Scheduling Workbooks through the Application Concurrent Manager

By Rod West, Cabot Consulting

Oracle Application users will be very familiar with the Applications concurrent manager and how to use it to schedule Applications processes. Many users also use Oracle Discoverer to generate reports from their applications data. This paper describes how Oracle Applications Concurrent Processing can be used to schedule Discoverer workbooks. The end user submits requests for a Discoverer report to the concurrent manager and can view the report generated with the request output viewer. Discoverer has a workbook scheduler built into the product but this is an alternative approach to using the Discoverer scheduling. This approach uses the Discoverer export functionality to generate reports in files that can be downloaded by the Applications users.

The Benefits of this Approach

We already know that scheduling Discoverer workbooks automates the production of reports and allows end users to fully utilise the capacity of the system by processing long running reports during off-peak hours.

Using the concurrent manager to schedule workbooks brings additional benefits to the Application User:

1. The standard submit request form is used to enter requests for Discoverer reports and all the features available in the concurrent manager are available to the end user. These features include work shifts, periodic scheduling, request sets and request groups. In particular, a request can be included as part of a request set which ensures that dependent Application jobs are completed before the Discoverer report is generated.
2. The reports generated are stored in files that can be viewed as output from the concurrent request by many Application users. This paper describes how Discoverer workbooks can be used to select and download reports that have been scheduled. This allows reports to be scheduled by an end user and shared with a responsibility. End users can then view a report that has been requested and generated by another user.
3. A Discoverer schedule request submitted through the concurrent manager can be used to submit further requests using a custom End User Layer (EUL) function described in this paper. This allows a single request to be used, for example, to generate a report for each organization, or, for example, to generate a report for each employee who joined during the current month.

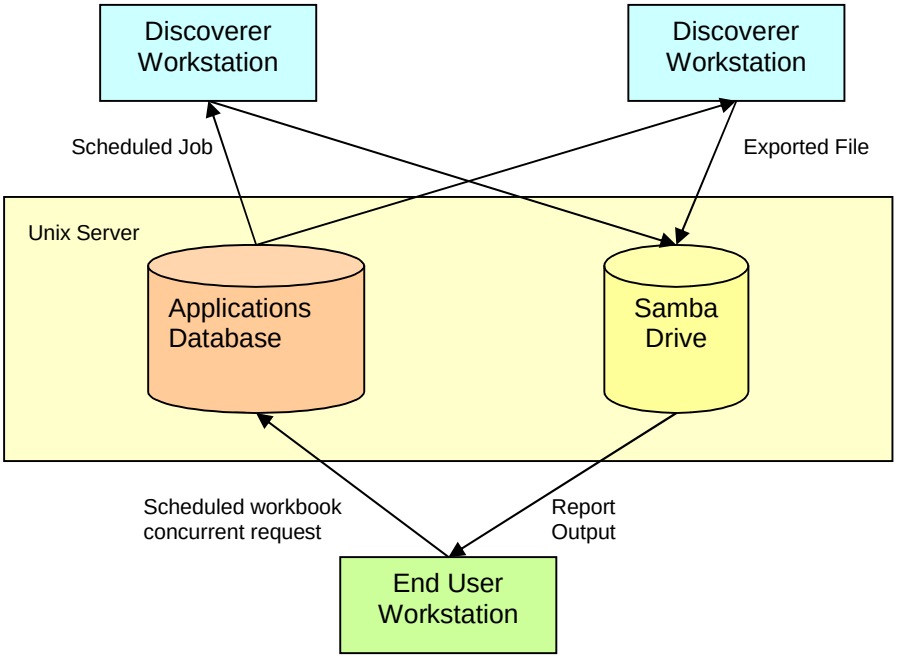
Infrastructure Requirements

This solution requires that the Discoverer reports are produced in files and we must use the Discoverer User Edition running on a Windows workstation to do this. The Discoverer User Edition can be used to generate output in HTML or Excel files from the command line. The Discoverer User edition can run on Windows NT/2000/2003/XP workstations and uses TNS to connect to the database server.

The design allows many Discoverer exports to be processed simultaneously on one or more Discoverer workstations. These workstations can be dedicated to processing Discoverer scheduled workbooks providing a secure mechanism for generating Discoverer reports. Your users don't need to connect to the Discoverer workstations; all the processing is completely automatic, controlled from the database server.

The Applications Concurrent Manager retrieves request output from the Unix server and in this architecture, the Unix server has a Samba drive which is shared with the Discoverer workstations so that the Discoverer User Edition can write output directly onto the Unix server.

Figure 1 – Discoverer Scheduling Components



Submitting a workbook request to the concurrent manager

The end users submit requests to process a workbook using the standard applications submit request form and a custom concurrent program. All the features available to standard concurrent requests can also be applied to processing Discoverer workbooks. These features include:

- Request sets - Requests can be included as part of a request set so that a report can be generated after an application process has completed.
- Scheduling – Requests can be run repeatedly, e.g. every weekday evening.
- Work shifts – Requests can be submitted can be queued and processed during a work shift.
- Request groups – Privileges to submit requests is controlled through their responsibility.

