Email Listener to update Work Order actual start and finish dates

NOTE: This information is provided as-is. The code authored for this scenario was tested with Maximo / Tpae 7.5.0.1 using MAXDEMO database.

Requirements

Adapt Maximo's email listener functionality to process email notifications containing updates to work order fields, specifically, actual start and finish dates.

Assumptions

1. Maximo 7.5 or higher
2. Sender that is composing email will use Maximo's formatted email capability and syntax
3. Email Listener will treat all incoming emails to the particular email account as updates to existing work orders
4. Email Listener will process email content and look specifically for four fields:
    a. WONUM – work order number, primary key on WORKORDER MBO
    b. SITEID – work order site, also primary key on WORKORDER MBO
    c. ACTSTART – actual start date of the work order recorded in the format: 03/06/12 2:00 PM
    d. ACTFINISH – actual finish date of the work order also recorded in the same format

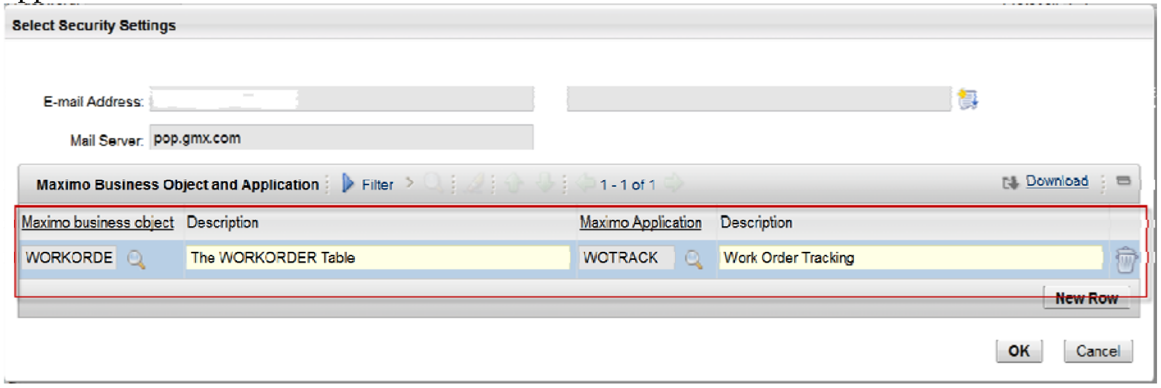
Approach

The following ingredients are assembled together to implement this requirement:

1. Dedicated email listener processing all incoming email from an account as work order updates
2. Dedicated workflow process launching the necessary actions to process work order updates from email
3. Dedicated scripted action launched from workflow process that retrieves the ACTSTART and ACTFINISH dates from the email and updates the correct work order
4. Extend existing Email Listener functionality: We let email listener pull up the email from the mail account and load its content into the INBOUNDCOMM staging table. The Listener parser functionality has already extracted the contents of the formatted email and prepared a simple data structure holding fields and their values. Knowing this is all done, makes our job simple. All we do is retrieve the contents of this data structure and process it to meet our needs.

Email Listener set up

1. I created a test email account and configured an email listener to monitor that email account. I linked this listener to the dedicated workflow process. I set up security authorization for this particular email listener against the Work Order Tracking application of Maximo. The email listener runs under the “MAXADMIN” account and this administrative account has update authorization against the Work Order Tracking application. See screen:



Select Security Settings

E-mail Address:

Mail Server: pop.gmx.com

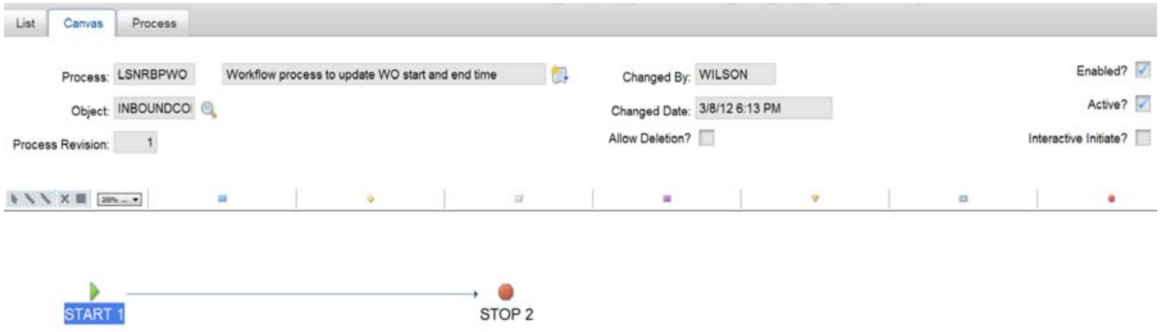
Maximo Business Object and Application Filter 1 - 1 of 1 Download

