ORACLE

Oracle Work in Process – Discrete Job Interface (WIP Mass Load)

An Oracle White Paper

ORACLE

Contents

ORACLE

1. Overview

The Objective of this white paper is to discuss about the Work order Interface which is used for importing Discrete job and Repetitive schedule information, and Discrete job operations, material, resource, and scheduling information from any source, using a single process.
We can perform the following actions using Work order interface.

- Can import Discrete jobs against the planned order which are released from planning workbench
- Can import Discrete job operations, components, resources, resource usage, and scheduling details
- update and reschedule recommendations for existing Discrete jobs, and Suggested Repetitive schedules.

2. Features

The Following are the different features of Discrete Job Interface (WIP Mass Load)

- You can insert records from any source for creating the work order such as barcode readers, automated test equipments, cell controllers and other manufacturing and execution systems.
- You can use to automatically create the discrete jobs from oracle advanced planning and scheduling.
- You can turn on or turn off the BOM and routing explosion.
- Manual importing of operations, resource and materials many more.

Operations supported in WIP mass load:

The following are the different operations performed by WIP Mass Load program

	New Discrete Job	Existing Discrete Job
Add Header	Supported	Not Applicable
Add Operation	Supported	Supported
Change Operation	Supported	Supported
Delete Operation	Not Supported	Not Supported
Add Component	Supported	Supported
Change component	Supported	Supported
Delete Component	Not Supported If Explosion Flag is No	Supported
Add operation Resource	Supported	Supported
Change Operation Resource	Supported	Supported
Delete Operation Resource	Not Supported If Explosion Flag is No	Supported
Update Operation Resource Usage	Supported	Supported

You can insert records from third party sources into the Work Order Interface tables by:

- Writing a PL/SQL program or SQL script that maps your source files to the columns in the Work Order Interface tables
- Using a third party program that can map your source files to the interface tables
- Using Oracle Advanced Planning and Scheduling High Level Scheduling Engine to schedule planned and unreleased work orders and import them into Work in Process.
- Using Oracle Master Scheduling/MRP's Planner Workbench to automatically import planned work orders into Work in Process

3. WIP Mass Load Parameters

We have to feed in two parameters while submitting WIP Mass Load concurrent program.

- *Group Id:* Is used for selecting specific list of jobs from Interface tables.
- *Print Report:* Is used to print the job status report for the WIP Mass Load

4. Control Columns

The Discrete Job Interface consists of three tables the WIP_JOB_SCHEDULE_INTERFACE table (wip job Schedule Interface table), the WIP_JOB_DTLS_INTERFACE table (WIP Job Details Interface table) and WIP_INTERFACE_ERRORS table.

WIP_JOB_SCHEDULE_INTERFACE contains header level information related to the requests to create or modify discrete jobs or to create pending repetitive schedules. Several non–WIP products feed this table in order to load Job/Schedule information into WIP like ASCP, Order Management etc. You can use this table to load data into WIP from external sources. You can refer the **Appendix A** to check all the mandatory, optional/derived columns

WIP_JOB_DTLS_INTERFACE contains the detail or line level information related to the requests to add, delete, and modify material, routing, resource usage, and scheduling requirements for new or existing discrete jobs. Third party Applications products use this table in order to load material and resource requirement information into WIP. You can refer the **Appendix B** to check all the mandatory, optional/derived columns.

We have one more table called WIP_INTERFACE_ERRORS which will be populated when WIP Mass Load Program errors. This table stores the complete error message for each error record.

The following control columns (table wise) used to control the behavior of the WIP Mass Load program. Other columns in the interface tables represent the actual data that is used to create or modify Discrete Job or Repetitive schedule behavior while running the WIP Mass Load program.

WIP_JOB_SCHEDULE_INTERFACE:

The following are the controlled columns in WIP_JOB_SCHEDULE_INTERFACE

LOAD_TYPE: We can perform multiple actions on a discrete job/schedule. LOAD_TYPE is used to specify the type of action you want to perform using WIP Mass Load program. Also it decides whether the interface table column is a requited or optional/derived column.

LOAD_TYPE	Type of control
1	Create Standard Discrete Job
2	Create Pending Repetitive Schedule
3	Update Standard or Non-Standard Discrete Job
4	Create Non-Standard Discrete Job

ALLOW_EXPLOSION: This column is used to determine whether the system uses the standard bills of material (BOM) and routing or a custom BOM and routing that you supply. If this flag is set to N or n, you must manually provide a custom BOM and routing; otherwise the system uses the standard BOM and routing.

