Contents

INTRODUCTION **2**

REPORT EXECUTION RE-CAP **2**

SCHEDULED REPORT EXECUTION **4**

SCRIPTING REPORT EXECUTION **4**

DESIGN 5
RECOGNIZING A PURCHASE ORDER STATUS CHANGE 5
CAPTURING THE INPUT PARAMETERS REQUIRED TO SCHEDULE REPORT EXECUTION 5
SCRIPT CODE 7
Imports *7*
Implementation Logic *7*
TESTING 9
LOGGING 10
THINGS TO REMEMBER 10

SUMMARY **11**

RESOURCES **11**

Introduction

Enterprise and on-demand reports are a cornerstone of any Maximo and SmartCloud Control Desk implementation. A number of options are available in the underlying Tivoli process automation engine (Tpae) 7.5 release to view, print and distribute reports. Report execution which precedes the actual viewing, printing and distribution is usually triggered from the particular business application being accessed by user. Report execution is initiated from the business application's user interface, typically, by running the "Run Reports" action available in the Select Action menu.

A customer need that I've often heard of is to automatically execute the report without having to manually initiate it through the "Run Reports" action and setting the various report execution option. For example, can I automatically run the report when the record is being saved or there is a status change?

The Work Order Tracking application has a feature like this where a report can be executed upon the change in status of a particular work order. The Change Status window offers a "Print Reports" check box checking which triggers report execution and delivers an Adobe PDF document to the user's client workstation as downloadable file. Adding this capability to other business applications requires enhancing the application's user interface code.

Are there any other options available in Tpae to automate report execution? What would be the approach taken to implement such automation?

This article describes an Automation Scripts approach to implementing the execution of reports. The goal here is automation of report execution without having to alter the business application or its user interface in any way. I have chosen the Purchase Order (PO) application to implement this scenario with.

NOTE: The Jython-based script code is merely an illustration of report execution may be triggered by means other than existing user-interface-based capabilities. It is not intended to be production-quality code nor is it supported by IBM in any form.

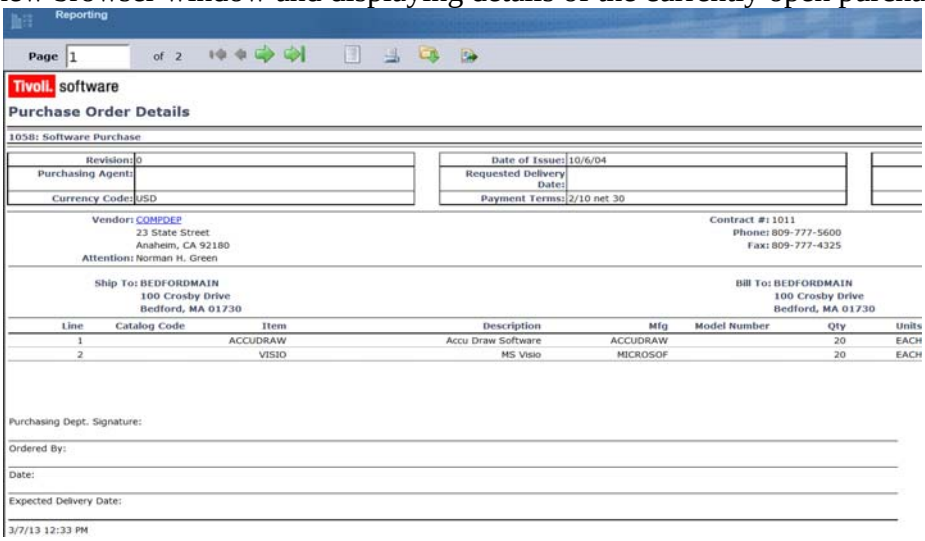
Report Execution re-cap

The Purchase Order application is supported by several reports that ship with the product. These reports include Purchase Order Details and PO Status Details. I'll be using the Purchase Order Details report as the report to be executed for this scenario.

