EMIS CASE STUDY: CREATING AN AUTOMATED AND SUSTAINABLE SOLUTION FOR OIL & GAS MANAGEMENT REPORTING

Prepared By:
Rajesh Mandya
Richard Jenks

TRINITY CONSULTANTS / T3
12770 Merit Drive
Suite 900
Dallas, TX 75251
(972) 661-8100

September 2012

Environmental solutions delivered uncommonly well

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The United States Environmental Protection Agency (EPA) developed the *Mandatory Reporting of Greenhouse Gases; Final Rule* (MRR) to collect greenhouse gas (GHG) data and other relevant information from large sources and suppliers in the United States. The rule was published in the Federal Register (FR) on October 30, 2009 and has been effective since December 29, 2009.[1] On November 30, 2010, the proposed Subpart for Petroleum and Natural Gas Systems, Subpart W, was finalized and published in the Federal Register (FR). Subpart W became effective December 30, 2010.[2] A large oil and natural gas exploration company impacted by this MRR rule choose T3/Trinity Consultants to implement an Environmental Management Information System (EMIS) to track, manage, and report greenhouse gas emissions and other priority pollutants across their organization's operations. The company elected to utilize the IHS opsInfo™ Version 8.0.4 platform for the repository of emission calculations and emission factors developed by the company's environmental group.

The EMIS has been used to generate aggregated corporate emission reports, priority pollutant emissions reports, monthly exports of carbon emissions, and to assist environmental operations in understanding the nature and detail of their emissions impact.

In a typical opsInfo implementation, the hierarchy and equipment objects are prebuilt for companies who have a relatively stable company structure. However, this is a major challenge for oil and gas companies who are constantly changing its organization and equipment ownership/location. To overcome this issue, Trinity Consultants developed a custom application to automate the EMIS implementation. This application collected operational records tracked in external systems and managed future operational and hierarchal changes within opsInfo, as well as:

- Provided the flexibility to change the collected data from various sources in the interface database due to changing production and accounting needs
- Utilized a dynamic tree structure in opsInfo software which is required due to the frequently changing nature of the organizational hierarchy
- Maintained a historical record of object movement or changes in the hierarchy for reporting purposes
- Minimized the need for future manual intervention
- Enabled the enforcement of tight implementation deadlines

The end result was a highly leveraged, custom interface that automatically collected well and facility information, including new operations, as well as the level of activity for operations and equipment scattered throughout the organization's operations. Additionally, field resources provided additional equipment inventory and activity data for sources not previously captured through the application's user interface. All of this information was then integrated into the opsInfo system to accurately represent operations as they change and applied the appropriate regulatory and business rules for calculating emissions from these operations. The solution was deployed to 9000+ well and facility operations and 80,000+ emission sources and tested to ensure the sustainability of the solution as acquisition, divestitures, and new exploration operations changed the data inputs and requirements of the system.