- If Allow Explosion flag is set to **Yes**, either the start or completion date is used if the other date is null.
- If Allow Explosion flag set to **No**, and values are provided for both start and completion dates, the job is rescheduled from completion date (that is, it is backward scheduled).
- If Allow Explosion flag set to **No**, and either start or completion date is provided, the WIP Mass load program fails because the other date cannot be derived.
- The **default value** for the allow explosion flag is **Yes.**

GROUP_ID: This column is used to identify the batch of records that need to be processed for each run of the WIP Mass Load concurrent program.

HEADER_ID: This column is used to determine individual job/schedule information in the interface table for a given group/batch. HEADER_ID used to link the header records with the detail records.

PROCESS_PHASE: This will be used in combination with PROCESS_STATUS to identify the current status of the record in WIP_JOB_SCHEDULE_INTERFACE table. The default value we need to populate is validation(2).

PROCESS_PHASE	Meaning
2	Validation
3	Explosion
4	Completion
5	Creation

PROCESS_STATUS: This will be used in combination with PROCESS_PHASE to identify the current status of the WIP Mass Load program.

PROCESS_STATUS	Meaning
1	Pending
2	Running
3	Error
4	Complete
5	Warning

Records should be inserted into the WIP_JOB_SCHEDULE_INTERFACE table with a PROCESS_PHASE = 2(Validation) and a PROCESS_STATUS = 1(Pending).

SCHEDULING_METHOD: This column is used for specifying the type of scheduling during the creation of the discrete job or repetitive schedule. The **default value** of SCHEDULING_METHOD is routing based (1).

SCHEDULING_METHOD	Meaning
1	Routing-Based
2	Lead Time
3	Manual

STATUS_TYPE: This column is used to specify the status of the job while loading through WIP Mass Load program. You can create a job in any of the following status.

STATUS_TYPE	Meaning
1	Unreleased
3	Released
4	Complete
5	Complete No Charge
6	On Hold
7	Cancelled
8	Pending Bill Load
9	Failed Bill Load
10	Pending Routing Load
11	Failed Routing Load
12	Closed
13	Pending- Mass Loaded
14	Pending Close
15	Failed Close

CLASS_CODE: If NULL on standard job creation uses default class from WIP parameters. Ignored on reschedule and derived for Repetitive schedules. If the value is NOT NULL, issues a warning message. If a wrong value specified for this column then WIP Mass Load program errors on records that fail validation.

INTERFACE_ID: This column is used for identifying each work order that is loaded individually.

WIP_JOB_DTLS_INTERFACE:

The following are the controlled columns in WIP_JOB_DTLS_INTERFACE

LOAD_TYPE: This is used to specify the type of modification you are planning against a Discrete Job or Repetitive schedule. Also LOAD_TYPE controls whether the interface table column is a requited or optional or optional/derived column.

LOAD_TYPE	Meaning
1	Loading a resource
2	Loading a component
3	Loading an operation
4	Loading multiple resource usage
5	Changing between primary and alternate resources
6	Associating serial numbers
7	Dispatching resource instance
8	Loading resource instance usage

SUBSTITUTION_TYPE: This column is used to specify how you are modifying the job operation, resource or component.

SUBSTITUTION_TYPE	Meaning
1	Delete
2	Add
3	Change
Any other values will cause an error.	

5. WIP Mass Load Behavior in 11i & R12

In 11i, when we are scheduling the job based on completion date as Monday 00:00:00, System is considering the completion time and performing the backward scheduling and calculating the start date based on the completion time.

In real time scenario, if you are giving the completion date to Monday 00:00:00 means the job has to be completed before Monday 00:00:00. As the previous day (Sunday) is going to be holiday (In generalized scenario) System need to do backward scheduling based on the last operation resource availability, which is not happening in release 11i. For better understanding you can refer the following example.

Real time Scenario.

I have a job XX1 for an assembly. This job needs to be completed by 6th January 2014 00:00:00. Calendar defined as Sunday as holiday and the last operation resource is available till 5th January 2014 06:00:00.

In this case the system will do the backward scheduling and calculate the start date. Based on the job scheduling the job will get completed by 5th January 2014 06:00:00.