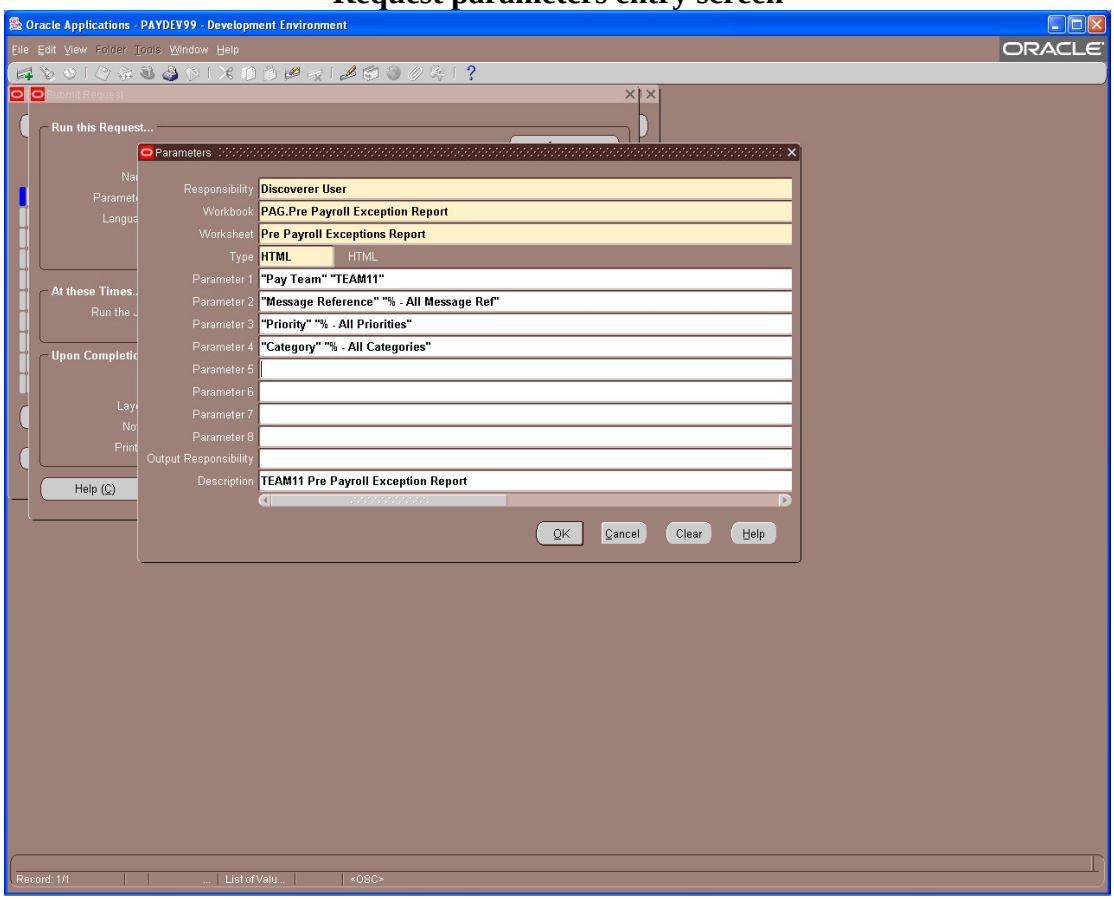
The end users enter the details of the workbook being scheduled into the parameters of the request. The schedule workbook request program has the following parameters:

- Responsibility – This is the responsibility that is used to process the workbook. The responsibility can be different from the current responsibility, but this responsibility must be assigned to the user who submits the request.
- Workbook – This can be any workbook in the database that the user can access that is shared with the responsibility. The end user selects the workbook from a valueset that lists all the workbooks in the EUL that the user can schedule.
- Worksheet – The name of the worksheet in the workbook to be processed.
- Type – The type of export required. Discoverer supports CSV, HTML and Excel exports.
- Parameter 1 to 8 – Theses are the parameters to be applied when the workbook is exported. In this solution up to 8 parameters can be specified for the export. Each parameter for the Discoverer workbook is entered as a separate parameter into the request. There are some limitations on the parameters that can be used, for example, if the request is scheduled to run periodically then each time the workbook is processed the parameters applied are the same. But more about parameters later.
- Output Responsibility –Optionally an alternative responsibility can be specified. This allows the user entering the request to share the output of the report with other users. An end user with this responsibility can view the output of the report but does not necessarily have to have the privileges to run the report in Discoverer.
- Description – This is an optional description or title that will be given to the output generated by this request.

An example showing the entry of a Discoverer scheduling request into the standard concurrent request form is shown below. In this example, the Discoverer workbook requires 4 parameters. The name of the parameter and the value for the parameter is entered into a single request parameter. If the name or value of the parameter contains spaces then the parameter name or value must be enclosed in double quotes.

So to summarise, the end user schedules a new workbook by creating and saving the workbook in the EUL, sharing the workbook with a responsibility and then submitting a request giving the details of the workbook to be processed.

