# RxE OpenTrac Update

This document describes the activity for the Rxe OpenTrac Upgrade project. This includes an explanation of the changes, the resulting Rxe Configuration and the creation of the OpenTrac RxE plugins.

Vendor Data to SYNC Tables Group 1

Contents

[RxE OpenTrac Update 1](#_Toc460998273)

[Overview 3](#_Toc460998274)

[Messages 3](#_Toc460998275)

[Maximo Vendor Message: MO01 4](#_Toc460998276)

[MO01 Vendor Message Overview 4](#_Toc460998277)

[NS\_OPENTRAC\_NS\_MD01 6](#_Toc460998278)

[MD01 Message XLSX 6](#_Toc460998279)

[OpenTrac Database: VENDOR\_MASTER\_TX 7](#_Toc460998280)

[Sample Data 8](#_Toc460998281)

[Maximo – Purchase Requisitions (PREQ) OM02 and MO02 9](#_Toc460998282)

[OpenTrac PREQ: OM02 9](#_Toc460998283)

[Maximo Sample 9](#_Toc460998284)

[OpenTrac PREQ DB: PREQ\_HEAD\_TX, PREQ\_LINE\_TX 10](#_Toc460998285)

[OpenTrac PREQ: M002 12](#_Toc460998286)

[Maximo Sample 12](#_Toc460998287)

[OpenTrac PREQ Response DB: PREQ\_RESP\_HEAD\_TX, PREQ\_RESP\_LINE\_TX 13](#_Toc460998288)

[Appendix – OpenTrac Director Documentation 14](#_Toc460998289)

[OpenTrac Director User Guide 14](#_Toc460998290)

[Synchronization 14](#_Toc460998291)

[Appendix – Local Testing 18](#_Toc460998292)

# Overview

Messages are consistent with the existing NSBSL RxE configuration, with messages of a specific type (such as Vendor) being named starting with a two-character descriptor that indicates the source and target systems, and then two digits that are assigned to the message type.

For example, consider Maximo-OpenTrac messages, such as MO01. The MO indicates Maximo is the source, and OpenTrac is the target. The 01 is assigned to the Vendor messages. If there are variations of the MO01, then they are assigned MO01a, MO01b, etc.

This document explains what data is being moved, and how RxE is configured.

### Messages

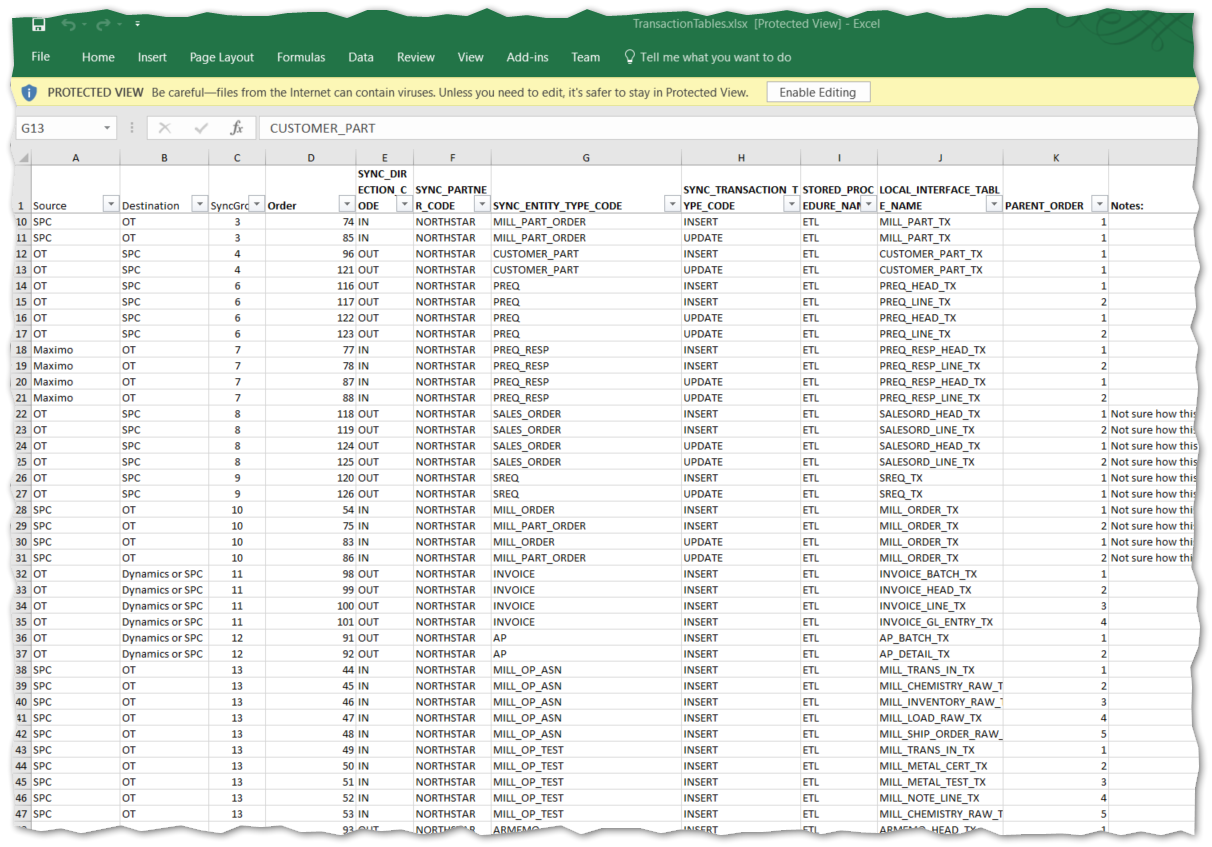


Figure - Some of Messages being Created

# SPC Messages

The SPC-OpenTrac messages are all database to database. Messages that are collected via a single ‘SKEY’ transaction (in either direction) are placed in a single message with <NewDataSet> elements delimiting messages from the same table, and <{table-name}> elements delimiting each record. The

When moving from-or-to a database, all data – including the updating of the TRANSACTION table is done within a DB Transaction.