In case of 11i, if we specify the completion date as 6th January 2014 00:00:00, after scheduling, the job completion date is not changing to 5th January 2014 06:00:00 which is not realistic.

In Release 12, Oracle changed the system behavior such that system behavior matches with real time shop floor environment, hence the job completion date is changing to 5th January 2014 06:00:00 after backward scheduling. So, the job will be completed before 6th January 2014 00:00:00 based on actual resource availability in the shop floor.

As many of the customers only see the date of the work order and doesn't see the time on their real shop floor. So they want it put on Monday, not Saturday (time is not a problem for the customer). Saturday is non-workday on customer's site, so according the R12 results of WIP, customer's feeling that they must work on Saturday; despite they enter completion data as Monday.

As per numerous customers request, Oracle introduced a new profile option in R12, to control the job completion date based on customer requirement. Development created new profile option in order to align Job Start and Completion dates to the organization calendar shifts when creating or re- scheduling discrete jobs from the discrete jobs form.

The patch ***14101323:R12.WIP.B*** introduces new Profile named 'WIP: Align Job Dates with Shift Calendar' having possible values as Yes /No with a **default value of 'Yes'**. This Patch is applicable on **R12.1.x**

New behavior is as follows:
'Yes' - then dates/times will be validated with Shift Calendar and will be aligned to calendar shifts.
'No' - then user entered dates will be retained as is with no alignment to calendar shifts.

NOTE: The actual behavior of WIP Mass Load is not changed with the new profile option. The profile option will be applicable on top of standard WIP Mass Load behavior. That mean you have to feed in respective values to ALLOW_EXPLOSION and SCHEDULING_METHOD

Following are some of the sample inserts statements for creating and updating discrete job

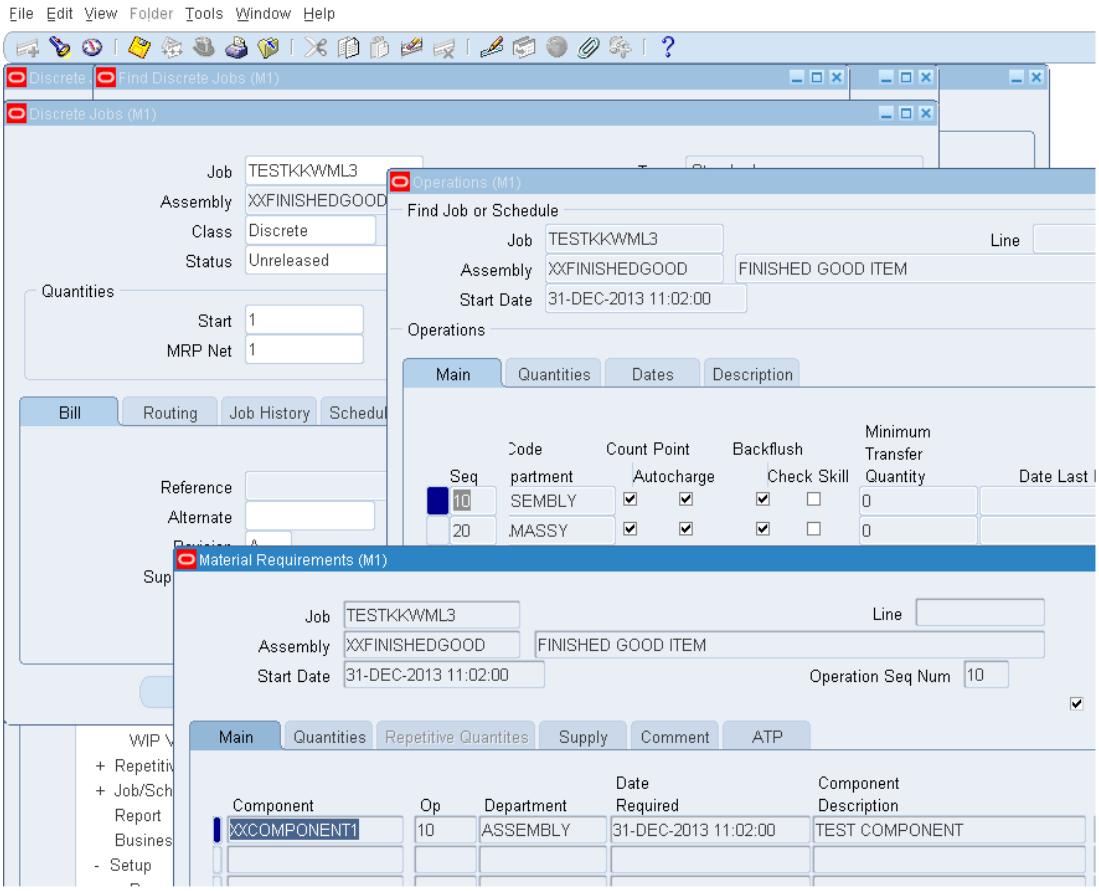
6. Discrete Job creation With ALLOW_EXPLOSION is set to Y