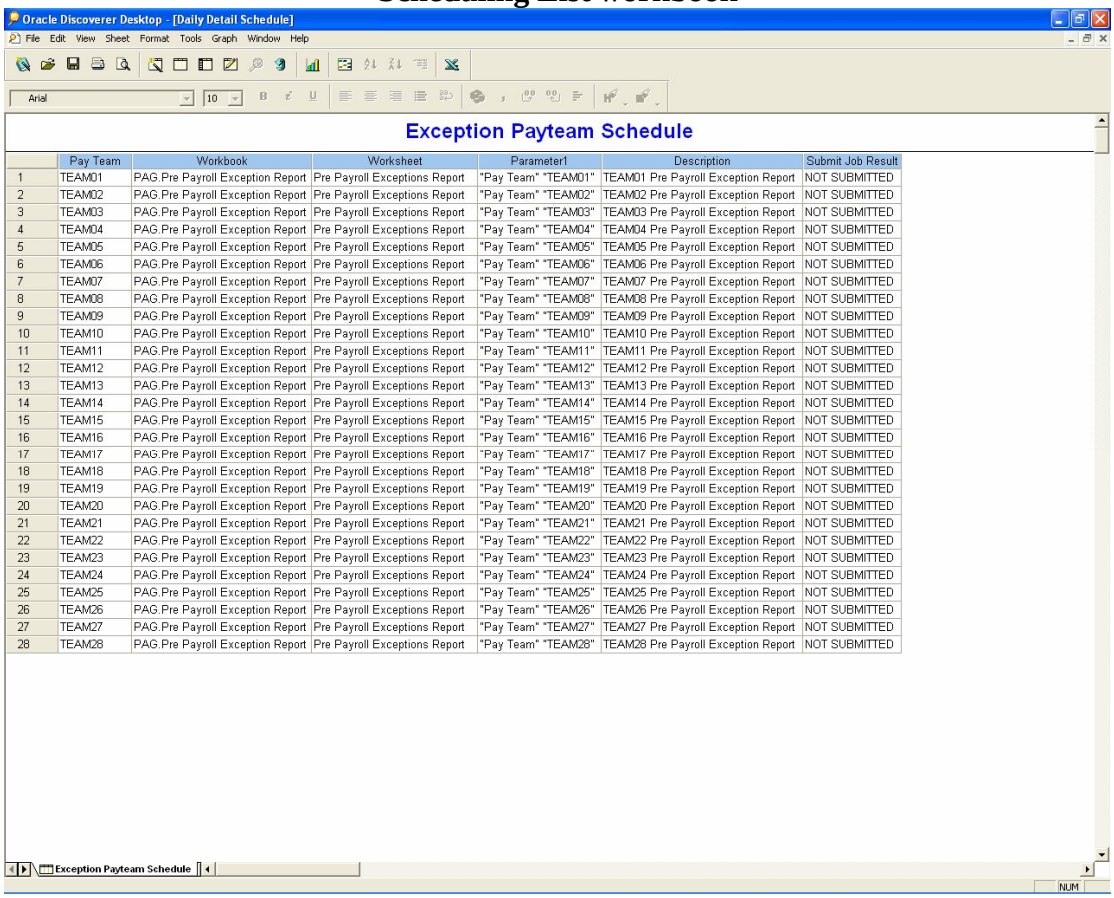
Request parameters entry screen



Frequently, the user will want to process a report with different combinations of parameters, for example, to run a report once for each organisation or customer. Manually submitting multiple requests into the concurrent manager would be impractical, and therefore the solution allows the user to create many requests to process workbooks by using a scheduling list workbook.

An example of a scheduling list workbook is shown below. In our example, the user wishes to generate a report for each pay team. The scheduling list workbook returns a row for each report that the user wishes to generate. There is a column in the workbook for each request parameter and in addition the workbook contains a calculated item calling a custom ‘submit job’ function. This custom function is mapped into the EUL and submits a Discoverer schedule workbook request to the concurrent manager when the workbook is processed.

Scheduling List workbook



	Pay Team	Workbook	Worksheet	Parameter1	Description	Submit Job Result
1	TEAM01	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM01"	TEAM01 Pre Payroll Exception Report	NOT SUBMITTED
2	TEAM02	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM02"	TEAM02 Pre Payroll Exception Report	NOT SUBMITTED
3	TEAM03	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM03"	TEAM03 Pre Payroll Exception Report	NOT SUBMITTED
4	TEAM04	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM04"	TEAM04 Pre Payroll Exception Report	NOT SUBMITTED
5	TEAM05	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM05"	TEAM05 Pre Payroll Exception Report	NOT SUBMITTED
6	TEAM06	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM06"	TEAM06 Pre Payroll Exception Report	NOT SUBMITTED
7	TEAM07	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM07"	TEAM07 Pre Payroll Exception Report	NOT SUBMITTED
8	TEAM08	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM08"	TEAM08 Pre Payroll Exception Report	NOT SUBMITTED
9	TEAM09	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM09"	TEAM09 Pre Payroll Exception Report	NOT SUBMITTED
10	TEAM10	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM10"	TEAM10 Pre Payroll Exception Report	NOT SUBMITTED
11	TEAM11	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM11"	TEAM11 Pre Payroll Exception Report	NOT SUBMITTED
12	TEAM12	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM12"	TEAM12 Pre Payroll Exception Report	NOT SUBMITTED
13	TEAM13	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM13"	TEAM13 Pre Payroll Exception Report	NOT SUBMITTED
14	TEAM14	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM14"	TEAM14 Pre Payroll Exception Report	NOT SUBMITTED
15	TEAM15	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM15"	TEAM15 Pre Payroll Exception Report	NOT SUBMITTED
16	TEAM16	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM16"	TEAM16 Pre Payroll Exception Report	NOT SUBMITTED
17	TEAM17	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM17"	TEAM17 Pre Payroll Exception Report	NOT SUBMITTED
18	TEAM18	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM18"	TEAM18 Pre Payroll Exception Report	NOT SUBMITTED
19	TEAM19	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM19"	TEAM19 Pre Payroll Exception Report	NOT SUBMITTED
20	TEAM20	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM20"	TEAM20 Pre Payroll Exception Report	NOT SUBMITTED
21	TEAM21	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM21"	TEAM21 Pre Payroll Exception Report	NOT SUBMITTED
22	TEAM22	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM22"	TEAM22 Pre Payroll Exception Report	NOT SUBMITTED
23	TEAM23	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM23"	TEAM23 Pre Payroll Exception Report	NOT SUBMITTED
24	TEAM24	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM24"	TEAM24 Pre Payroll Exception Report	NOT SUBMITTED
25	TEAM25	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM25"	TEAM25 Pre Payroll Exception Report	NOT SUBMITTED
26	TEAM26	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM26"	TEAM26 Pre Payroll Exception Report	NOT SUBMITTED
27	TEAM27	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM27"	TEAM27 Pre Payroll Exception Report	NOT SUBMITTED
28	TEAM28	PAG.Pre Payroll Exception Report	Pre Payroll Exceptions Report	"Pay Team" "TEAM28"	TEAM28 Pre Payroll Exception Report	NOT SUBMITTED