## Message SO01

# Maximo Messages

# Maximo Vendor Message: MO01

### MO01 Vendor Message Overview

The MO01 message is created by Maximo for consumption by OpenTrac.

Maximo will create messages under the following conditions:

* Vendor Type is F or P (ns\_vendortype)
* Vendor is not disabled
* Vendor is Approved
* One of the associated attributes has changed, or a new record is approved.

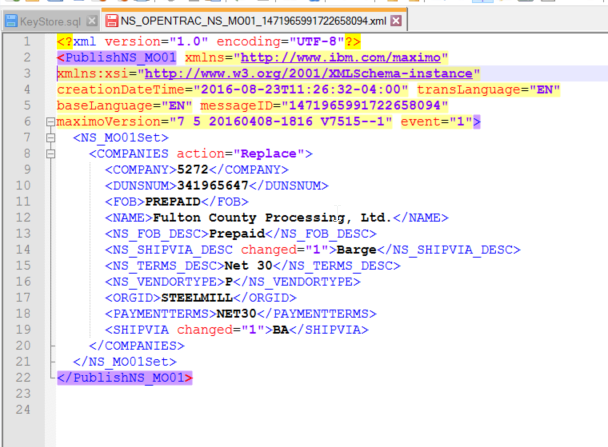
The message from Maximo will be created as NS\_OPENTRAC\_NS\_MO01\_*1234*.xml

Attribute Mapping:

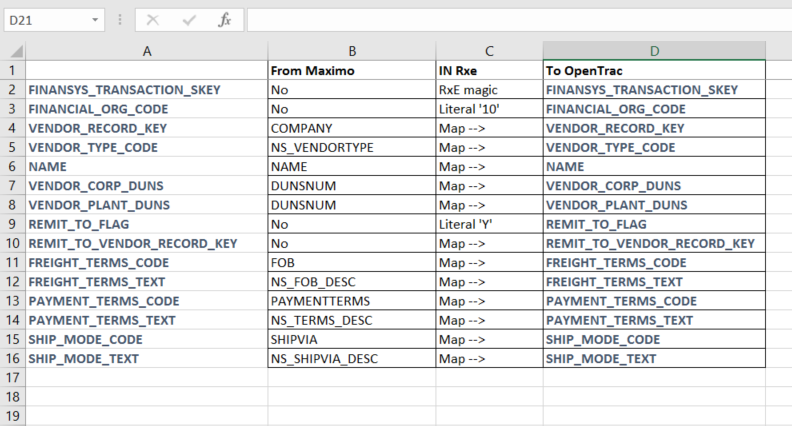
|  |  |  |
| --- | --- | --- |
| **From Maximo** | **IN Rxe** | **To OpenTrac** |
| No | RxE magic | **FINANSYS\_TRANSACTION\_SKEY** |
| No | Literal '10' | **FINANCIAL\_ORG\_CODE** |
| COMPANY | Map --> | **VENDOR\_RECORD\_KEY** |
| NS\_VENDORTYPE | Map --> | **VENDOR\_TYPE\_CODE** |
| NAME | Map --> | **NAME** |
| DUNSNUM | Map --> | **VENDOR\_CORP\_DUNS** |
| DUNSNUM | Map --> | **VENDOR\_PLANT\_DUNS** |
| No | Literal 'Y' | **REMIT\_TO\_FLAG** |
| NS\_VENDORTYPE | Map --> | **REMIT\_TO\_VENDOR\_RECORD\_KEY** |
| FOB | Map --> | **FREIGHT\_TERMS\_CODE** |
| NS\_FOB\_DESC | Map --> | **FREIGHT\_TERMS\_TEXT** |
| PAYMENTTERMS | Map --> | **PAYMENT\_TERMS\_CODE** |
| NS\_TERMS\_DESC | Map --> | **PAYMENT\_TERMS\_TEXT** |
| SHIPVIA | Map --> | **SHIP\_MODE\_CODE** |
| NS\_SHIPVIA\_DESC | Map --> | **SHIP\_MODE\_TEXT** |