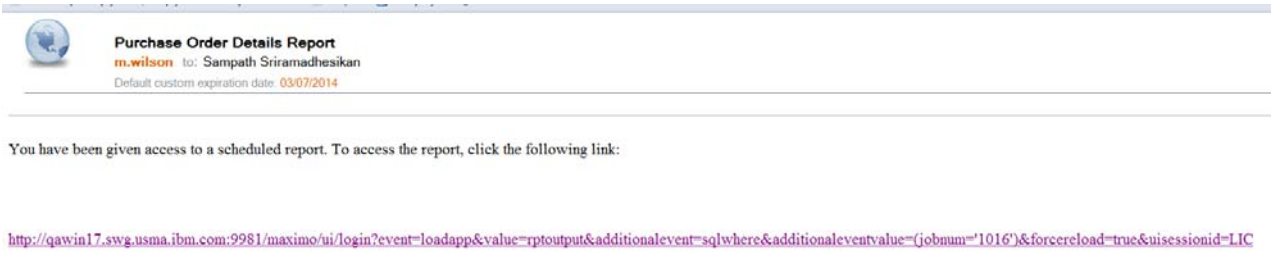
Upon selecting a particular purchase order from the List tab, the typical user initiated report execution tasks are shown in the table on page 3.

Task	Description
1	If Run Reports is available to the user, click this action
2	In the Reports selection window that appears, click the Purchase Order Details Report
3	In the Request Page window that appears, chose or specify the following: a. Schedule (Immediate, At This Time, Recurring) b. Email (Recipients, Subject, Comments, File Type, Report Delivery Format)
4	On the Request Page click Submit
5	If report execution was scheduled to run at a later time, review and close Schedule Confirmation message

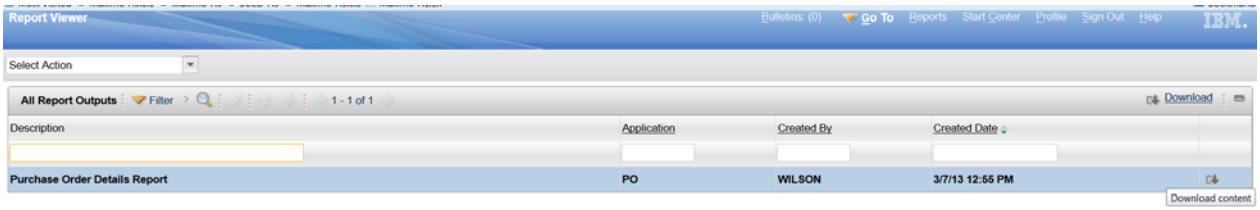
If the Immediate option was chosen, the report runs immediately, bringing up the BIRT Report Viewer in a new browser window and displaying details of the currently open purchase order.



If the Schedule | At This Time option was chosen, then E-mail details such as recipient should be specified and the report is executed in the background. Depending upon the Report Delivery Format chosen, a notification is sent to the recipient either including the report as PDF or Excel spreadsheet or as a live URL that can be clicked by the recipient.



Clicking the URL brings the user to a product's login page and upon completing login the Report Viewer is displayed. The Report Viewer shows a single entry representing the just executed report. A Download Content link to the right of the entry enables the user to download the report as PDF or XLS.



A couple of important points are to be noted:

1. The recipient of the report must have an active person record in the product.
2. If the report is being scheduled for execution at a later time, the time slot chosen must be in the future.

Scheduled report execution

There are two report execution options that involve scheduling:

- Running a report once in the future ("At this Time" selection)
- Running a report on a recurring basis in the future ("Recurring" selection)

Once a report is scheduled, there are two configuration records that drive report execution:

Report schedule record	The report schedule record is used by the reporting framework to hold the scheduling details for a report
Cron task instance record	The cron task instance record for the REPORTSCHEDULE cron task is used by the Cron Task manager to execute the report on a schedule by placing a job into the report queue. The REPORTSCHEDULE cron task looks up the report schedule record to obtain details of the report execution.

When the selection is "At this Time", meaning, one time execution, the reporting framework prepares a one-time cron task instance with the REPORTSCHEDULE cron task. Once the report has been executed, this one-time instance record is removed from the Cron Task instance table.

When the selection is "Recurring", meaning, multiple execution, the reporting framework prepares a cron task instance with the REPORTSCHEDULE cron task. This cron task instance remains in the product environment until an administrator deletes the report schedule record using the View Scheduled Reports user interface in Report Administration application.