---

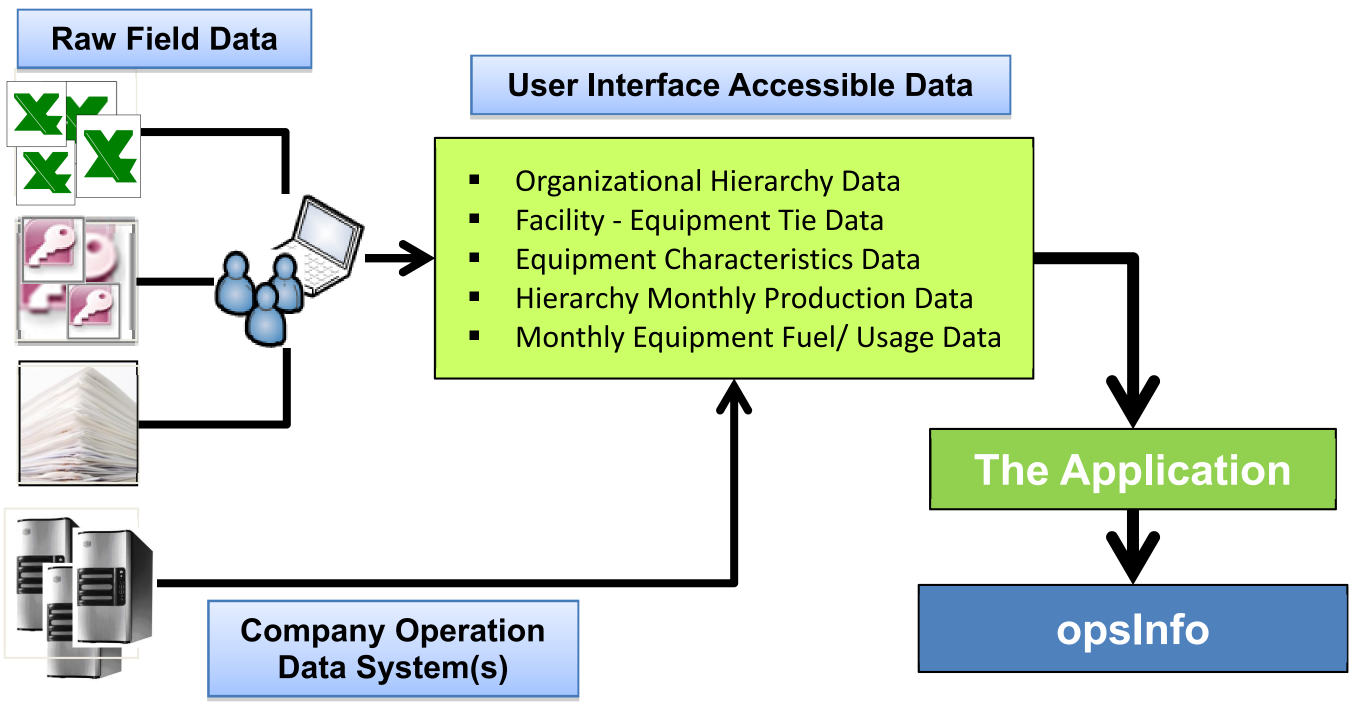
[1] Federal Register, Vol. 74, No. 209, Friday, October 30, 2009, pages 56260 through 56519.

[2] Federal Register, Vol. 75, No. 229, Tuesday, November 30, 2010, pages 74458 through 74515.

DATA GATHERING FUNCTION

The application for the opsInfo GHG emission EMIS was designed to gather data from the various companies' systems along with the ability to enter data by field resources into the user interface for sources not being captured by the company's other system.

Figure 1. Application Interface Diagram



AUTOMATED CREATION OF opsINFO OBJECTS

The application transforms the data through the interface on a routine schedule, as shown in Figure 1, to automatically build the necessary hierarchy level and equipment level objects on the opsInfo tree structure. The objects built on the opsInfo tree are designed to meet various reporting requirements, such as aggregated corporate emission reports, monthly exports of GHG, and criteria pollutant emissions, to a data warehouse.

The application was designed to facilitate the changes requested in the hierarchy due to the movement of equipment to a different location or internal company hierarchy re-organization. In all, approximately 90,000 objects were created involving about 8000 hierarchy **levels** and related equipment.

PRE-DEFINED EQUIPMENT OBJECTS AND AREA LEVEL OBJECTS

Equipment for the oil & gas industry are classified under one of the following 27 equipment types:

- Boats
- Compressor
- Dehydrator
- Drilling Rig
- Electricity
- Engine
- External Combustion Device
- Flare
- Fugitive Dust
- Fugitives
- Gas Jack Internal Combustion (IC)
- Gas Turbine
- Helicopter
- Natural Gas IC Engine
- Other IC Engine
- Pneumatic Device
- Pneumatic Pump
- Tank
- Truck Loading
- Vehicle Combustion
- Venting
- Well Completion
- Well Workover

A pre-defined object class was created to represent each equipment type above. Each pre-defined object class had attributes or parameters that were used to store data related to the hierarchy classes. All parameter sets were inherently defined to contain Begin_Date and End_Date attributes that were common across all parameters. These dates tracked and managed the historical changes to data within an object. Expressions represented equations used to calculate parameters within parameter sets. These equations were derived from various sources, including Subpart W guidance documents published by the U.S. EPA, AP-42 equations and factors, facility calculation methodologies and factors, and example data.