### NS\_OPENTRAC\_NS\_MD01





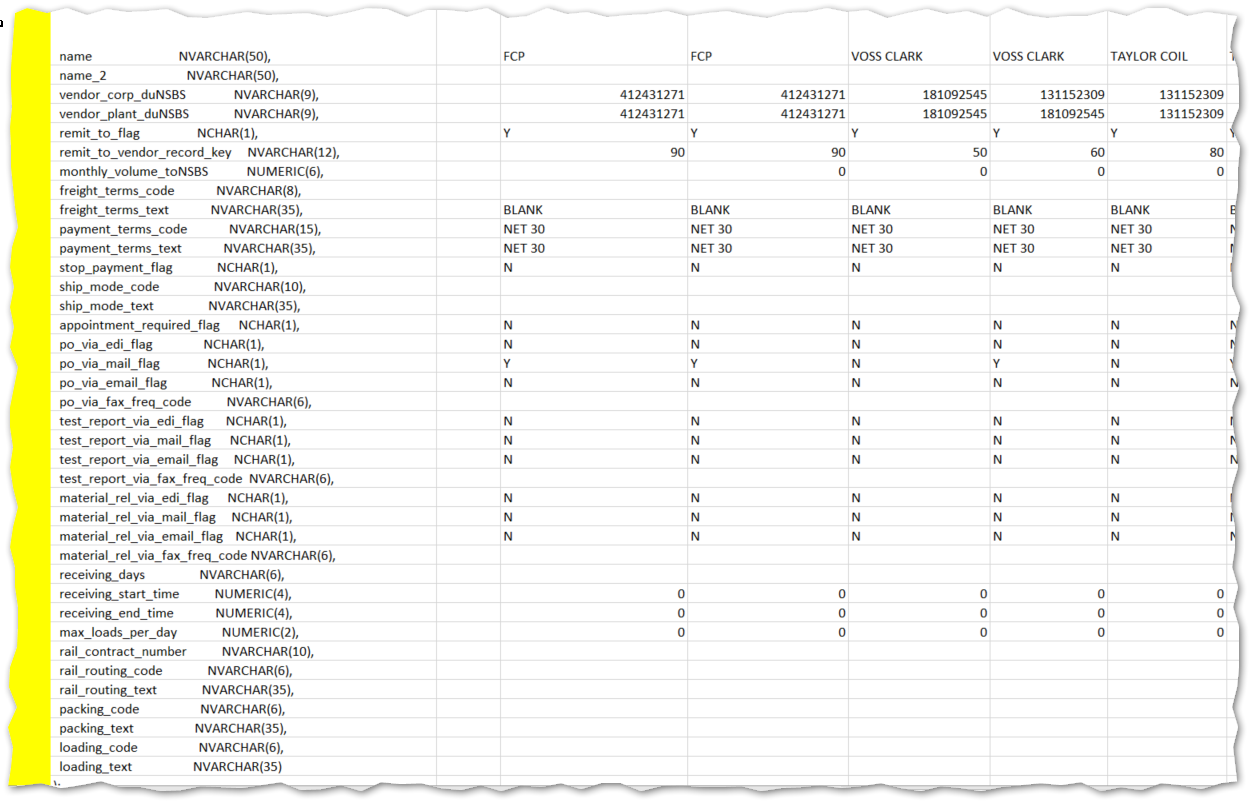
### MD01 Message XLSX



### OpenTrac Database: VENDOR\_MASTER\_TX



### Sample Data



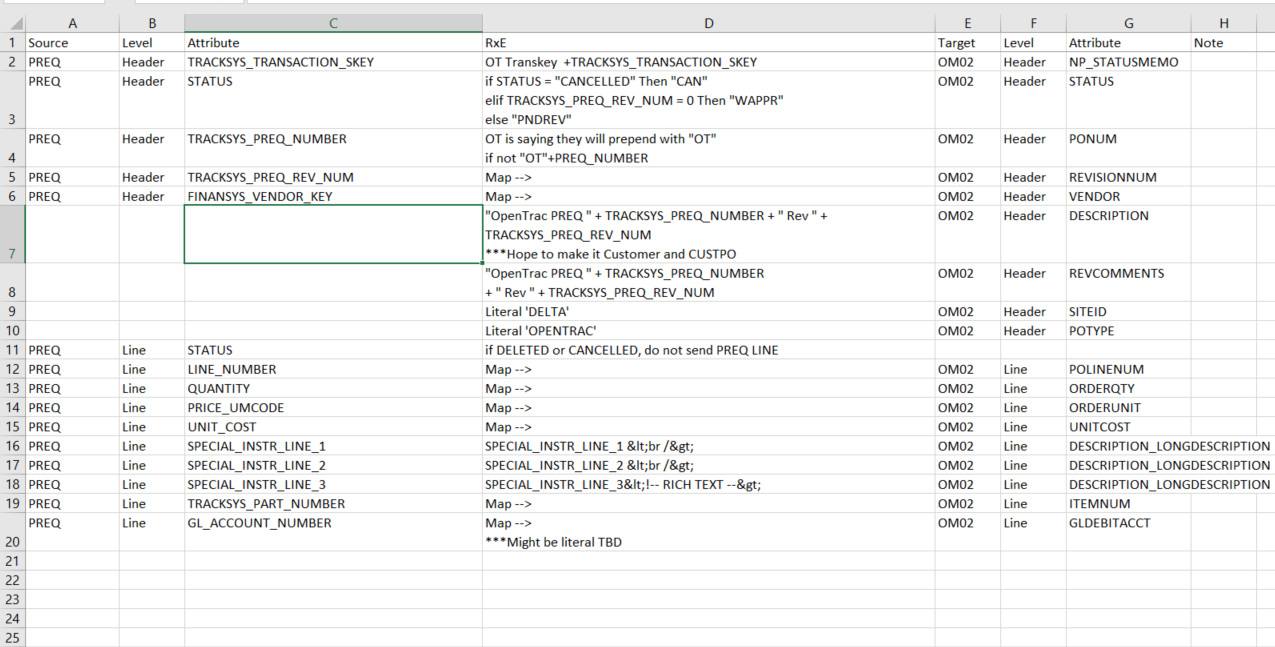
# Maximo – Purchase Requisitions (PREQ) OM02 and MO02

This sections discusses the messages PREQ and the responses (PREQ\_RESP)

