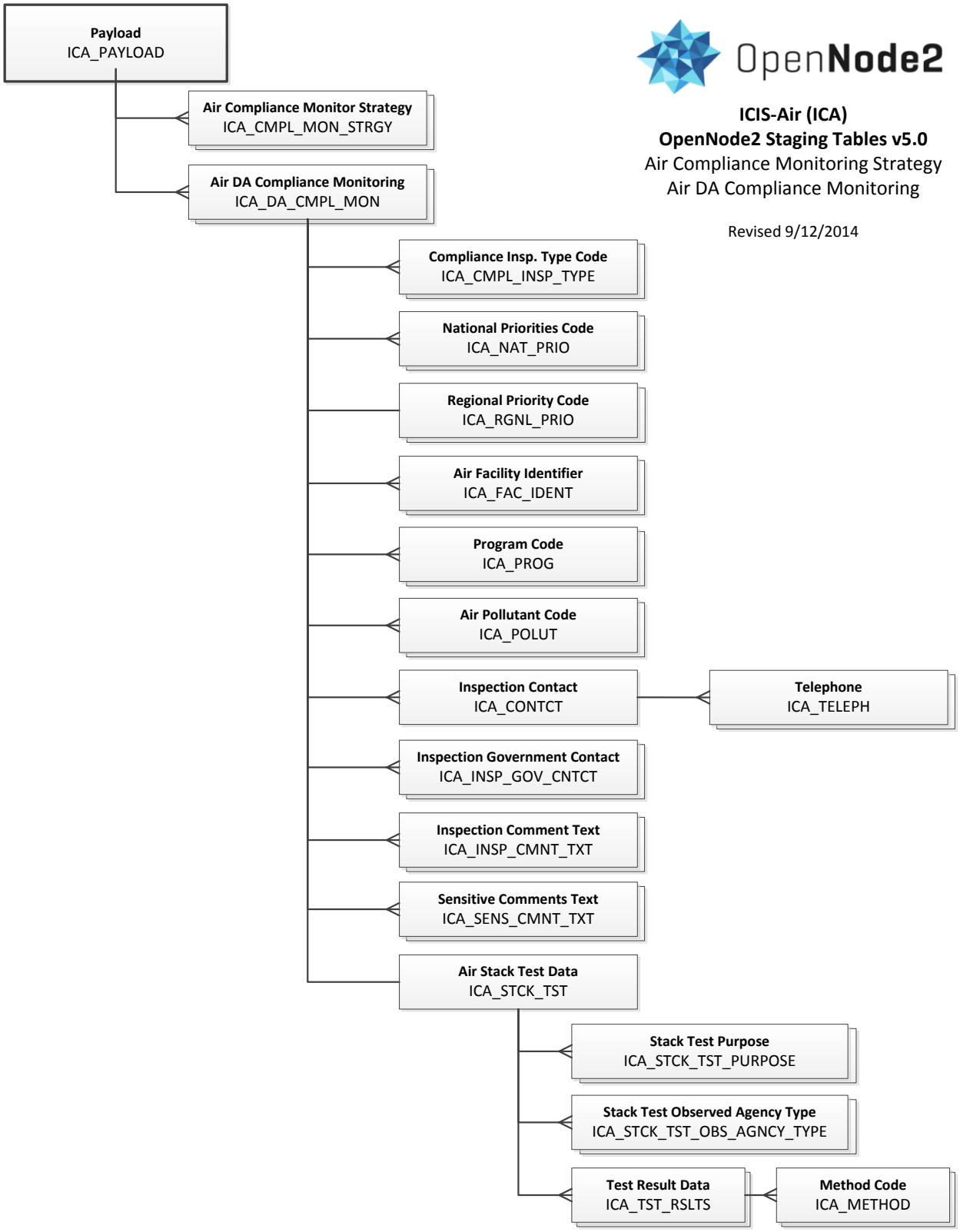
Appendix A - ICIS-Air XML Payload Staging Tables