Maximo business object	Description	Maximo Application	Description
WORKORDE	The WORKORDER Table	WOTRACK	Work Order Tracking

New Row

OK Cancel

2. I called the workflow process LSNRBPWO indicating its use with Work Orders. The workflow process is defined against the INBOUNDCOMM MBO. This MBO manages the staging table for all incoming email. There may be other email accounts being processed by other listeners. All listeners share the same INBOUNDCOMM staging table. The workflow process contains just a Start and Stop nodes and a connecting line. The scripted action on the connecting line is all that’s needed to drive the work order update logic. IMPORTANT: This workflow should be enabled and active to be used in Email Listener. Screen:



List | Canvas | Process

Process: LSNRBPWO Workflow process to update WO start and end time

Object: INBOUNDCO

Process Revision: 1

Changed By: WILSON

Changed Date: 3/8/12 6:13 PM

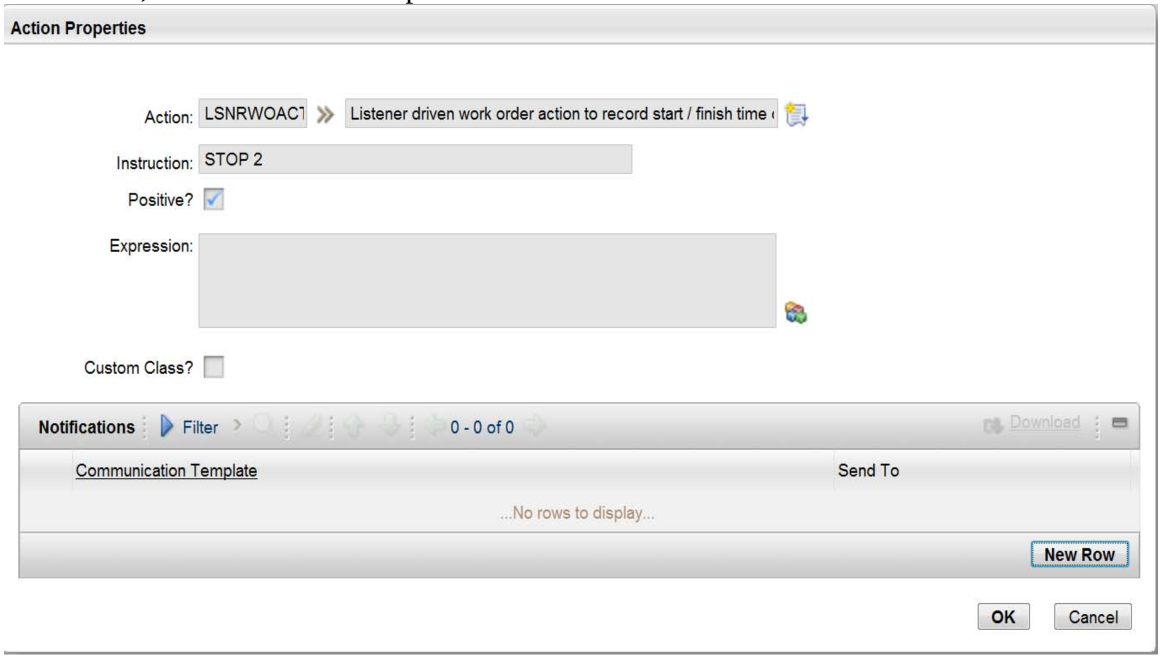
Allow Deletion? ☐

Enabled? ✓

Active? ✓

Interactive Initiate? ☐

3. The action, LSNRWOACT is specified as shown in this screen:


Action Properties

Action:	LSNRWOACT — Listener driven work order action to record start / finish time
Instruction:	STOP 2
Positive?	✓
Expression:	
Custom Class?	