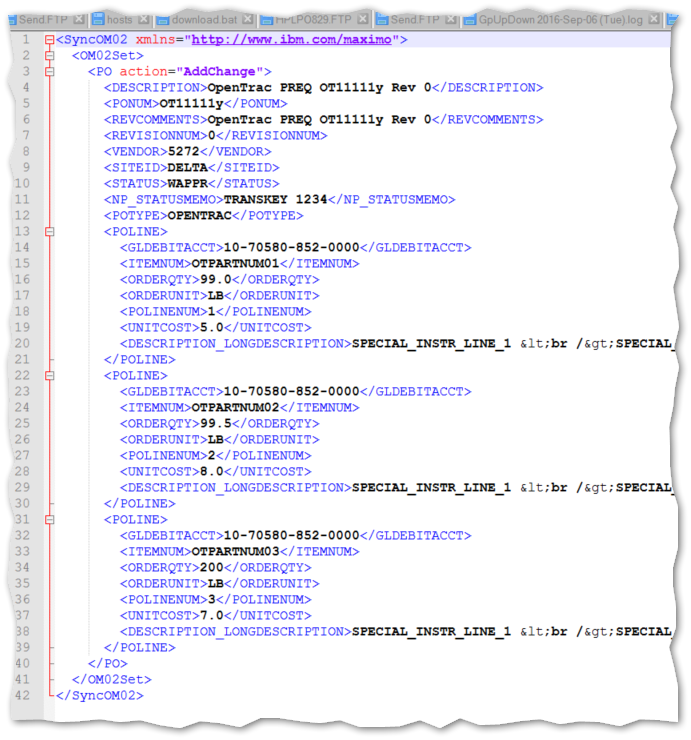
## OpenTrac PREQ: OM02

Message received from OpenTrac

Maximo Mapping (OT\_MX Messages.XLSX)



### Maximo Sample



### OpenTrac PREQ DB: PREQ\_HEAD\_TX, PREQ\_LINE\_TX

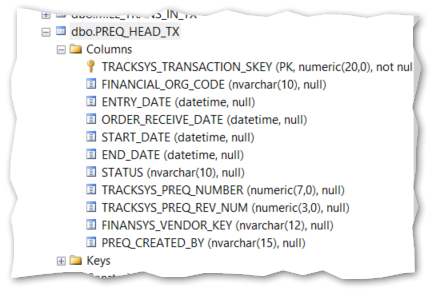


Figure 2- PREQ\_HEAD\_TX

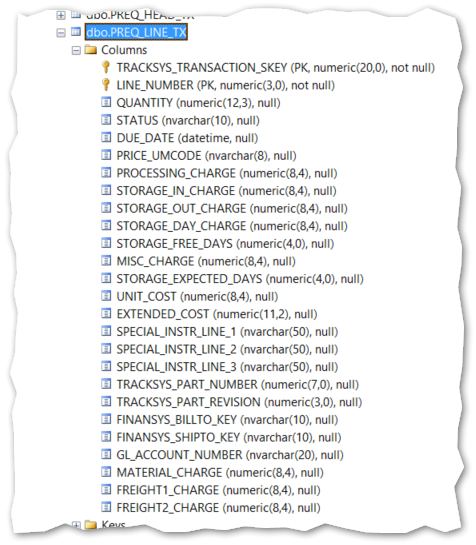
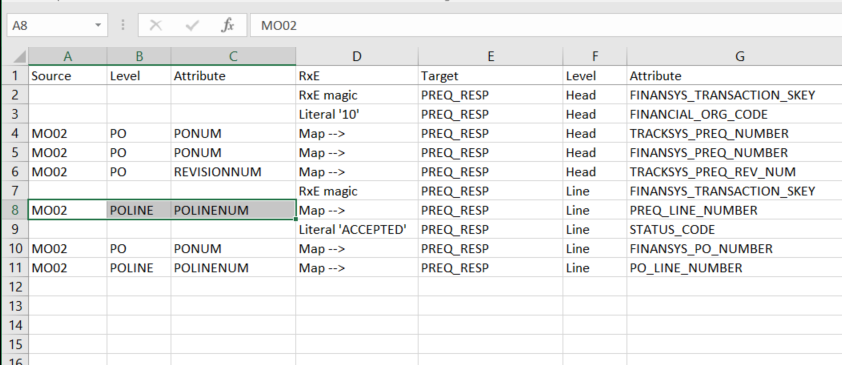