Equipment shared by many wells and fields under an area was classified as an area level object whereas equipment dedicated to a single well or a field was classified as an object under that field or well. For example, boats, drilling rigs, and electricity represented an area level object, whereas compressors, dehydrators, and fugitives (typical in a field or well) represented a well or field level object. Also, data that were common to various equipment residing at the well or field level were stored at the area level and data unique to each piece of equipment were stored at the well level. For example, the dehydrator profiles and flared gas properties were entered at the area level and used by all the objects, whereas the compressor operating hours and tank throughputs were entered as a field or well level object via the data from the various companies' systems.

HANDLING OPERATIONAL CHANGES

One of the critical factors of success in the application's design was to accurately represent operations as they change and apply the appropriate regulatory and business rules for calculating emissions from these operations. The changes in the operations could be broadly divided into two categories: hierarchy level and equipment level changes. Under each category the changes could be further subdivided into four subcategories.

Movement: Equipment, such as compressors or pumps, moved from one well to a different well were handled by the application by moving the equipment from one parent to the other on the hierarchy tree structure. Additionally, the application updated the time stamp on the end date for the row bearing the previous well equipment relationship and a new row was inserted with the updated well equipment relationship and new begin date. Likewise, an organization change like a team, area, field, facility or a well being moved from one business unit, team, area, field or facility was handled by moving the entire branch bearing the respective item and all the items under to the new model along with changing the parent child relationships.

Change: Changes to the attributes of equipment or organization class objects were done by updating the time stamp on the end date for the row bearing the previous attributes and inserting a new row with the updated attributes bearing the new begin date.

Addition: Adding a new organization group or equipment class resulted in the creation of new objects on the hierarchy tree structure with the right parent class along with the addition of a row of data to represent the various attribute values.

Removal: Any organization group or equipment being removed from service were deactivated in the opsInfo system. If at any point in time a decision was made to bring them back, the objects could be reactivated in the application.

The application was built to handle various data integrity and error handling issues as well as manage the following tasks:

- Avert building duplicate objects by checking the new object to be built against the objects already on the tree structure
- Flag all of the processed rows and avoid reprocessing the same rows
- Record a log of each object built, change executed, and error that occurred
- Prevent double processing rows during process interruptions and resume from the point where the processing was stopped
- Detect user defined exceptions for cases such as no data found to avoid run time errors

T3/Trinity Consultants successfully implemented the opsInfo software using our custom application to create more than 90,000 objects in the organization and equipment hierarchy. The implementation was done in a cost and time effective manner and deployed in five short months. The following table compares the number of hours required to implement the solution using the standard, manual manner versus using our custom, automated application.

Table 1. Comparison of Implementation Time

Task	Custom, Automated Application (Hours)	Standard, Manual Process (Hours)
Build New Classes	16	16
Build the Application	500	
Test the Application	300	
Build Objects	400	4500
Build Reports	300	300
Total Implementation Time	1516	4816

Since deploying the custom solution in 2010, the application was tested by the following circumstances:

1. The company went through two extensive overhauls of the corporate hierarchy since deployment. The application was tested on its ability to handle hierarchical changes during this massive reorganization. It's performance was critical to the sustainability of the system.

2. The EPA made changes to the GHG rule, emission calculation equations, and reporting requirements. The mandated changes were effectively applied by making a few adjustments to the application's code.

3. The application's on going sustainability made acquisition of another company smooth and easy.

The application concept was built and deployed for several other oil and gas companies to manage their Subpart W requirements as well as at a pharmaceutical company to manage its batch chemical formulation/recipe. It was also used to deploy 150,000 new objects and expressions as a global metric reporting solution for a large, integrated petroleum company with worldwide operations.

CONCLUSION

T3/Trinity Consultants' challenge was to design and implementation a highly automated solution that was capable of tracking equipment and organization changes as well as maintain a historical record of these changes Typical change scenarios include the following:

- Movement of equipment from one facility to other
- Active equipment going out of service
- Inactive equipment becoming active
- Renaming of equipment or a facility
- Reorganization changes to the entire tree structure or being renamed
- Data changes made to the equipment like flow rate etc

In addition to recordkeeping, object changes needed to move with the object to facilitate easy viewing of changes and reproduction of accurate reports before and after the change. Prototypes were built and a significant amount of time was spent testing production data for each conceivable situation to ensure the application was a sustainability solution.

Lastly, additional status reports had to be built to inform users when changes were made. These reports were necessary to notify system users in real time as the application made these automated changes.