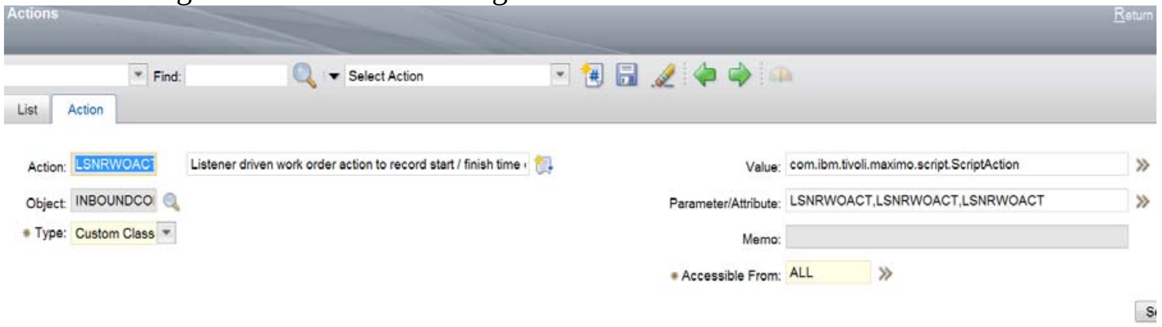
Notifications — Filter — 0 - 0 of 0

Communication Template	Send To
...No rows to display...	

New Row

OK Cancel

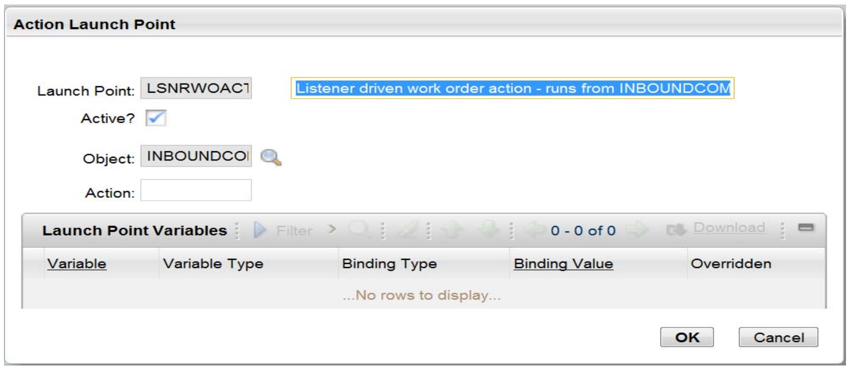
4. To keep things simple, from here on, the name LSNRWOACT is used for the action, the action launch point and the script itself. In the Actions application, no work is required. With Maximo 7.5 Automation Scripting application, defining an Action Launchpoint automatically generates the action and links the action to the action launch point. Here's a screen of the generated action. Nothing should be altered here.



Action:	LSNRWOACT — Listener driven work order action to record start / finish time
Object:	INBOUNDCO
Type:	Custom Class
Value:	com.ibm.tivoli.maximo.script.ScriptAction
Parameter/Attribute:	LSNRWOACT,LSNRWOACT,LSNRWOACT
Memo:	
Accessible From:	ALL

Notice that the Action Type is Custom Class. The class reference is supplied in the Value field. This is a Java class that Maximo supplies out of the box. The Parameter/Attribute field is pre-populated with certain values. These should not be altered otherwise the link with scripting will be lost.

5. Action Launchpoint LSNRWOACT is defined using the Action Launch Point wizard in the Automation Scripts application. The MBO that we want to target is INBOUNDCOMM. No input or output variables are defined currently. Remember, everything in this implementation operates against the INBOUNDCOMM table, not against WORKORDER. The script code ALONE operates against WORKORDER. Screen:



Launch Point: LSNRWOACT Listener driven work order action - runs from INBOUNDCOM
Active? ✓
Object: INBOUNDCOI
Action:

Variable	Variable Type	Binding Type	Binding Value	Overridden
...No rows to display...				

IMPORTANT: The Launch Point Active check box controls whether or not the launch point AND script execute. If unchecked, the script won't run. Ignore the Status field you see on the Automation Scripts application. That is irrelevant and has no effect on Launch Point.