Figure 3- PREQ\_LINE\_TX

## OpenTrac PREQ: M002 – PREQ Response

Message sent to OpenTrac



### Maximo Sample

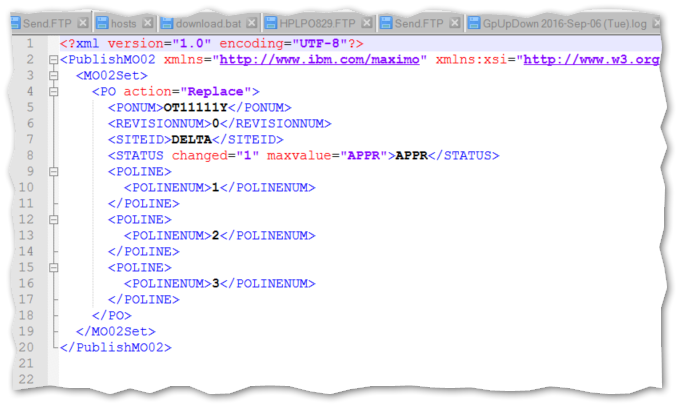


Figure - MO02 - PREQ Response

### OpenTrac PREQ Response DB: PREQ\_RESP\_HEAD\_TX, PREQ\_RESP\_LINE\_TX

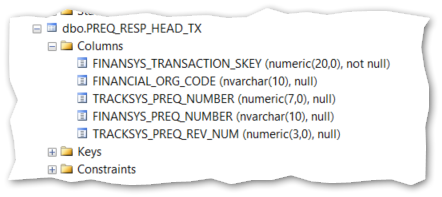


Figure 5- PREQ\_RESP\_HEAD\_TX

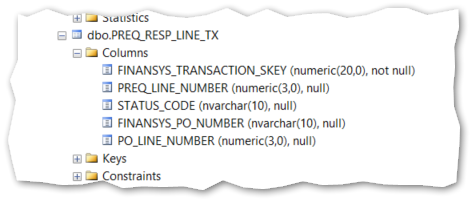


Figure 6- PREQ\_RESP\_LINE\_TX

# Appendix – OpenTrac Director Documentation

## OpenTrac Director User Guide

### Synchronization

* **Purpose**
  + Provide abilities to synchronize Opentrac with the business systems of its users
  + Manage the incoming and outgoing transactions through synchronization
  + Monitor the incoming and outgoing transactions through synchronization
* **Background Information**
  + There are 2 major kinds of synchronization (SYNC) transactions
    - Inbound Transactions – These are the transactions that are originated from the user’s home business systems and destined towards Opentrac
    - Outbound Transactions – These are the transactions that are originated from Opentrac and destined towards the user’s home business systems
  + There are two main types of Inbound SYNC Transactions
    - Inbound SYNC Transactions for sake of Order Management in Opentrac
      * Customer Masters with their contacts information (abbr. CUSTOMER\_MASTER)
      * Vendor Masters with their contacts information (abbr. VENDOR\_MASTER)
      * Mill Parts (abbr. MILL\_PART)
      * Purchase Order Requision Responses (abbr. PREQ\_RESP)
      * Mill Orders (abbr. MILL\_ORDER)
      * Cost &/or Sale Figures for Profitability Analysis (abbr. PAE)
    - Inbound SYNC Transactions to carry EDI Data (by-pass EDI system)
      * ASN from Mills to Outside Processors (abbr. MILL\_OP\_ASN)
      * Test Results from Mills to Outside Processors (abbr. MILL\_OP\_TEST)
      * ASN from Mills to Customers (abbr. MILL\_CUST\_ASN)
      * Test Results from Mills to Customers (abbr. MILL\_CUST\_TEST)
  + Outbound SYNC Transactions
    - Customer Parts (abbr. CUSTOMER\_PART)
    - Accounts Payable Batches (abbr. AP)
    - Accounts Receivable Batches (abbr. INVOICE)
    - Inventory with add-on data of validate EDI Transactions (abbr. EDI)
    - Sales Orders (abbr. SALES\_ORDER)
    - Purchase Requisions (abbr. PREQ)
    - Steel Requisions (abbr. SREQ)

All of the SYNC transactions are managed through Opentrac. SYNC transactions are handled by four different Queues - that are controlled by Opentrac System - which are asynchronously fulfilling synchronization jobs.

* **Before you begin**
  + The following duties should be completed before SYNC systems can be used
    - All of the SYNC tables should be created in the remote business systems
    - Opentrac System should be able to access the SYNC tables of the remote business systems
    - Opentrac System should have the following privileges on the SYNC tables of the remote business systems
      * Insert
      * Delete
      * Update
    - Interface Check tool of the SYNC System should be able to validate all of the remote SYNC tables
  + Opentrac SYNC System supports all of the following types of remote Relational Database Management Systems (current version of Opentrac supports Oracle 9i and above in any platform, and IBM DB2 4.5 and above through IBM DB2 Enterprise Connect Gateway in IBM AS400)
    - All ODBC compliant databases
    - All OLE-DB compliant databases
* **General data entry flow and rules of the Opentrac SYNC Interface**

1. Explanation of the terms used in the Opentrac SYNC Interface  
   1. ***Remote Interface Tables (a.k.a. Synchronization Tables):*** *The synchronization tables that reside in the remote database management system for the sake of communication with Opentrac System.*
   2. ***Remote Parent Interface Table:*** *A transaction table that has at least one Child Table. Parent Tables have header level information. In other words, for every one record in a Parent Table there can be zero, one or more than one record in its child tables.*
   3. ***Remote Child Interface Table:*** *A transaction table that has a parent table. Child Tables have detailed level of information. In other words, for every one or more than one record in a Child Table, there is always one record in its Parent Table.*
   4. ***Transaction:*** *Transmission attempt to move collection of data that serves for the achievement of only one specific business need.*
   5. ***Inbound Transactions:*** *This term is used relative to – in other words, from the perspective of – the Opentrac System. Any transaction that is originated from Remote Business Systems via Opentrac Systems’ Interface, destined to Opentrac System is called an Inbound Transaction.*
   6. ***Outbound Transactions:*** *This term is used relative to – in other words, from the perspective of – the Opentrac System. Any transaction that is originated from Opentrac System via Opentrac Systems’ Interface, destined to Remote Business System is called an Outbound Transaction.*
   7. ***Listeners:*** *Asynchronously running applications that are used to queue new transactions and process queued transactions on different scheduled intervals.*
2. Each Remote Parent Interface Table will have unique keys across the board. Child tables will have the same Transaction IDs with their Parent Tables. Summary Tables will have the Transaction IDs from the related Parent Interface Tables.
3. Summary Tables have the following information:  
   1. TRANSACTION\_INBOUND Table
      1. TRACKSYS\_TRANSACTION\_SKEY 🡪 Unique Key across the Parent Transaction Tables
      2. ENTITY\_TYPE\_CODE 🡪 Code of the Inbound Transaction: i.e. CUSTOMER, VENDOR, etc
      3. TRANSACTION\_TYPE\_CODE 🡪 Insert, Delete or Update
      4. SUBMISSION\_DATETIME 🡪 The Date & Time of the creation of the Inbound Transaction
   2. TRANSACTION\_OUTBOUND Table
      1. TRACKSYS\_TRANSACTION\_SKEY 🡪 Unique Key across the Parent Transaction Tables
      2. ENTITY\_TYPE\_ CODE 🡪 Code of the Inbound Transaction: i.e. MILL\_ORDER, etc
      3. TRANSACTION\_TYPE\_CODE 🡪 Insert, Delete or Update
      4. SUBMISSION\_DATETIME 🡪 The Date & Time of the creation of the Outbound Transaction
      5. PROCESSING\_STATUS\_CODE 🡪 New, Complete, Fail\_Temp, Fail\_Perm, Received
4. Inbound and Outbound transactions will be processed as FIFO (First In First Out) way.
5. Remote Interface Tables will have unique transaction numbers that will be coming from Remote Business Systems and they will be unique across the all of the Remote Interface tables.
6. The following must happen to update one of the previously created transactions:
   1. A unique Transaction ID has to be generated by the remote Business System
   2. Same unique Transaction ID for the specific transaction must be used
   3. The related Parent Transaction Table must be populated
   4. The related Child Table(s) must be populated
   5. A matching transaction summary record must be created in the TRANSACTION\_OUTBOUND Table
   6. In the TRANSACTION\_OUTBOUND Table, TRANSACTION\_TYPE\_CODE must be set as “UPDATE”
   7. An Update transaction will find the matching record in the Opentrac System and over-write every related record with all the data that is passed over. The fields that are left as NULL will be set as NULL
7. Opentrac System will
   1. Maintain its own unique Transaction IDs for each and every transaction
   2. Expect unique Remote Transaction IDs for each and every transaction
   3. Expect Remote Transaction IDs, which are previously recorded as a result of successful transactions, to be maintained and never changing or deleted from the Remote Business Systems
8. There are Transaction Queue Tables in the Opentrac system which will have summary data of Inbound and Outbound Transactions
9. Both Inbound and Outbound Transactions will be processed by the Opentrac Listeners.
10. There will be four types of listener applications:  
    1. REMOTE\_INBOUND\_QUEUE\_LISTENER\_<Financial\_Organization>
    2. LOCAL\_INBOUND\_QUEUE\_LISTENER\_<Financial\_Organization>
    3. REMOTE\_OUTBOUND\_QUEUE\_LISTENER\_<Financial\_Organization>
    4. LOCAL\_OUTBOUND\_QUEUE\_LISTENER\_<Financial\_Organization>
11. There will be a separate listener running for each type of the listener given the Financial Organization.
12. The data in the Remote Interface Tables (the synchronization tables that reside in the Remote Business Systems) will never be deleted by the Remote Business Systems. Opentrac system will run a janitor service to do this job when the systems settle down after all of the phases of development is over.
13. TRANSACTION\_INBOUND and TRANSACTION\_OUTBOUND record must be created after the related transaction records are created.