NOTE: The REPORTSCHEDULE cron task is a read-only cron task. To view it in the Cron Task Setup application, you must set the Access Level field to READONLY and search.

Scripting report execution

Automation scripting in Tpae 7.5 is restricted to the business object framework executing on the application server. Scripting does not support any facet of the user interface framework except posting error and warning messages. These latter are also managed from the business object framework.

The immediate execution of a report and the display of the BIRT Report Viewer window is initiated and controlled in the user interface layer of the product and involves URL re-direction and servlet use. This capability cannot be scripted. What can be scripted, instead, are the scheduled execution of reports and the sending of report notification to intended recipients. What can also be scripted is the preparation of the report schedule configuration in response to a status change in the business application. This article describes how this can be implemented.

For the report execution scenario I want to automate, I will mimic the "At this Time" feature of scheduled report execution.

Design

The low-level design of this script does the following:

- Check status change
- Create report schedule record
- Create cron task instance record
- Create cron task parameters (copying parameters from the report's parameter records)
- Save

Recognizing a Purchase Order status change

We need to trigger report execution when the status of a PO changes to CLOSE. We will use an Object Launchpoint to implement this check as well as well as schedule the report execution. The scripting framework does not offer an event point directly against the status change event. So we'll configure the launch point to run on an update event of the PO.

Capturing the input parameters required to schedule report execution

Since this script automates report execution (no user interface), but the report scheduling and cron task must prepared with appropriate parameters, we exploit the variables functionality of automation scripting. Since my goal is to ensure report execution, I do not require any OUT variables to be set back into the PO record. I would have liked to post a message to the user indicating that the report has been successfully scheduled for execution, but the automation scripting framework does not have the requisite support to show such a message in the navigation bar of the PO application.

Input variables are listed in the table below:

Variable	Type	Binding	Value	Remarks
v_appname	IN	LITERAL	PO	The name of the application against which the report will be executed; the value is used to generate the file name for the report as well as locate the report from the available report folders. This will be set into the REPORTSCHED record.
v_emailcomments	IN	LITERAL	Ad Hoc Report has been made available to you...	Text that is added to the message body of the email notification sent to recipients upon report execution. This will be set into the REPORTSCHED record.
v_emailfiletype	IN	LITERAL	PDF	Specifies the type of the report output file that is attached to the email notification if user has chosen to attach the report. Valid values are 'PDF' or 'XLS'. This will be set into the REPORTSCHED record.
v_emailsubject	IN	LITERAL	Ad Hoc PO Report via script	Text that is set as the subject line of the email notification sent to recipients upon report execution. This will be set into the REPORTSCHED record.
v_emailto	IN	LITERAL	<Comma-separated email addresses> NOTE: Each email address must correspond to a PERSON record in the product	Specifies a comma-separated list of email addresses to whom the email notification is sent. Each email address must correspond to a valid person record in the product. This will be set into the REPORTSCHED record.
v_emailtype	IN	LITERAL	url	Specifies the type of the email notification. Valid values are 'url' or 'attach'. This will be set into the REPORTSCHED record.
v_maximourl	IN	LITERAL	<Base URL to access the product – usually http://myhost:myport/maximo>	Specifies the product base URL that will be used to generate a complete URL to access the Report Viewer application. This will be set into the REPORTSCHED record.
v_paramdelimiter	IN	LITERAL	||	Specifies the delimiter used by the reporting framework to identify internal parameters to run the report with. These are different than the standard report parameters associated with the report design.

Variable	Type	Binding	Value	Remarks
v_paramstring	IN	LITERAL	appHierarchy=PO||displayWhere =	Required report framework internal parameters: appHierarchy should be set to the application name from where the report execution is being scheduled; displayWhere is a dynamic where clause – this script does not use displayWhere as it uses the currently open PO's record identifier
v_status	IN	ATTRIBUTE	STATUS	The status of the PO
v_reportname	IN	LITERAL	poprint.rptdesign	The name of the report to be executed – this will be set into the REPORTSCHED record

From this list of input variables, only the v_status variable is bound to the PO business object's STATUS attribute. All other variables serve merely to specify the necessary input parameters to the report scheduling APIs.

Script code

The script code is available as a file with this article. I will walk through the code to explain the implementation logic. The code predominantly relies upon Java-based APIs.