The following SQL can be used to create the discrete job with ALLOW_EXPLOSION Flag set as 'Yes'. This will create the discrete job with any automatic explosion means the components, operations and resources are copied from existing Bill of Materials and Routing respectively.

```
INSERT INTO   WIP_JOB_SCHEDULE_INTERFACE
 (last_update_date
,last_updated_by
,creation_date
,created_by
,Last_update_login
,group_id
,organization_id
,load_type
,wip_entity_id
,process_phase
,process_status
,header_id
,allow_explosion
,net_quantity
,start_quantity
,primary_item_id
,COMPLETION_SUBINVENTORY
,CLASS_CODE
,JOB_NAME
,first_unit_start_date
,FIRST_UNIT_COMPLETION_DATE
,STATUS_TYPE
)
VALUES (
 SYSDATE,       -- LAST UPDATE DATE
 1318,          -- LAST UPDATED BY
 SYSDATE,       -- CREATION DATE
 1318,          -- CREATED BY
 1318,          -- LAST UPDATE LOGIN
 1318,          -- GROUP ID
 207,           -- ORG ID
 1,             --1. Create standard DJ, 2.creative pending repetitive schedule, 3. Update standard or non standard
                -- job, 4.  Create non standard job
 null,          -- WIP ENTITY ID
 2,             --2. Validation, 3.  EXPLOSION, 4. COMPLETION, 5. CREATION
 1,             --1. Pending 2. running, 3. Error 4. Complete 5. Warning
 1318    --HEADER ID
 ,'Y'           --ALLOW EXPLOSION
 ,1             --NET QUANTITY
 ,1             --START QUANTITY
 ,323963        --PRIMARY ITEM ID or ASSEMBLY ITEM ID
```

```
,'FGI'          -- COMPLETION SUB INVENTORY
,'Discrete'     --CLASS CODE
,'TESTKKWML3'   -- DISCRETE JOB NAME
,sysdate        -- FIRST UNIT START DATE
,sysdate+1      -- FIRST UNIT COMPLETION DATE
,1               --STATUS_TYPE
);
```

The follow screenshot shows the created job.