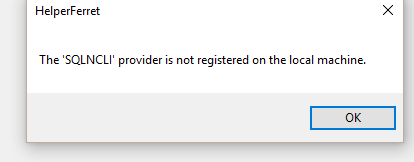
* **Screen Shots**

When there are records that exceed 500, system stops at 500 records and give users the option to continue or stop

# Appendix – Local Testing

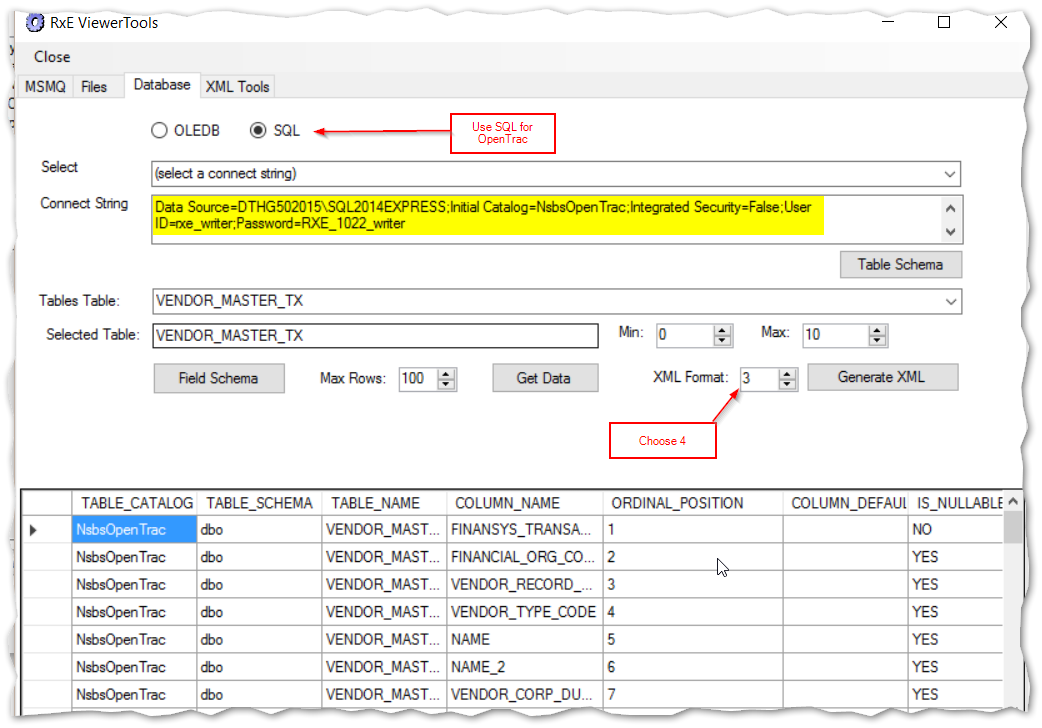
## SPC Simulation

SPC is Sybase, and it uses an OLEDB connection, The simulation environment is SQL Server, but it is a better simulation to employ OLEDB, which is SQLNCLI. It must be installed.