In our example the user submits a single request to process the scheduling list workbook. When this workbook is processed on the Discoverer workstation a further set of concurrent requests are created to generate a report for each pay team. These requests are then processed in turn by the concurrent manager creating all the reports required.

The end user can create the scheduling list workbook using an EUL folder that returns a row for each pay team. Most of the items shown in the workbook would be calculated items containing the text for the parameters of the workbook being scheduled.

The use of scheduling list workbook provides a convenient and flexible mechanism for processing workbooks. In our example, if a new pay team was entered into the system the workbook scheduling would not require any change. The new pay team would be included automatically in the scheduling list workbook and hence a new Discoverer report would be generated automatically for the new pay team.

Generating the Report

The Discoverer reports are generated automatically when the concurrent request runs without the need for any further user intervention. The concurrent request does not generate the Discoverer report; it just adds the request to a Discoverer schedule jobs queue.

An export script, running periodically on the Discoverer workstations, checks for jobs on the Discoverer schedule jobs queue and queries the database for the Discoverer export command that is to be run. The Discoverer workstation then runs the Discoverer export connecting to Oracle using a dummy Applications user and the responsibility that was supplied as a parameter in the scheduling request.

The use of separate Discoverer schedule jobs queue (implemented as a table in the database) has a number of benefits:

- Multiple workstations can process requests and generate Discoverer reports. Database locking manages the contention between the workstations.
- The Discoverer export command is built within a database view so that the script on the workstation does not need to be aware of usernames, responsibilities, workbooks and other components of the export command. This greatly reduces the complexity of the export script.
- A simple Discoverer workbook can be used by the end user to monitor the schedule jobs queue and track progress of the Discoverer scheduled jobs.

Retrieving Report Output

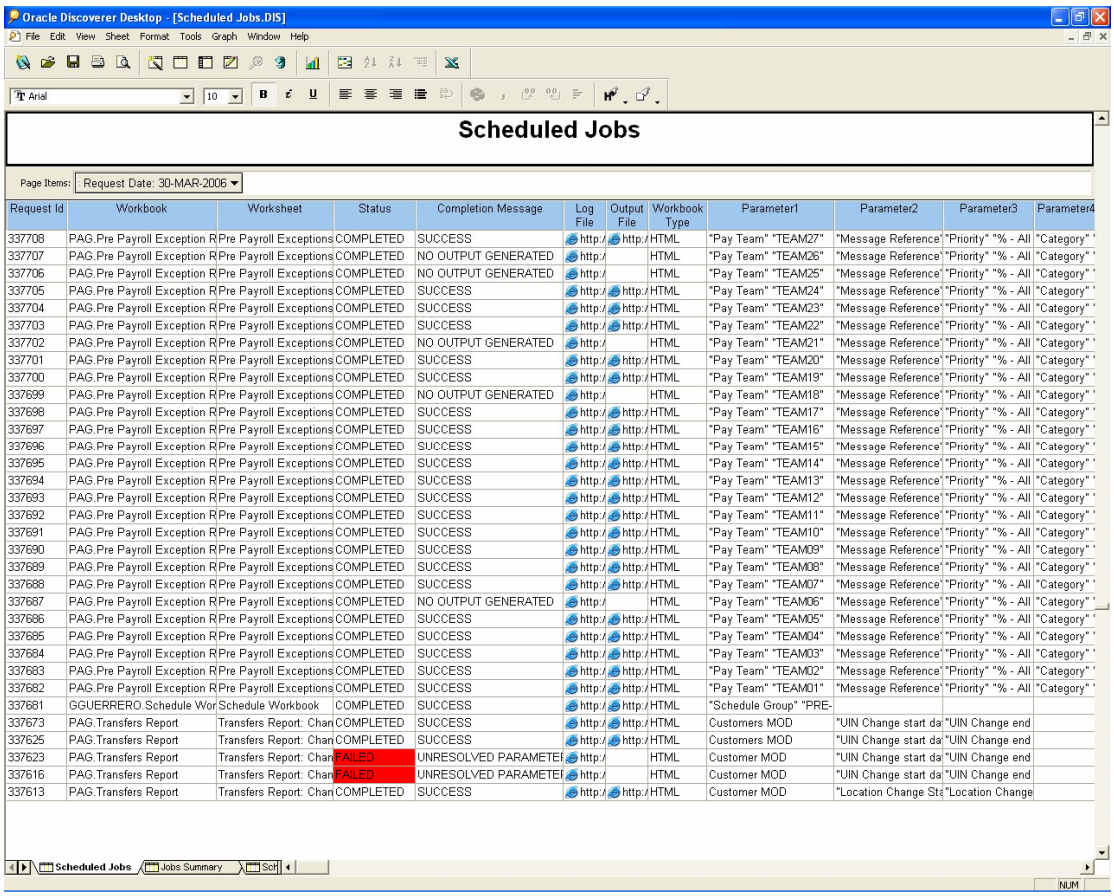
The standard request output viewer can be used by the end user to view reports. The desktop program (usually Excel or Internet Explorer) that is used to view the report can be configured using Application profiles.