Imports

Several Java package imports placed at the beginning of the script are required to invoke APIs.

```
1  from java.util import Calendar
2  from java.util import Date
3  from psdi.app.report import ReportUtil
4  from psdi.server import MXServer
5  from psdi.mbo import MboConstants
6  from psdi.mbo import SqlFormat
```

The Calendar and Date APIs are used to specify the point in time when the REPORTSCHEDULE cron task should execute the report. The ReportUtil methods are utilized to convert the Date object into a string representation of the time when the cron task should execute the report. The MXServer methods are used to retrieve report schedule, report parameters, crontask definition, crontask instance and crontask parameter data sets (MBOSets). A record each will be created of the report schedule, cron task instance and cron task parameters. The MBOConstants class is used to specify field flags when setting attribute values on a record (MBO). The SqlFormat class is used to set up SQL criteria to retrieve the report parameters for the chosen report.

Implementation Logic

Lines 15 and 16 of the script code check the status change of the PO and verify that the status is being set to 'CLOSE'.

```
15 if (str(v_status) != str(v_status_initial)) and v_status=='CLOSE':
16         print "Status change is happening from " + str(v_status_initial) + " to " + str (v_status)
```

Lines 18, 20 and 21 set up a Java Date object that's initialized to 30 seconds from the current system time. This Date object is passed to the ReportUtil's convertOnceToSchedule() method.

This method accepts three parameters: Date object, Locale object and TimeZone object. It returns a string that represents the cron schedule accepted by Tpae's cron task engine. On lines 24 and 25, a Locale object and Tpae UserInfo object are retrieved as they will be utilized later on in the script code. Notice that these objects are retrieved from the current MBOSet that the PO record is part of.

```
17          # set a schedule for the report
18          c = Calendar.getInstance()
19          # add 60 seconds to current time to allow preparing REPORTSCHED cron task instance
20          c.add(Calendar.SECOND,30)
21          d = c.getTime()
22          thisposet = mbo.getThisMboSet()
23          if thisposet is not None:
24                  locale = thisposet.getClientLocale()
25                  userinfo = thisposet.getUserInfo()
26          schedule = ReportUtil.convertOnceToSchedule(d,locale,c.getTimeZone())
27          print "Schedule we have to set into REPORTSCHED Cron task is: " + str(schedule)
```

Lines 32 through 47 add a new record to the REPORTSCHED table. All the required attributes of the REPORTSCHED record are set. Some of the attributes are not required but nevertheless set: COUNTRY and VARIANT to ensure locale-sensitive handling; EMAILCOMMENTS and EMAILSUBJECT enabling configurability of the email being sent to recipients; MAXIMOURL so that a link to the executed report can be included in the email.

```
28          if schedule is not None:
29                  reportschedset = MXServer.getMXServer().getMboSet("REPORTSCHED",userinfo)
30                  if reportschedset is not None:
31                          print "Obtained REPORTSCHED set"
32                          reportsched = reportschedset.add()
33                          reportsched.setValue("REPORTNAME",v_reportname)
34                          reportsched.setValue ("appname",app)
35                          reportsched.setValue ("USERID",userinfo.getUserName())
36                          reportsched.setValue ("TYPE","once")
37                          reportsched.setValue("EMAILTYPE",v_emailtype)
38                          reportsched.setValue("MAXIMOURL",v_maximourl)
39                          reportsched.setValue("EMAILUSERS",v_emailto)
40                          reportsched.setValue("EMAILSUBJECT",v_emailsubject)
41                          reportsched.setValue("EMAILCOMMENTS",v_emailcomments)
42                          reportsched.setValue("EMAILFILETYPE",v_emailfiletype)
43                          reportsched.setValue("COUNTRY",locale.getCountry())
44                          reportsched.setValue("LANGUAGE",locale.getLanguage())
45                          reportsched.setValue("VARIANT",locale.getVariant())
46                          reportsched.setValue("TIMEZONE",thisposet.getClientTimeZone().getID())
47                          reportsched.setValue("LANGCODE",userinfo.getLangCode())
```

The code could be made more robust by checking that input parameters to the script, represented by v_ naming convention are not null.

Also notice that some of the values being set into the REPORTSCHED record are conveniently retrieved from the script framework itself ("app" implicit variable), or previously retrieved objects (userinfo, locale).