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

Submission Type	Staging Table Name	Business Key Fields
AirComplianceMonitoringStrategySubmission	ICA_CMPL_MON_STRGY	AirFacilityIdentifier
AirDACaseFileSubmission	ICA_DA_CASE_FILE	CaseFileIdentifier
AirDAComplianceMonitoringSubmission	ICA_DA_CMPL_MON	ComplianceMonitoringIdentifier
AirDAEnforcementActionLinkageSubmission	ICA_DA_ENFRC_ACTN_LNK	AirDAEnforcementActionIdentifier
AirDAEnforcementActionMilestoneSubmission	ICA_DA_ENFRC_ACTN_MILSTN	AirDAEnforcementActionIdentifier MilestoneTypeCode
AirDAFormalEnforcementActionSubmission	ICA_DA_FRML_ENFRC_ACTN	AirDAEnforcementActionIdentifier
AirDAInformalEnforcementActionSubmission	ICA_DA_INFRML_ENFRC_ACTN	AirDAEnforcementActionIdentifier
AirFacilitySubmission	ICA_FAC	AirFacilityIdentifier
AirPollutantsSubmission	ICA_POLUTS	AirFacilityIdentifier AirPollutantsCode
AirProgramsSubmission	ICA_PROGS	AirFacilityIdentifier AirProgramCode
AirTVACCSubmission	ICA_TVACC	ComplianceMonitoringIdentifier
CaseFileLinkageSubmission	ICA_CASE_FILE_LNK	CaseFileIdentifier
ComplianceMonitoringLinkageSubmission	ICA_CMPL_MON_LNK	ComplianceMonitoringIdentifier