However, the standard request form is inflexible and not suitable for distributing reports to many users. The request output can only be retrieved by the user who submitted the request (unless the profile allows any user with the same responsibility to retrieve the output). Also the form does not allow the end user to search for requests based on request parameters and therefore an end user cannot search the concurrent requests for a request for a specific workbook.

We can overcome all these restrictions by developing a Discoverer workbook that allows users to select and download reports. The workbook selects from a view of the FND_CONCURRENT_REQUESTS table; the Applications table that contains details of the requests that have been processed by the concurrent manager. The end user can then use the workbook to select the reports of interest. A custom Discoverer function has been developed that generates a URL to download the request output. This function is used to add a hyperlink into a workbook so that when the user clicks the hyperlink in the report the request output is loaded into a new browser window.

All end users can run this workbook and therefore can view output from reports that have been scheduled by other users. Security that restricts the requests that the end user can select is built into the EUL or database view. When the workbook is run Discoverer checks the end users' responsibility against the output responsibility entered as a parameter to the request and the responsibility that was used to run the report and decides whether the end user has privilege to view the report output.

Schedule Jobs workbook



Request Id	Workbook	Worksheet	Status	Completion Message	Log File	Output File	Workbook Type	Parameter1	Parameter2	Parameter3	Parameter4
337708	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM27"	"Message Reference	"Priority" "% - All	"Category" "
337707	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	NO OUTPUT GENERATED	http:/		HTML	"Pay Team" "TEAM26"	"Message Reference	"Priority" "% - All	"Category" "
337706	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	NO OUTPUT GENERATED	http:/		HTML	"Pay Team" "TEAM25"	"Message Reference	"Priority" "% - All	"Category" "
337705	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM24"	"Message Reference	"Priority" "% - All	"Category" "
337704	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM23"	"Message Reference	"Priority" "% - All	"Category" "
337703	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM22"	"Message Reference	"Priority" "% - All	"Category" "
337702	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	NO OUTPUT GENERATED	http:/		HTML	"Pay Team" "TEAM21"	"Message Reference	"Priority" "% - All	"Category" "
337701	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM20"	"Message Reference	"Priority" "% - All	"Category" "
337700	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM19"	"Message Reference	"Priority" "% - All	"Category" "
337699	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	NO OUTPUT GENERATED	http:/		HTML	"Pay Team" "TEAM18"	"Message Reference	"Priority" "% - All	"Category" "
337698	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM17"	"Message Reference	"Priority" "% - All	"Category" "
337697	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM16"	"Message Reference	"Priority" "% - All	"Category" "
337696	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM15"	"Message Reference	"Priority" "% - All	"Category" "
337695	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM14"	"Message Reference	"Priority" "% - All	"Category" "
337694	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM13"	"Message Reference	"Priority" "% - All	"Category" "
337693	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM12"	"Message Reference	"Priority" "% - All	"Category" "
337692	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM11"	"Message Reference	"Priority" "% - All	"Category" "
337691	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM10"	"Message Reference	"Priority" "% - All	"Category" "
337690	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM09"	"Message Reference	"Priority" "% - All	"Category" "
337689	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM08"	"Message Reference	"Priority" "% - All	"Category" "
337688	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM07"	"Message Reference	"Priority" "% - All	"Category" "
337687	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	NO OUTPUT GENERATED	http:/		HTML	"Pay Team" "TEAM06"	"Message Reference	"Priority" "% - All	"Category" "
337686	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM05"	"Message Reference	"Priority" "% - All	"Category" "
337685	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM04"	"Message Reference	"Priority" "% - All	"Category" "
337684	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM03"	"Message Reference	"Priority" "% - All	"Category" "
337683	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM02"	"Message Reference	"Priority" "% - All	"Category" "
337682	PAG.Pre Payroll Exception R	Pre Payroll Exceptions	COMPLETED	SUCCESS	http:/	http:/	HTML	"Pay Team" "TEAM01"	"Message Reference	"Priority" "% - All	"Category" "
337681	GGUERRERO.Schedule Wor	Schedule Workbook	COMPLETED	SUCCESS	http:/	http:/	HTML	"Schedule Group" "PRE-			
337673	PAG.Transfers Report	Transfers Report: Chan	COMPLETED	SUCCESS	http:/	http:/	HTML	Customers MOD	"UIN Change start da	"UIN Change end	
337625	PAG.Transfers Report	Transfers Report: Chan	COMPLETED	SUCCESS	http:/	http:/	HTML	Customers MOD	"UIN Change start da	"UIN Change end	
337623	PAG.Transfers Report	Transfers Report: Chan	FAILED	UNRESOLVED PARAMETEI	http:/		HTML	Customer MOD	"UIN Change start da	"UIN Change end	
337616	PAG.Transfers Report	Transfers Report: Chan	FAILED	UNRESOLVED PARAMETEI	http:/		HTML	Customer MOD	"UIN Change start da	"UIN Change end	
337613	PAG.Transfers Report	Transfers Report: Chan	COMPLETED	SUCCESS	http:/	http:/	HTML	Customer MOD	"Location Change Sta	"Location Change	