Having prepared the REPORTSCHED record, we now need to create a CRONTASKINSTANCE record against the REPORTSCHEDULE cron task. The code is between lines 51 and 64. When setting values into the CRONTASKINSTANCE record, the instance name is set to a numeric value that's a function of the current date and time (line 59). The RUNASUSERID is set to the currently logged in user (line 62). The AUTOREMOVAL flag is set to "True" indicating that this cron instance should be automatically deleted after report execution. Lines 66 and 67 retrieve Cron Task name and instance and link back to the REPORTSCHED record so the reporting framework will find the scheduling details for the cron when it begins execution.

```
48
49          print "About to work with REPORTSCHEDULE cron task"
50          crontaskdef = reportsched.getMboSet("$parent","crontaskdef","crontaskname='REPORTSCHEDULE'").getMbo(0)
51          if crontaskdef is not None:
52                  crontaskinstset = crontaskdef.getMboSet("CRONTASKINSTANCE")
53                  if crontaskinstset is not None:
54                          print "About to work with Cron task instance of REPORTSCHEDULE cron task"
55                          crontaskinst = crontaskinstset.add(MboConstants.NOACCESSCHECK | MboConstants.NOVALIDATION_AND_NOACTION)
56                          if crontaskinst is not None:
57                                  d = Date()
58                                  crontaskinstname = str(d.getTime())
59                                  crontaskinst.setValue("CRONTASKNAME","REPORTSCHEDULE",MboConstants.NOACCESSCHECK | MboConstants.NOVALIDATION_AND_NOACTION)
60                                  crontaskinst.setValue("INSTANCENAME",crontaskinstname,MboConstants.NOACCESSCHECK | MboConstants.NOVALIDATION_AND_NOACTION)
61                                  crontaskinst.setValue("SCHEDULE",schedule,MboConstants.NOACCESSCHECK | MboConstants.NOVALIDATION_AND_NOACTION)
62                                  crontaskinst.setValue("ACTIVE",1,MboConstants.NOACCESSCHECK | MboConstants.NOVALIDATION_AND_NOACTION)
63                                  crontaskinst.setValue("RUNASUSERID",userinfo.getUserName(),MboConstants.NOACCESSCHECK | MboConstants.NOVALIDATION_AND_NOACTION)
64                                  crontaskinst.setValue("HASLD",0,MboConstants.NOACCESSCHECK | MboConstants.NOVALIDATION_AND_NOACTION)
65                                  crontaskinst.setValue("AUTOREMOVAL",True,MboConstants.NOACCESSCHECK | MboConstants.NOVALIDATION_AND_NOACTION)
66                                  print "have set all cron task instance values for REPORTSCHEDULE cron task"
67          reportsched.setValue("CRONTASKNAME",crontaskinst.getString("CRONTASKNAME"))
            reportsched.setValue("INSTANCENAME",crontaskinst.getString("INSTANCENAME"))
```

The final chunk of code is between lines 68 and 105. This chunk of code prepared the parameters for the cron task instance. To set up those cron task parameters, the code must retrieve report parameters for the chosen report from the REPORTPARAM table(lines 71 through 79). Those parameters are retrieved into the 'reportparamset' object. The code then iterates through these parameters and sets them into a new CRONTASKPARAM record. Lines 90 through 95 retrieve the unique ID (identifier) column name and value from the current PO record so that the report will be executed against that particular PO record. This is the only criteria set against the PO.