### SPC OLEDB Connect String

Provider=SQLNCLI11;Server=dthg502015\sql2014express;Database=nsbsrxe;user id=rxeconfig;password=rxec0n



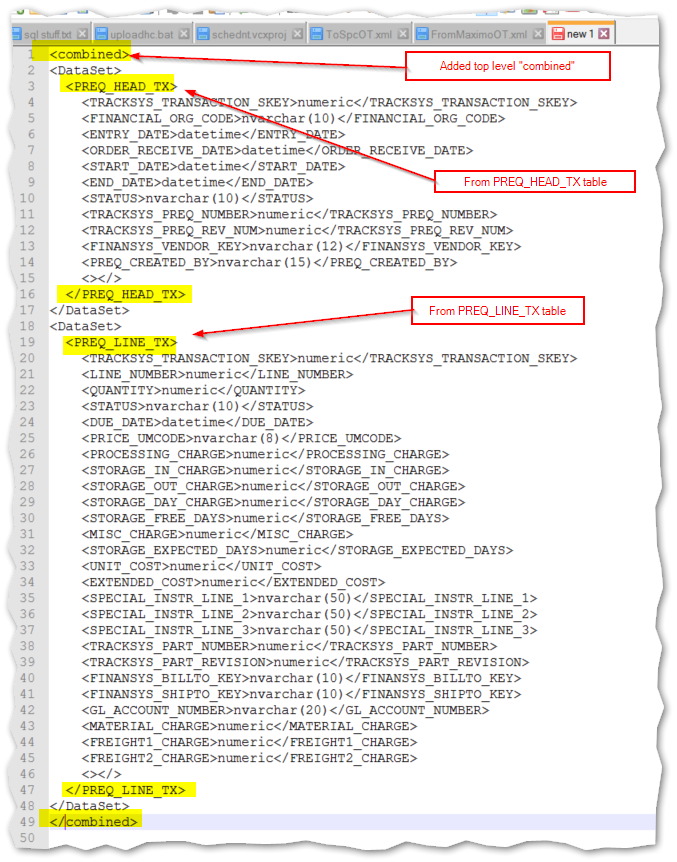
# RxE Development

The RxE system requires “template” messages that can be used for its Pump node Transforms.

The easiest way to produce these for database records is to use RxeViewer and then select “Tools” from the menu.

When the Tools form opens, go to the Database tab and then:

* 1. Select SQL radio button
  2. Insert a connect string (e.g. from “d:\rxe\NSBSTest\(temp)\SqlStuff.txt
  3. Press the “Table Schema” button
  4. Select a table from the dropdown “Select Table”.
  5. Switch the XML Format combo to “4”
  6. Press the “Generate XML” button to put the table into the clipboard
  7. Paste the clipboard content into a text editor. See example results below.



After the message is constructed, it is placed into the folder:

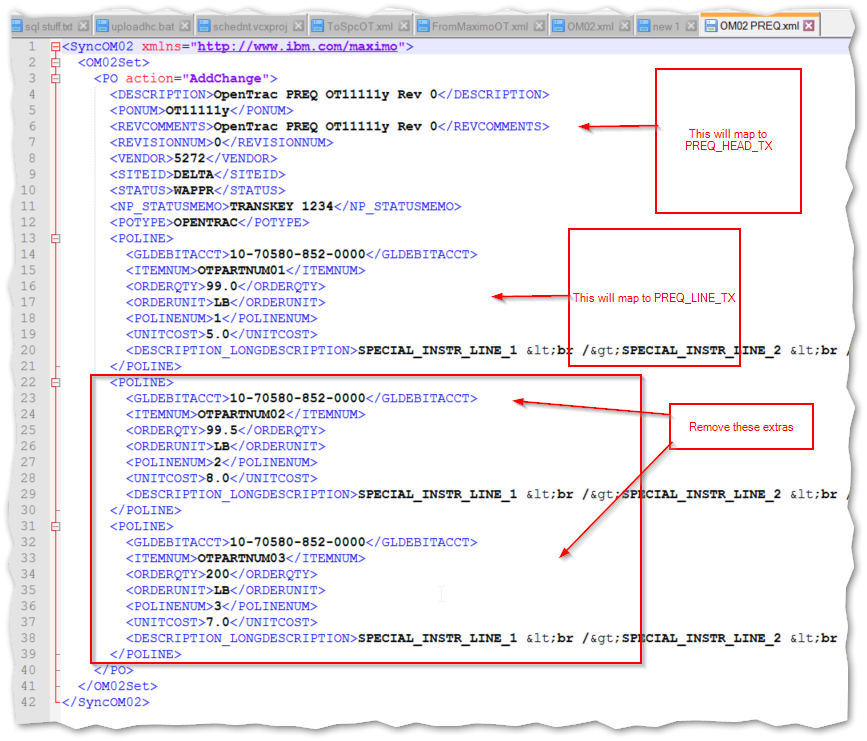
…\configuration\messageTemplates\OpenTrac\FromOpenTrac\OM02.xml

Where it can then be referenced when building a transform.