Custom Development Required

This section gives an overview of the design of the components required to implement a scheduling solution. None of the components are difficult or complex to develop but all the components must be designed to work together.

Some details of the implement of the more complex components have been included in this document to illustrate how the solution can be developed, but the detailed implementation will depend on the exact user requirements.

The solution can be divided in three parts:

1. Entering and queuing the details of the workbooks to be processed in Applications
2. Processing the workbooks on the Discoverer workstation
3. Downloading the report output from a Discoverer workbook.

Entering the concurrent request…

The schedule workbook concurrent program is a PL/SQL procedure that is run when the concurrent manager processes the request. The procedure validates the parameters that have been entered by the end user. The program has custom valuesets associated with the responsibility, type and workbook parameters that provide a list of values for the user to select. The valueset for the workbook selects the names of the workbooks stored in the EUL that the user will be able to schedule.

The concurrent program does not process the Discoverer workbook; once the parameters have been validated the request adds the workbook details to the scheduled jobs queue. The schedule jobs queue is implemented as a database table as defined below. Initially, the job status is set to 'WAITING'.

```
CREATE TABLE XXDIS.XXDIS_SCHEDULE_QUEUE
(REQUEST_ID NUMBER(15,0) NOT NULL,
 USER_NAME VARCHAR2(100),
 RESPONSIBILITY_NAME VARCHAR2(100),
 WORKBOOK VARCHAR2(240) NOT NULL,
 WORKSHEET VARCHAR2(240) NOT NULL,
 WORKBOOK_TYPE VARCHAR2(10) DEFAULT 'HTML',
 PARAMETER1 VARCHAR2(240),
 PARAMETER2 VARCHAR2(240),
 PARAMETER3 VARCHAR2(240),
 PARAMETER4 VARCHAR2(240),
 PARAMETER5 VARCHAR2(240),
 PARAMETER6 VARCHAR2(240),
 PARAMETER7 VARCHAR2(240),
 PARAMETER8 VARCHAR2(240),
 STATUS VARCHAR2(20) NOT NULL DEFAULT 'WAITING',
 START_DATE DATE,
 WORKSTATION VARCHAR2(100),
 PROCESSED_BY VARCHAR2(100),
 COMPLETION_DATE DATE,
 COMPLETION_MESSAGE VARCHAR2(2000),
 CONSTRAINT XXDIS_SCHEDULE_QUEUE_PK PRIMARY KEY (REQUEST_ID) USING INDEX )
```

The end users can develop scheduling list workbooks; a workbook containing a list of workbooks that need to be processed. These workbooks call a SUBMIT_JOB function that submits a request to the concurrent manager.

This PL/SQL SUBMIT_JOB function is mapped into the Discoverer EUL and uses the standard Oracle API to create a new concurrent request to process the Discoverer workbook that have been supplied in parameter to the function. The end user can call this function by creating a calculated item in a workbook. The function then checks which Application user is running the workbook, so that only when the workbook is run on the Discoverer workstations are the new concurrent requests created.

Processing the Discoverer workbook…

VB scripting is used to process the job on the workstation. VB scripts run on any Windows environment and can be scheduled to run using the Windows task manager. VB scripts can be used to select data directly from the database and to run Discoverer from the command line interface. The VB script is kept simple by using PL/SQL database functions to start and complete jobs in the database and a database view to build the Discoverer export command that are to be processed on the workstation.

The VB script connects to the database every minute and calls a PL/SQL function to check if there are any jobs on the Discoverer scheduled jobs queue. The VB Script connects to the database using a database account that has been granted access only to this package and view.