7. Discrete Job creation With ALLOW_EXPLOSION is set to N

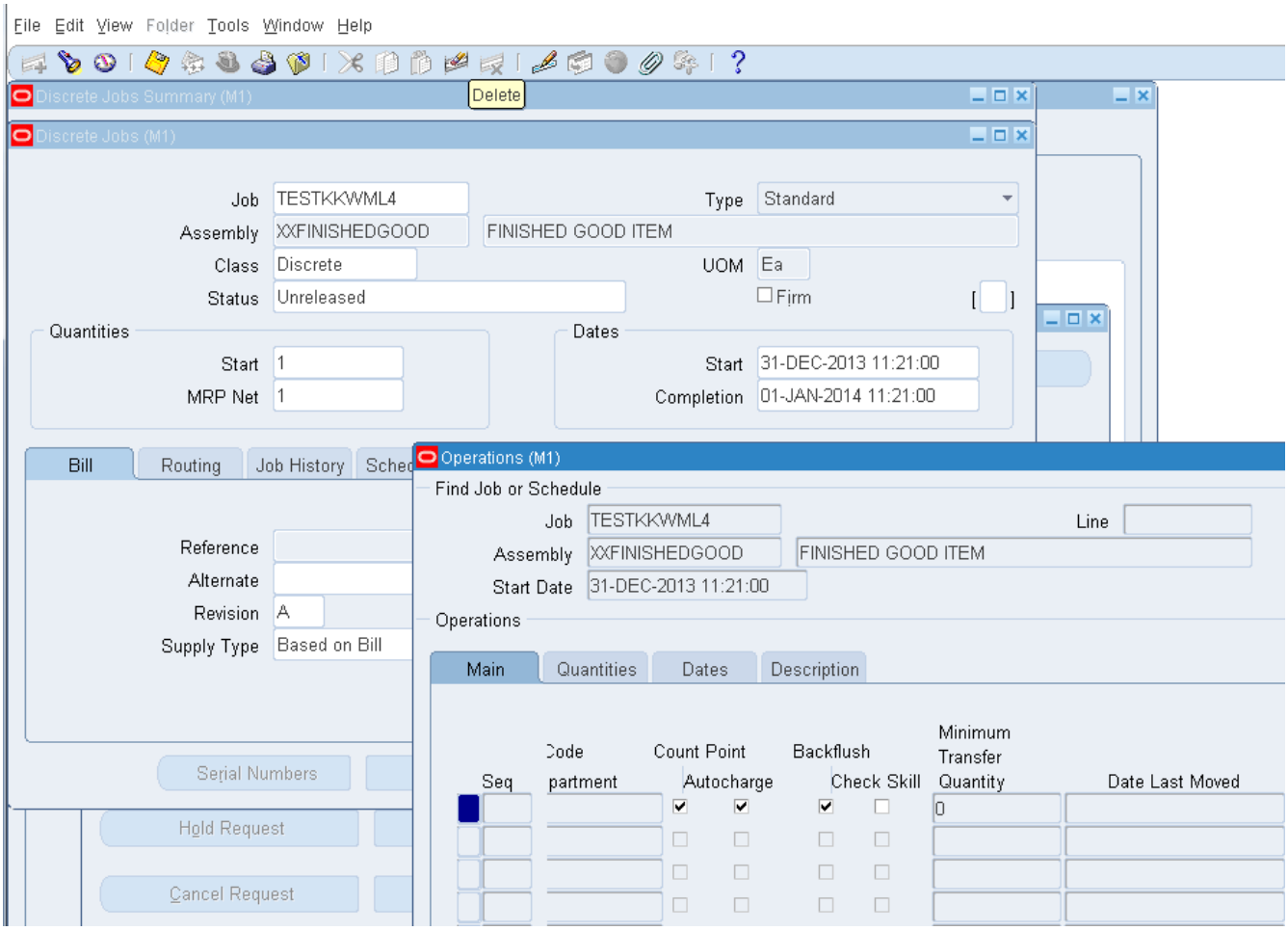
The following SQL can be used to create the discrete job with ALLOW_EXPLOSION Flag set as 'N'. This will create the discrete job without any automatic explosion. If we are setting the ALLOW_EXPLOSION flag as 'No', then we have to give the scheduling method along with first and last unit start and completion dates.

```
INSERT INTO   WIP_JOB_SCHEDULE_INTERFACE
  (last_update_date
,last_updated_by
,creation_date
,created_by
,Last_update_login
,group_id
,organization_id
,load_type
,wip_entity_id
,process_phase
,process_status
,header_id
```

```
,allow_explosion
,net_quantity
,start_quantity
,primary_item_id
,COMPLETION_SUBINVENTORY
,CLASS_CODE
,JOB_NAME
,first_unit_start_date
,FIRST_UNIT_COMPLETION_DATE
,scheduling_method
,last_unit_start_date
,LAST_UNIT_COMPLETION_DATE
,STATUS_TYPE
)
VALUES (
  SYSDATE,        -- LAST UPDATE DATE
  1318,           -- LAST UPDATED BY
  SYSDATE,        -- CREATION DATE
  1318,           -- CREATED BY
  1318,           -- LAST UPDATE LOGIN
  1318,           -- GROUP ID
  207,            -- ORG ID
  1,              --1. Create standard DJ, 2.creative pending repetitive schedule, 3. Update standard or non standard
                  -- job, 4. create non standard job
  null,           -- WIP ENTITY ID
  2,              --2. validation, 3.  EXPLOSION, 4. COMPLETION, 5. CREATION
  1,              --1. pending 2. running, 3. error 4. complete 5. warning
  1318            --HEADER ID
  ,'N'            --ALLOW EXPLOSION
  ,1              --NET QUANTITY
  ,1              --START QUANTITY
  ,323963         --PRIMARY ITEM ID
  ,'FGI'          -- COMPLETION SUB INVENTORY
  ,'Discrete'     --CLASS CODE
  ,'TESTKKWML4'   -- DISCRETE JOB NAME
  ,sysdate        -- FIRST UNIT START DATE
  ,sysdate+1      -- FIRST UNIT COMPLETION DATE
  ,3                  --scheduling method  1- Routing based; 2- Lead Time; 3- Manual
  ,sysdate          --last unit start date
  ,sysdate+1        --last unit completion date
 ,1                   --STATUS_TYPE
  );
```