Appendix B – ICIS-Air XML Schema/Staging Table Mapping Diagrams





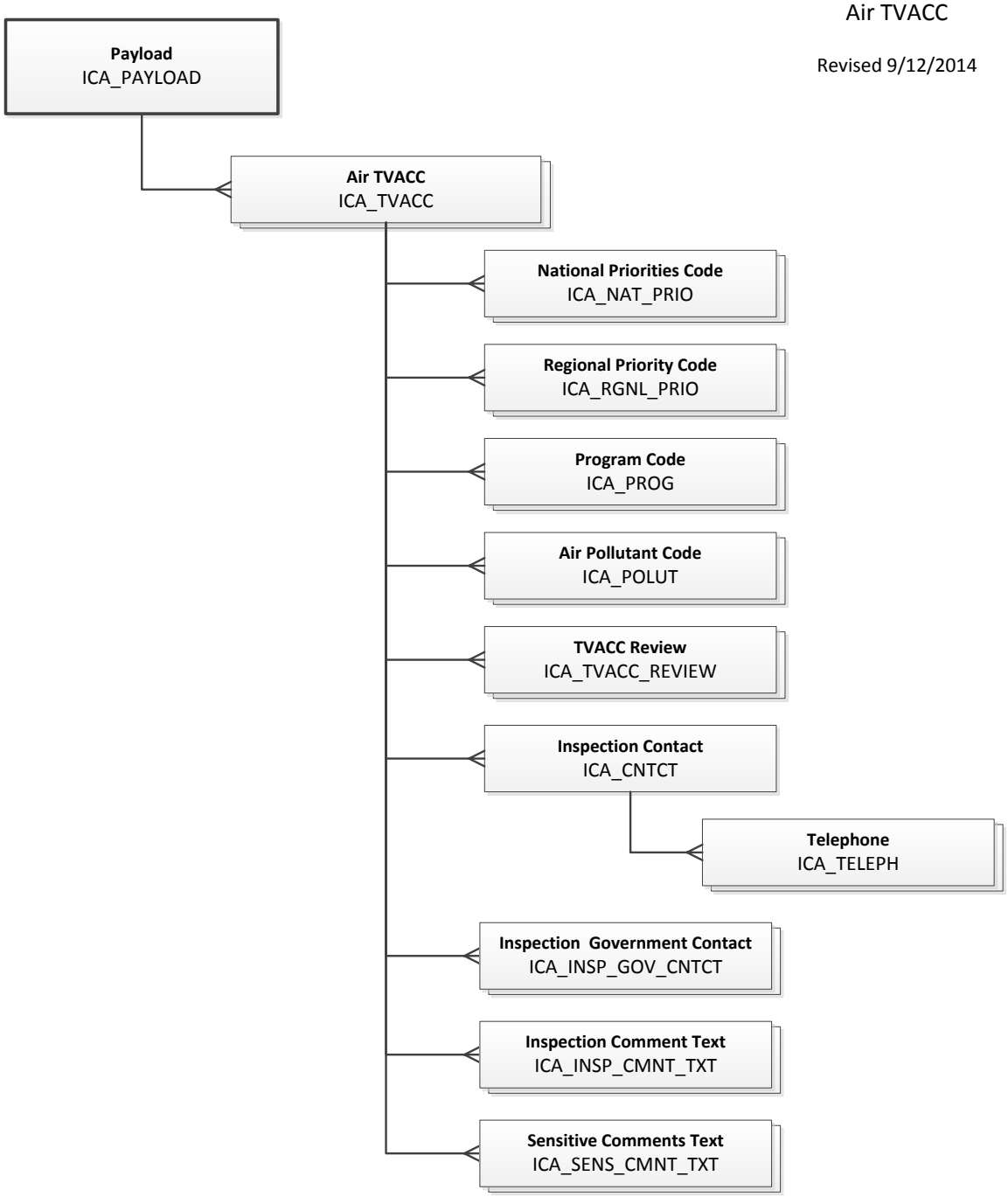
OpenNode2