The function locks the Discoverer schedule jobs table and if a job is waiting to be processed the function updates the state of the selected job to be 'STARTED'. The database lock ensures that contention between the Discoverer workstations and each job is only processed once. The code fragment below shows how a VB script can connect to an Oracle database and run the START_JOB PL/SQL procedure in the XXDIS_SCHEDULE_PKG package. Global string variables such as gsUser are loaded from a configuration file at the start of the script.

```
Set tkgoDB =CreateObject("ADODB.Connection")
' open a connection to the database
msConnect = "UID=" & gsUser & ";PWD=" & gsPassword & ";CONNECTSTRING=" & gsInstance
tkgoDB.Open "Driver={Microsoft ODBC for Oracle};" & msConnect

Do While True
  ' Check if there is a job to process
  Set moRS=tkgoDB.execute( _
          "SELECT apps.xxdis_schedule_pkg.start_job REQUEST_ID FROM dual")
  moRS.MoveFirst
  msRequest = moRS("REQUEST_ID")

  ' If no jobs then exit
  If msRequest = "0" Then
     Exit Do
  End If
```

The START_JOB function checks the queue for any jobs that are to be processed. The code below shows the start of the START_JOB PL/SQL function.

```
FUNCTION start_job RETURN VARCHAR2
IS
PRAGMA AUTONOMOUS_TRANSACTION;
v_request_id                  NUMBER(15);
BEGIN
-- lock table to prevent any other sessions trying to select the min request id
LOCK TABLE XXDIS.XXMDIS_SCHEDULE_QUEUE IN SHARE ROW EXCLUSIVE MODE;

-- find the next job
SELECT request_id
INTO v_request_id
FROM xxmod.xxdis_schedule_queue
WHERE request_id = (SELECT MIN(request_id)
                    FROM xxdis.xxdis_schedule_queue
                    WHERE status = 'WAITING')
FOR UPDATE;
```

The START_JOB function then updates the schedule queue table adding the details of which workstation and Application user is processing the request. The function then returns the number of the request being processed to the workstation.

```
-- Release lock
COMMIT;
RETURN (v_request_id);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
    COMMIT;
    RETURN (0);
END start_job;
```

The VB script then gets the details of the workbook that needs to be exported from a database view. The database view below returns the Discoverer export command that is to be processed.

```
CREATE OR REPLACE VIEW XXDIS_EXPORT_CMD_V
AS
SELECT xq.request_id, TRANSLATE(
  '/CONNECT '||xq.processed_by||':"'||xq.responsibility_name||
  '"/'||NVL(fu.email_address, 'password')||'@$TNS$'||
  '/OPENDB "'||xq.workbook||'" '||
  '/SHEET "'||xq.worksheet||'" '||
  CASE WHEN xq.parameter1 IS NOT NULL THEN '/PARAMETER '||parameter1||' ' END ||
  CASE WHEN xq.parameter2 IS NOT NULL THEN '/PARAMETER '||parameter2||' ' END ||
  CASE WHEN xq.parameter3 IS NOT NULL THEN '/PARAMETER '||parameter3||' ' END ||
  CASE WHEN xq.parameter4 IS NOT NULL THEN '/PARAMETER '||parameter4||' ' END ||
  CASE WHEN xq.parameter5 IS NOT NULL THEN '/PARAMETER '||parameter5||' ' END ||
  CASE WHEN xq.parameter6 IS NOT NULL THEN '/PARAMETER '||parameter6||' ' END ||
  CASE WHEN xq.parameter7 IS NOT NULL THEN '/PARAMETER '||parameter7||' ' END ||
  CASE WHEN xq.parameter8 IS NOT NULL THEN '/PARAMETER '||parameter8||' ' END ||
  '/EXPORT '||xq.workbook_type||
  ' "$SAMBA$\o'||TO_CHAR(xq.request_id, 'fm999999999990')||'.out" '
  '/LOGFILE l'||TO_CHAR(xq.request_id, 'fm999999999990')||'.log '||
  '/BATCH', CHR(10)||chr(13), '  ') cmd
FROM xxdis.xxdis_schedule_queue xq, fnd_user fu
WHERE xq.processed_by = fu.user_name (+);
```

Notice that in this implementation the password of the dummy Application user is held in the email address field of the Application user. This is just a convenient place to hold the unencrypted password that Discoverer will need to connect as the Application user.

The code fragment below shows how the VB script uses the Discoverer User Edition command line to run the export command. The VB script uses the Discoverer /CMDFILE command and a command file to overcome the limitations of the Windows command line.

```
' Process job
Set moRS=tkgoDB.execute( _
     "SELECT cmd EXPORT_CMD FROM apps.xxdis_export_cmd_v " & _
     "WHERE request_id = " & msRequest)
moRS.MoveFirst
msExpCmd = moRS("EXPORT_CMD")

' write command into a temporary file
msCmdFile = "r" & msRequest & ".cmd"
Set moOutputStream = tkgoFileSystem.CreateTextFile(msCmdFile, True)

' Substitute $SAMBA$ and $TNS$ locally configured variables
moOutputStream.Write Replace(Replace(msCmd, "$SAMBA$", gsOutDir),_
                                            "$TNS$", gsInstance) & vbCRLF
moOutputStream.Close

' Call Discoverer to process the command
msCommand = gsBinDir & gsDiscoExe & " /EUL " & gsEUL &  " /CMDFILE " & msCmdFile

Call tkgoShell.Run (msCommand, 1, true)
```

The script goes on to create a log file for the job and calls a COMPLETE_JOB PL/SQL function to set the status of the job to ‘COMPLETED’ in the database.