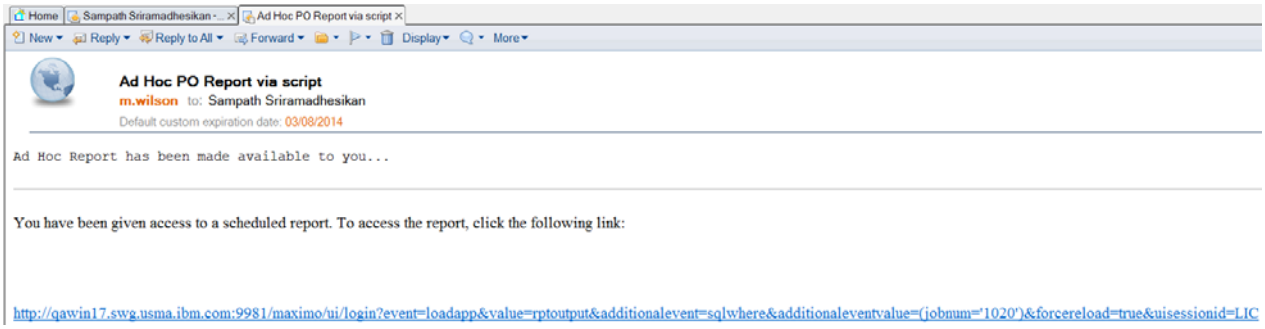
Finally, on line 106, the 'reportschedset' object is saved which results in the REPORTSCHED, CRONTASKINSTANCE and CRONTASKPARAM records being saved.

When this script is saved, the launch point is active.

Testing

Testing this script is straightforward. I tested this report against the Purchase Order Details (poprint.rptdesign) report that ships with the product and an Ad Hoc report that I created myself.

1. Create a Purchase Order. Add lines. Save.
2. Take the newly created Purchase Order through typical status changes.
3. Close the Purchase Order – look for the navigation bar confirmation message that the PO was set to status CLOSE.
4. Quickly access the REPORTSCHEDULE cron task in the Cron Task Setup application (remember to set the Access Level to READONLY to find this cron task).
5. You should see a new cron task instance associated and cron task params that show the unique ID of the PO that you just closed.
6. Wait for a short duration of time, perhaps a couple of minutes to allow the cron task to execute the report.
7. If you have configured a mail client to receive the email notification from Maximo, you should see a new email in the Inbox of the client that looks similar to this:



8. Clicking the link will bring you back into the product environment (with a re-direction to log in, if required). The Report Viewer application shows one entry that corresponds to the Purchase Order you just closed. You can download the PDF or XLS using the Download Content link to the right of the entry.

Logging

As with any automation script, testing in a development environment is best practice before the script and launch point are promoted to upper environments. I placed a few print statements in the body of the script to help me trace execution. Combined with the DEBUG log level of the scripting framework, I can get useful execution information from the product log files:

```
08 Mar 2013 16:55:17:141 [DEBUG] [MXServer7503] [CID-MXSCRIPT-19335] execution completed for cached compiled script POREPORT for launch point POREPORT
08 Mar 2013 16:55:17:142 [DEBUG] [MXServer7503] [CID-MXSCRIPT-19335] PO status old value: APPR
PO status new value: CLOSE
Status change is happening from APPR to CLOSE
Schedule we have to set into REPORTSCHED Cron task is: 1y,46,55,16,8,2,*,*,*,*
Obtained REPORTSCHED set
About to work with REPORTSCHEDULE cron task
About to work with Cron task instance of REPORTSCHEDULE cron task
have set all cron task instance values for REPORTSCHEDULE cron task
Now going to work with Cron task PARAMETERS
working with REPORTPARAM mbo set
going to copy values from REPORTPARAM into CRONTASKPARAM
going to copy values from REPORTPARAM into CRONTASKPARAM
If condition for v_paramdelimiter
going to copy values from REPORTPARAM into CRONTASKPARAM
If condition for v_paramstring
going to copy values from REPORTPARAM into CRONTASKPARAM
going to copy values from REPORTPARAM into CRONTASKPARAM
going to copy values from REPORTPARAM into CRONTASKPARAM

08 Mar 2013 16:55:17:142 [INFO] [MXServer7503] [CID-MXSCRIPT-19335] The total time taken to execute the POREPORT script for the POREPORT  launch point is 540 ms.
```

My print statements are more reflective of my development effort around the script code and should be improved for traceability.