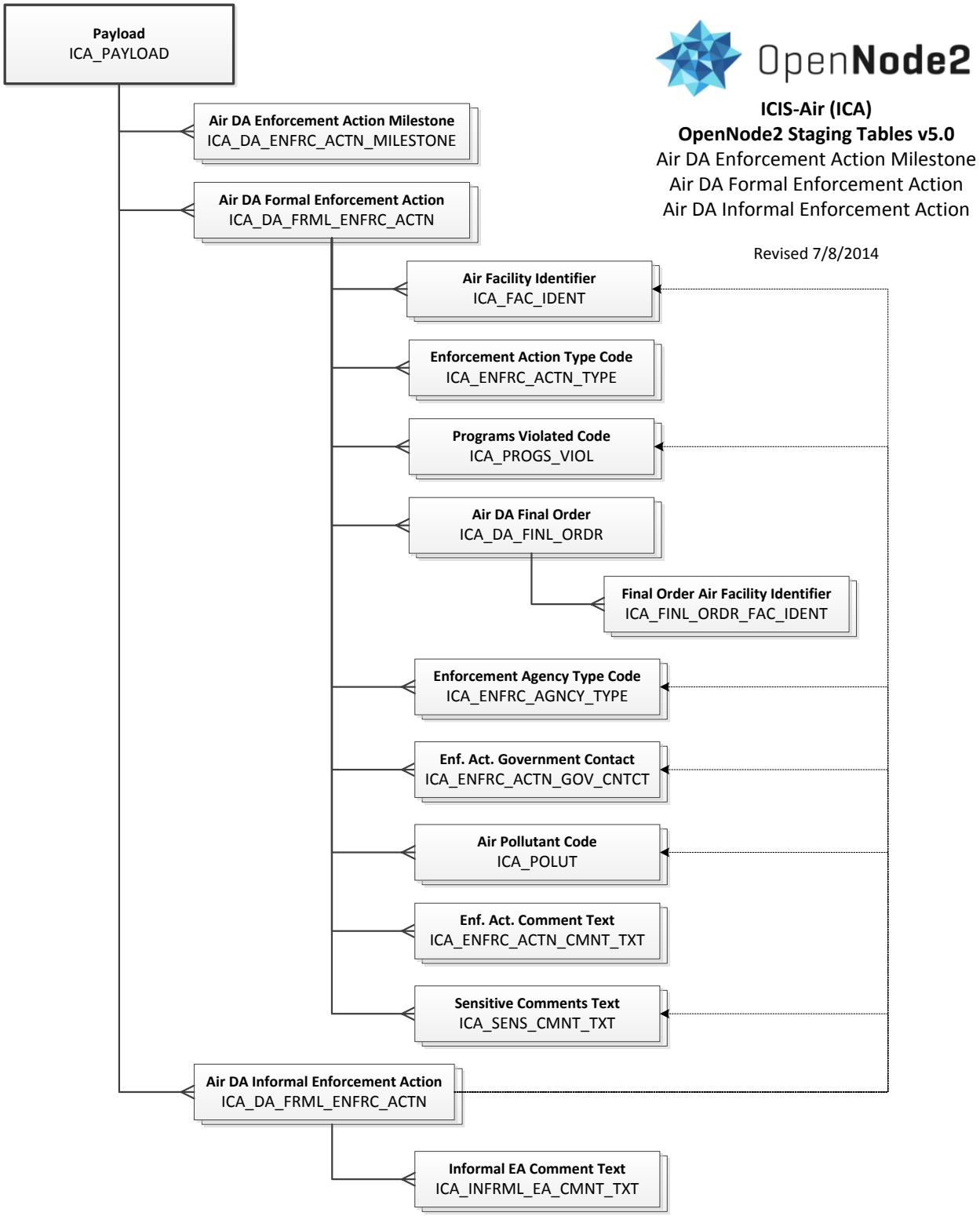
ICIS-Air (ICA)
OpenNode2 Staging Tables v5.0
Air Compliance Monitoring Strategy
Air DA Compliance Monitoring