The following screenshot shows the result job

In this case I have feed the scheduling method as manual (3). Hence I did not feed any operation or resource information. If you are choosing the scheduling method as routing based (1) then you have to feed in the operations and resource through WIP_JOB_DTLS_INTERFACE table. Otherwise the job will not be created.



8. Adding an operation to the Existing Discrete Job

If you want to update an existing job you have to feed in records in both WIP_JOB_SCHEDULE_INTERFACE and WIP_JOB_DTLS_INTERFACE. Following sample script will help you to add an operation to existing job. In this case the ALLOW_EXPLOSION flag is set as "N" and the LOAD_TYPE changed to "3" which means "update standard or non standard job". In this example in WIP_JOB_DTLS_INTERFACE, I choose the SCHEDULING_METHOD as "3" which means manual scheduling, LOAD_TYPE as 3 and SUBSTITUTION_TYPE as 2 which means adding the operation to the existing job.

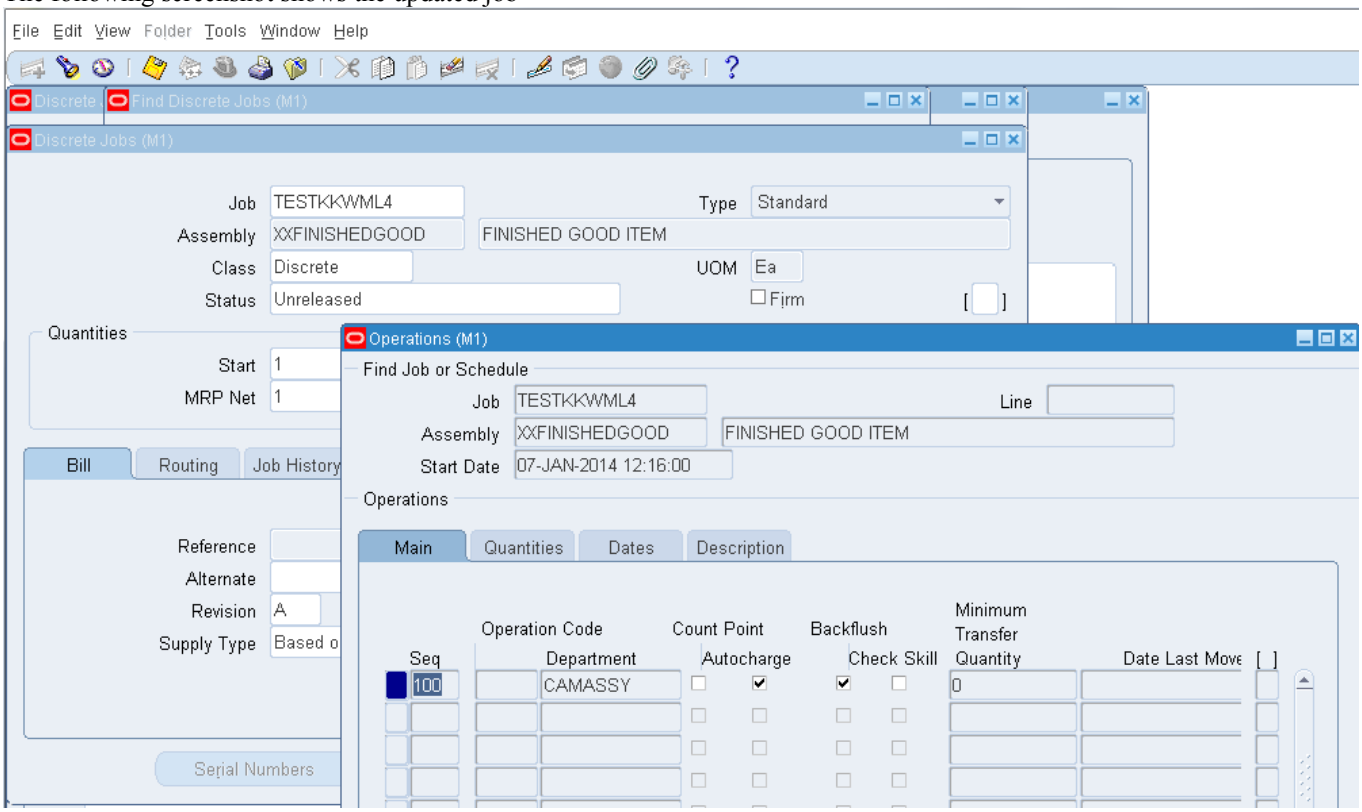
INSERT INTO **WIP_JOB_SCHEDULE_INTERFACE**
(last_update_date
,last_updated_by
,creation_date
,created_by
,Last_update_login
,group_id
,organization_id
,load_type
,wip_entity_id
,process_phase
,process_status
,header_id
,allow_explosion
,net_quantity
,start_quantity

```
,primary_item_id
,COMPLETION_SUBINVENTORY
,CLASS_CODE
,JOB_NAME
,first_unit_start_date
,FIRST_UNIT_COMPLETION_DATE
,scheduling_method
,last_unit_start_date
,LAST_UNIT_COMPLETION_DATE
,STATUS_TYPE
)
VALUES (
 SYSDATE,       -- LAST UPDATE DATE
 1318,          -- LAST UPDATED BY
 SYSDATE,       -- CREATION DATE
 1318,          -- CREATED BY
 1318,          -- LAST UPDATE LOGIN
 1318,          -- GROUP ID
 207,           -- ORG ID
 3,             --3. update standard or non standard job
 850086,                -- WIP ENTITY ID