6. Script LSNRWOACT is authored in the multi-line text editor. I have kept this implementation to the bare-minimum code based on the assumptions listed earlier. The code can be more elaborate, generic, factored into functions, etc. But the goal here is to demonstrate how quickly a basic listener can be assembled. Code in its entirety:

```
from java.util import Hashtable
from java.util import Date
from java.text import SimpleDateFormat

print '$$SAM$$ - entering LSNRWOACT action'

# COMMENT: from the INBOUNDCOMM MBO retrieve all of the attributes and
values in the body of the email
attrhashtable = mbo.getAttrHash()
print '$$SAM$$ - total elements in hashtable: ' + str (attrhashtable.size())
dynclause = ''
actualstartdate = attrhashtable.get('ACTSTART')
print '$$SAM$$ - actual start date from email is:' + str(actualstartdate)
actualfinishdate = attrhashtable.get('ACTFINISH')
print '$$SAM$$ - actual finish date from email is:' + str(actualfinishdate)
dynclause = 'WONUM=' + "'" + attrhashtable.get('WONUM') + "' and " +
'SITEID=' + "'" +  attrhashtable.get('SITEID') + "'"
print '$$SAM$$ - where clause is: ' + dynclause
woset = mbo.getMboSet("$lsnrwoset","WORKORDER",dynclause)
# now use the keys to lookup the specific WO record
woset.setQbeExactMatch("true")
woset.reset()
wocount = woset.count()
print '$$SAM$$ - total work order records found is: ' + str(wocount)
if wocount>0:
   wo = woset.getMbo(0)
   if wo is not None:
         sdf = SimpleDateFormat ("MM/dd/yyyy hh:mm aaa")
         date_actstart = sdf.parse(actualstartdate)
         print '$$SAM$$ - formatted date is : ' + date_actstart.toString()
         wo.setValue ("ACTSTART",date_actstart)
         date_actfinish = sdf.parse(actualfinishdate)
         print '$$SAM$$ - formatted date is : ' + date_actfinish.toString()
         wo.setValue ("ACTFINISH",date_actfinish)

print '$$SAM$$ - leaving LSNRWOACT action'
```

Explanation for the code

a. The Email Listener maintains a HASH (data structure) of the attributes and values specified in the body of the formatted email. We take advantage of that in our code. Hence we import Hashtable from the Java world.
b. Our code must generate date / time information and set it into the Work Order object for Maximo to save. Hence we import Date and SimpleDateFormat from the Java world.
c. The first important step in the code is to retrieve the HASH of attributes and values. We do that by calling mbo.getAttrHash(). MBO is a special key word in the scripting word. Placing this keyword in the script allows the script code to work against the current MBO which is the INBOUNDCOMM record we want to process. The getAttrHash() method is public in the Inboundcomm MBO and is used to retrieve the data structure.
d. Once I have this HASH, I can retrieve the desired values from it using Hashtable methods. I retrieve ACTSTART, ACTFINISH, SITEID and WONUM. There may be other values, I ignore them. This listener is geared to processing actual start and finish times. I do need to know which work order should be targeted; hence the work orders primary key information should be supplied in the email. This leads to a set of attrhashtable.get() calls.
e. Now I need to prepare a work order data set to work with. I want to retrieve the right work order. I prepare a SQL where clause using string formatting. Look at the variable dynclause to see what's getting constructed.
f. I now create a new work order data set (MBO set) from the current INBOUNDCOMM MBO by using a standard Maximo Java method called getMboSet(). This method accepts 3 parameters:
    i. Name of the MBO set – a unique name - $lsnrwoset
    ii. Name of the MBO – WORKORDER
    iii. Where clause used to retrieve the desired records – dynclause variable is used
g. I want to make sure just one record matching that SQL where clause will be returned. I want an exact match. Hence I call setQbeExactMatch() method on the work order data set with parameter "true". If this is not done, Maximo executes a wildcard search and returns multiple work orders where work order number contains , say, 1001.
h. I reset the work order data set forcing Maximo to retrieve data from the database.
i. I check the count of returned records. There should be just one.
j. I retrieve that single record into the variable 'wo'.
k. The last few lines of code use the Java SimpleDateFormat class to convert the string value retrieved from the email and set a proper Date object into the work order MBO record represented by 'wo'.