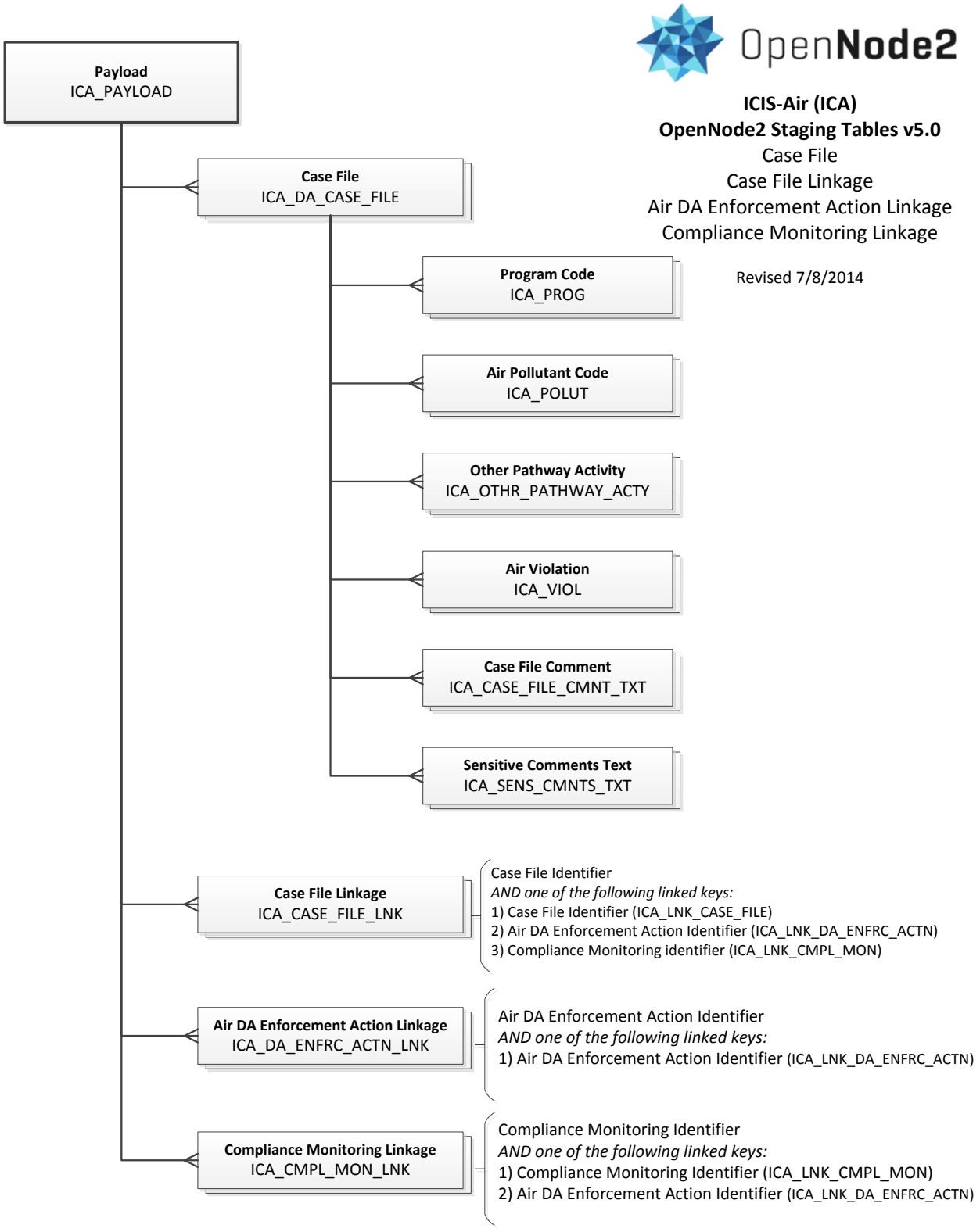
Revised 9/12/2014

ICIS-Air (ICA)
OpenNode2 Staging Tables v5.0
Air TVACC

Revised 9/12/2014




Payload
ICA_PAYLOAD
Air DA Enforcement Action Milestone
ICA_DA_ENFRC_ACTN_MILESTONE
Air DA Formal Enforcement Action
ICA_DA_FRML_ENFRC_ACTN
Air Facility Identifier
ICA_FAC_IDENT
Enforcement Action Type Code
ICA_ENFRC_ACTN_TYPE
Programs Violated Code
ICA_PROGS_VIOL
Air DA Final Order
ICA_DA_FINL_ORDR
Final Order Air Facility Identifier
ICA_FINL_ORDR_FAC_IDENT
Enforcement Agency Type Code
ICA_ENFRC_AGNCY_TYPE
Enf. Act. Government Contact
ICA_ENFRC_ACTN_GOV_CNTCT
Air Pollutant Code
ICA_POLUT
Enf. Act. Comment Text
ICA_ENFRC_ACTN_CMNT_TXT
Sensitive Comments Text
ICA_SENS_CMNT_TXT
Air DA Informal Enforcement Action
ICA_DA_FRML_ENFRC_ACTN
Informal EA Comment Text
ICA_INFRML_EA_CMNT_TXT
OpenNode2
ICIS-Air (ICA)
OpenNode2 Staging Tables v5.0
Air DA Enforcement Action Milestone
Air DA Formal Enforcement Action
Air DA Informal Enforcement Action
Revised 7/8/2014


OpenNode2
ICIS-Air (ICA)
OpenNode2 Staging Tables v5.0
Case File
Case File Linkage
Air DA Enforcement Action Linkage
Compliance Monitoring Linkage
Revised 7/8/2014
Payload
ICA_PAYLOAD
Case File
ICA_DA_CASE_FILE
Program Code
ICA_PROG
Air Pollutant Code
ICA_POLUT
Other Pathway Activity
ICA_OTHR_PATHWAY_ACTY
Air Violation
ICA_VIOL
Case File Comment
ICA_CASE_FILE_CMNT_TXT
Sensitive Comments Text
ICA_SENS_CMNTS_TXT
Case File Linkage
ICA_CASE_FILE_LNK
Case File Identifier
AND one of the following linked keys:
1) Case File Identifier (ICA_LNK_CASE_FILE)
2) Air DA Enforcement Action Identifier (ICA_LNK_DA_ENFRC_ACTN)
3) Compliance Monitoring identifier (ICA_LNK_CMPL_MON)
Air DA Enforcement Action Linkage
ICA_DA_ENFRC_ACTN_LNK
Air DA Enforcement Action Identifier
AND one of the following linked keys:
1) Air DA Enforcement Action Identifier (ICA_LNK_DA_ENFRC_ACTN)
Compliance Monitoring Linkage
ICA_CMPL_MON_LNK
Compliance Monitoring Identifier
AND one of the following linked keys:
1) Compliance Monitoring Identifier (ICA_LNK_CMPL_MON)
2) Air DA Enforcement Action Identifier (ICA_LNK_DA_ENFRC_ACTN)