Downloading the Report Output…

Retrieving the concurrent request output and log files from the database server is fairly straightforward using the Oracle 9iAS and mod_plsql. You first need to create database directories that are mapped onto the Unix directories contain the request output.

```
CREATE OR REPLACE DIRECTORY XXDIS_OUT AS '/u01/ins_dev/comn/admin/out/INS_dev1';
CREATE OR REPLACE DIRECTORY XXDIS_LOG AS '/u01/ins_dev/comn/admin/log/INS_dev1';
```

The Oracle supplied WPG_DOCLOAD.DOWNLOAD_FILE procedure can be used to download a server file as shown in the procedure below. You will need to register this procedure in Applications using the 'Web Enable PL/SQL' form before it can be used.

```
PROCEDURE download_file(F IN VARCHAR2,    -- encypted filename
                        T IN VARCHAR2,    -- mime type
                        D IN VARCHAR2)    -- directory
IS
l_bfile                 bfile;
BEGIN
IF D IN ('LOG', 'OUT')  AND F IS NOT NULL THEN
    l_bfile := bfilename('XXDIS_'||D, xxdis_encrypt_pkg.decrypt_name(F));

    -- look up ,mime type
    owa_util.mime_header(CASE T
                            WHEN 'HTML' THEN 'text/html'
                            WHEN 'PDF'  THEN 'application/pdf'
                            WHEN 'XLS'  THEN 'application/vnd.ms-excel'
                            WHEN 'TXT'  THEN 'text/plain'
                            WHEN 'TEXT' THEN 'text/plain'
                            WHEN 'CSV'  THEN 'application/vnd.ms-excel'
                            WHEN 'XML'  THEN 'application/vnd.ms-excel'
                                        ELSE 'text/html'
                         END, FALSE, 'ISO-8859-1');
    -- download the file
    wpg_docload.download_file(l_bfile);
END IF
END download_file;
```

Finally, you will need a function to build the URL that contains a call to the DOWNLOAD_FILE function. The function returns a URL that loads the request output into a new browser window. The function below can be mapped into the EUL and returns a URL that can be used as a hyperlink in Discoverer workbooks. You will need to set the content type of the item to be 'FILE' in the EUL for Discoverer to recognise the item as a hyperlink in the workbook.

```
-- return download output file URL
FUNCTION download_ourl (p_request_id IN NUMBER,
                        p_filetype IN VARCHAR2) RETURN VARCHAR2
IS
BEGIN
RETURN
fnd_profile.VALUE('APPS_WEB_AGENT')||'/xxdis_report_output_pkg.download_file?'||
     'F='||xxdis_encrypt_pkg.encrypt_name('o'||TO_CHAR(p_request_id,
'fm099999')||'.out')||'&'||
     'T='||p_filetype||'&'||
     'D=OUT';
END download_ourl;
```

Notice that the procedure and the function use a custom package to encrypt the filename appearing in the URL. The filename is encrypted for two reasons; the filename in the URL cannot be changed by an end user and only alphanumeric characters are used to define the filename in the URL. This package is a wrapper around the DBMS_OBFUSCATION_TOOLKIT encryption package and the encrypt and decrypt functions are shown below.

```
g_key  RAW(16); -- initialised as part of the package initialisation

FUNCTION encrypt_name (p_name IN VARCHAR2) RETURN VARCHAR2
IS
BEGIN
IF p_name IS NULL THEN RETURN NULL; END IF;
RETURN RAWTOHEX(DBMS_OBFUSCATION_TOOLKIT.DESEncrypt(
  input => utl_raw.cast_to_raw(RPAD(p_name, (TRUNC(LENGTH(p_name)/8)+1)*8, chr(0))),
  key   => g_key));

END encrypt_name;

FUNCTION decrypt_name (p_name IN VARCHAR2) RETURN VARCHAR2
IS
BEGIN
IF p_name IS NULL THEN RETURN NULL; END IF;
RETURN (RTRIM(utl_raw.cast_to_varchar2(DBMS_OBFUSCATION_TOOLKIT.DESDecrypt(
  input => HEXTORAW(RPAD(p_name, (TRUNC((LENGTH(p_name)-1)/16)+1)*16, '0')),
  key   => g_key)), chr(0)));

EXCEPTION WHEN VALUE_ERROR THEN RETURN NULL;
END decrypt_name;
```

Further considerations

There are a number of areas where the solution may need to be adapted to meet more specific requirements.

Workbook Processing – The solution can be extended to carry out further processing on the Discoverer workstations. Adobe Acrobat and the Discoverer print command can be used to generate Discoverer reports in PDF. This may be required because PDF is not a Discoverer export option for Application users. The Discoverer workstation could also be used to automatically email reports to other users and mail merge Discoverer output into documents.

Report Output Distribution – Many end users may wish to retrieve reports directly, for example, through a portal, rather than using a Discoverer report to select and download output from a request. The Discoverer workstations can be used to produce HTML reports and then it is straightforward to use the DOWNLOAD_FILE function shown above and database views to build the Discoverer reports into a web based application. Hence this type of Discoverer scheduling can be used to maintain reports, for example, on a corporate web site.

About the Author

Rod West has been using Oracle databases since 1985 and is principal consultant at Cabot Consulting. He specialises in Oracle Applications 11i and Discoverer. Rod can be contacted at rodwest@cabotconsulting.co.uk.