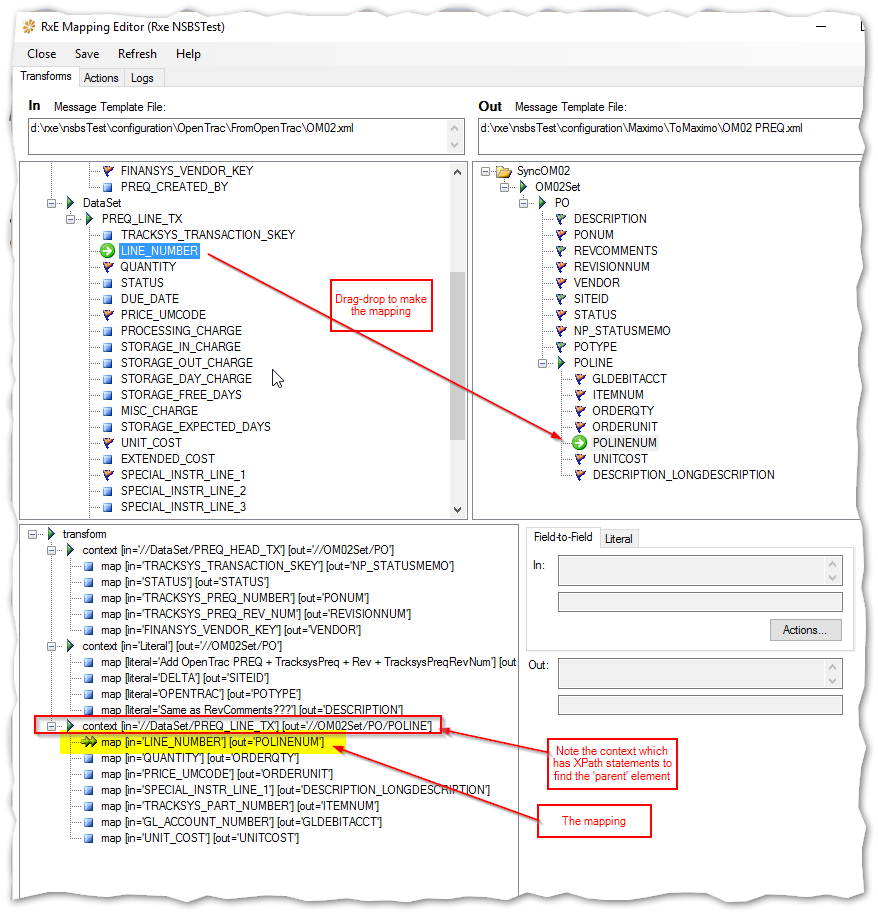
Now consider another type of message: the Maximo messages, which are text files in an XML format.

The example messages provided by Maximo personnel (e.g. Brian) are sufficient unto themselves as the template.

Behold the OM02 message sample, which comprises the OM02 PREQ Master and individual lines. The sample contains three lines, but only one is necessary in the template, as RxE will take the multiples of the PREQ\_LINE records and create new POLINE elements (and their children) as necessary.



Once you have the message templates for the ‘in’ and ‘out’, the RxE Editor can be used to visually make the mappings.

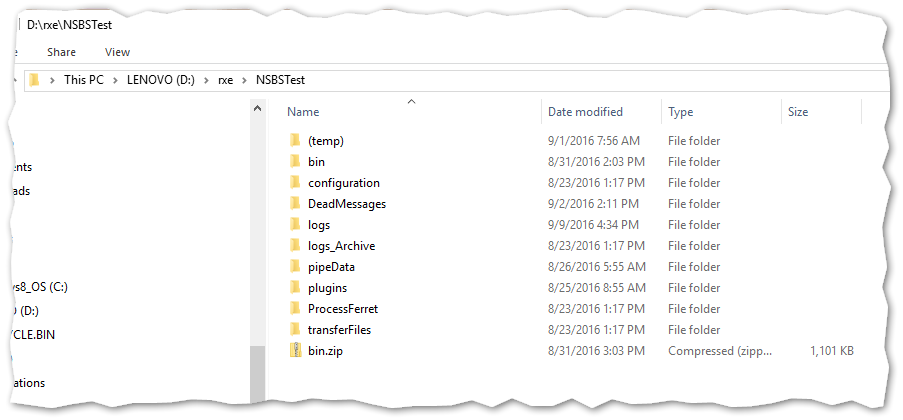


When a pump node is given this transform, it will automatically move the contents from the ‘in’ message to the ‘out’ message before passing the message on to the next pipe.

If more elaborate logic is needed, then it can be added programmatically in the CustomLogic DLL.

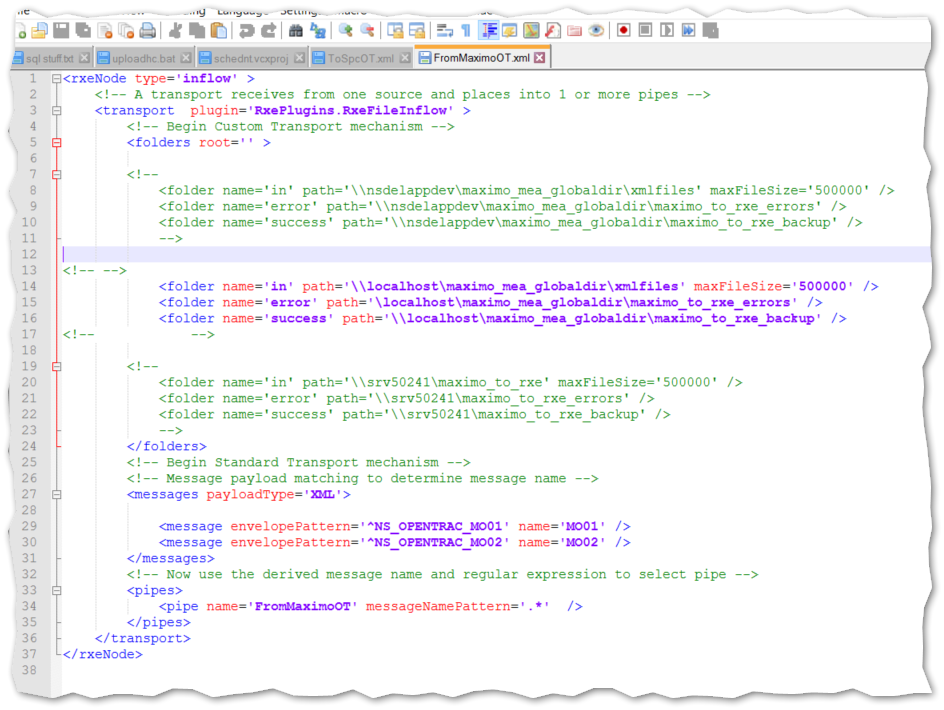
## RxE Test System

The test environment is on D:\NsbsTest



## Maximo Test Environment

Maximo requires file shares for dropping the XML files.



The share for this is on D:, under NsbsTestData