Things to remember

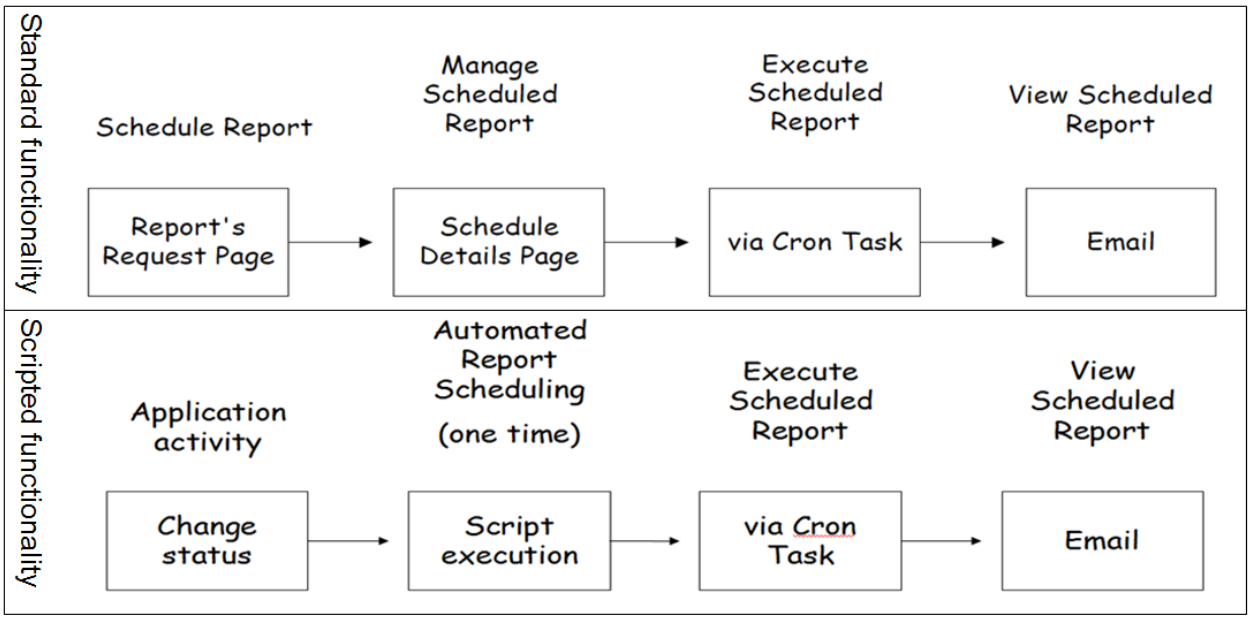
1. If report execution is no longer desired during the close of a PO, simply de-activate the launch point.
2. The launch point for this script code runs against the PO MBO only. If you would like to implement this against a different object, then you must prepare a new launch point against that object.
3. The script code is itself generic enough to run against any main MBO and business application that have reports associated with them. Some clean up is required as fragments of the code refer to “PO” (for example, variable names and print statements).
4. The variable binding values associated with the launch point will also need to change if a different MBO and application are being targeted.
5. The launch point will not run when Admin Mode is in effect (neither will any Cron Task).
6. Depending upon the load on the Cron Task JVM, the actual execution of report can vary by a very small fraction of time. Similarly, email delivery times may vary based on the mail server configurations.

7. Although I have a number of checks for errors in my code, this is, by no means, production quality code. As I stated previously additional checks should be added to deal with null and other unexpected values.
8. In the code an assumption is made that the logged in user who is also being set as the REPORTSCHEDULE Cron Task's Run As user has sufficient authorizations to successfully execute the report. If this assumption is not always true, an IN variable could be added to hold the desired Run As user's ID to be set.
9. The launch point and script code are designed to execute the desired report against the current PO record. Other types of reports against the Purchase Order application such as the Vendor Performance report cannot be executed with this script. These reports do not use a particular PO as parameter.

Summary

The approach described in this article illustrates how scripting can be adapted to report execution. Knowledge of the scheduling capabilities of the reporting framework was a pre-requisite to providing an implementation. The solution also illustrates how multiple Tpae product configurations (report design, parameters, cron tasks, cron task instances) are linked together in order to deliver functionality to end users.

I can now contrast the standard scheduled report execution sequence against the scripted report execution sequence:



The key difference is that in the scripted approach, report execution is scheduled automatically in response to an application event.

Resources

Application Report display configurations
http://www.ibm.com/developerworks/wikis/display/maximo/Administering+Reports+-+Application+Report+Display+Configurations

Report features guide

http://www-01.ibm.com/support/docview.wss?uid=swg21498433
(Refer the Report Scheduling section on page 36)

Tpae automation scripting cookbook
http://ibm.co/pPl32E