 2,             --2. validation, 3.  EXPLOSION, 4. COMPLETION, 5. CREATION
 1,             --1. pending 2. running, 3. error 4. complete 5. warning
 1318    --HEADER ID
 ,'N'           --ALLOW EXPLOSION
 ,1             --NET QUANTITY
 ,1             --START QUANTITY
 ,323963        --PRIMARY ITEM ID
 ,'FGI'         -- COMPLETION SUB INVENTORY
 ,'Discrete'    --CLASS CODE
 ,'TESTKKWML4'   -- DISCRETE JOB NAME
 ,sysdate       -- FIRST UNIT START DATE
 ,sysdate+1     -- FIRST UNIT COMPLETION DATE
 ,3                 --scheduling method  1- Routing based; 2- Lead Time; 3- Manual
 ,sysdate        --last unit start date
 ,sysdate+1      --last unit completion date
,1                 --STATUS_TYPE
  );

Insert into WIP_JOB_DTLS_INTERFACE
(group_id
,organization_id
,operation_seq_num
,department_id
,load_type
,substitution_type
,process_phase
,process_status
,last_update_date
,last_updated_by
,creation_date
,created_by
,parent_header_id
,COUNT_POINT_TYPE
,BACKFLUSH_FLAG
)
values(
1318            --group id
```

```
,207            -- ORG ID
,100            --operation sequence number
,23872          --department id (CAMASSY)
,3               --load_type
,2            --substitution_type
,2            --process_phase
,1            -- process_status
,sysdate       --last update date
,1318         --last updated by
,sysdate       --creation date
,1318         --created by
,1318         --header id
,2
,1
);
```

The following screenshot shows the updated job



9. Adding a resource to an existing job

In this scenario we will add a resource to an existing job operation. Hence we are choosing the load type as 3 in WIP_JOB_SCHEDULE_INTERFACE which mean we are updating the existing discrete job. In this example in WIP_JOB_DTLS_INTERFACE, I am taking the, LOAD_TYPE as 1 and SUBSTITUTION_TYPE as 2 which means adding a resource to the operation of the existing job