My code is complete. All I need to do is test. To aid in debugging I place a few Jython print statements in the code. These print statements are linked to a LOG LEVEL of DEBUG and once I have the 'autoscript' logger (in Logging application) also set to DEBUG, the output of those statements shows up in the log file. Example:

```
15 Dec 2012 21:23:17:879 [DEBUG] [MXServer] [CID-MXSCRIPT-1300] execution
completed for cached compiled script LSNRWOACT for launch point LSNRWOACT
15 Dec 2012 21:23:17:879 [DEBUG] [MXServer] [CID-MXSCRIPT-1300] $$SAM$$ -
entering LSNRWOACT action
$$SAM$$ - total elements in hashtable: 4
$$SAM$$ - actual start date from email is:03/06/12 2:00 PM
$$SAM$$ - actual finish date from email is:03/06/12 7:35 PM
$$SAM$$ - where clause is: WONUM='1001' and SITEID='BEDFORD'
$$SAM$$ - total work order records found is: 1
$$SAM$$ - formatted date is : Sun Mar 06 14:00:00 EST 12
$$SAM$$ - formatted date is : Sun Mar 06 19:35:00 EST 12
$$SAM$$ - leaving LSNRWOACT action

15 Dec 2012 21:23:17:880 [INFO] [MXServer] [CID-MXSCRIPT-1300] The total time
taken to execute the LSNRWOACT script for the LSNRWOACT  launch point is 57 ms.
```

As can be seen from the log statements, very basic processing is incorporated in the script to retrieve the dates from the email and set them into the right work order record.

I do not need to issue any Save command in the script. The Maximo Action framework handles that automatically. I just need to make sure that the work order data set I am working with is grouped into the same transaction as that used by the INBOUNDCOMM data set. Putting both data sets in the same transaction keeps things simple.

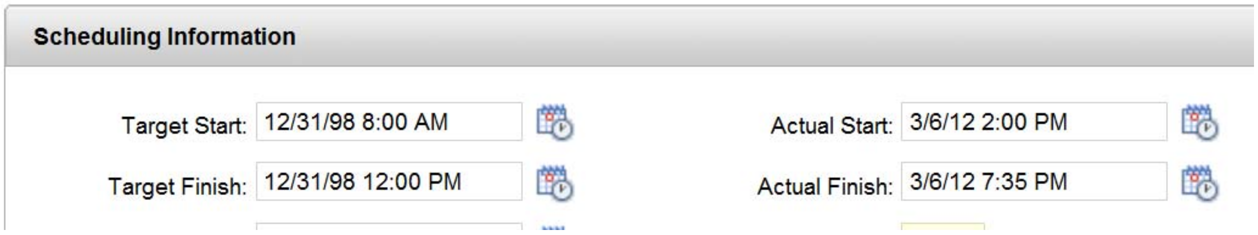
What does the work order email look like?

A simple formatted email to test this code looks like this:

```
#MAXIMO_EMAIL_BEGIN
LSNRACTION=UPDATE
;
LSNRAPPLIESTO=WORKORDER
;
WONUM=1001
;
SITEID=BEDFORD
;
ACTSTART=03/06/12 2:00 PM
;
ACTFINISH=03/06/12 7:35 PM
;
#MAXIMO_EMAIL_END
```

The email should be composed exactly as shown above. The actual start and finish date values as well as the work order number and site ID values can be plugged into the right of the equals sign as desired. Rest of the email should be exactly as shown above.

Once email listener runs, it finds and updates the work order which then displays the values in the application as shown:


Scheduling Information

Target Start:	12/31/98 8:00 AM	Actual Start:	3/6/12 2:00 PM
Target Finish:	12/31/98 12:00 PM	Actual Finish:	3/6/12 7:35 PM

Improvements

1. I did not implement any error handling. Code that’s written to go into an actual implementation would have to be more robust than this.
2. I kept my code simple to operate with just primary keys of work order and the actual start and finish dates. If additional fields should also be processed, more code would have to be added.
3. I did not break down the code into functions. If the complexity of the implementation increases and more code is required, we would have to break it into functions for improved readability.
4. I have set the date pattern as ‘`MM/dd/yyyy hh:mm aaa`’. You can check any number of Java web sites to determine if there’s better pattern you can work with. I have found Maximo to be sensitive to date and time strings. I picked something that I found Maximo can accept. Another option is not to hold the date pattern inside the code. You could declare an IN variable for the Action launch point and bind that variable to the literal date pattern. This way, if you want to play with different date patterns you simply change the launch point bind values without actually changing the code.