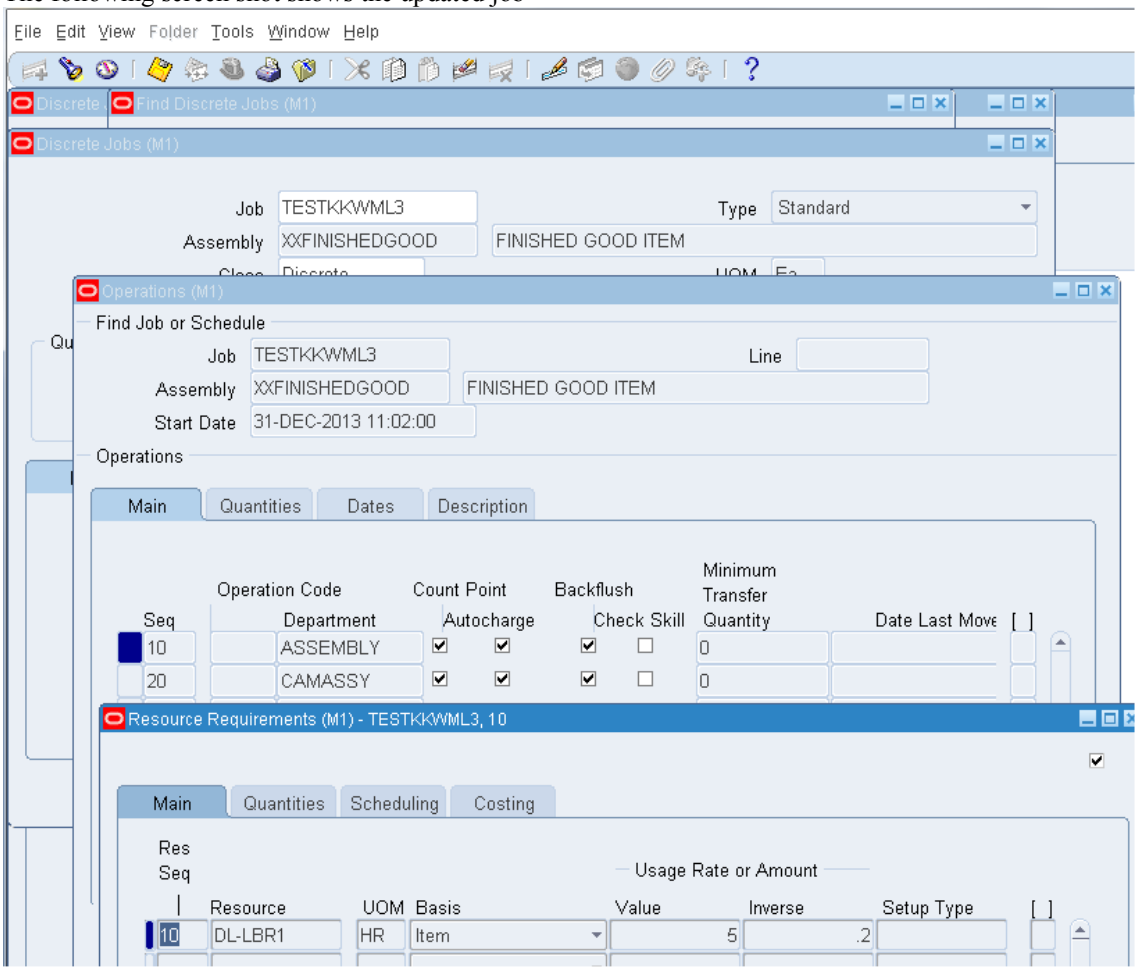
```
INSERT INTO   WIP_JOB_SCHEDULE_INTERFACE
  (last_update_date
,last_updated_by
,creation_date
,created_by
,Last_update_login
```

```
,group_id
,organization_id
,load_type
,wip_entity_id
,process_phase
,process_status
,header_id
,allow_explosion
,net_quantity
,start_quantity
,primary_item_id
,COMPLETION_SUBINVENTORY
,CLASS_CODE
,JOB_NAME
,first_unit_start_date
,FIRST_UNIT_COMPLETION_DATE
,scheduling_method
,last_unit_start_date
,LAST_UNIT_COMPLETION_DATE
,STATUS_TYPE
)
VALUES (
 SYSDATE,       -- LAST UPDATE DATE
 1318,          -- LAST UPDATED BY
 SYSDATE,       -- CREATION DATE
 1318,          -- CREATED BY
 1318,          -- LAST UPDATE LOGIN
 1318,          -- GROUP ID
 207,           -- ORG ID
 3,             --3. update standard or non standard job
 850086,                -- WIP ENTITY ID
 2,             --2. validation, 3.  EXPLOSION, 4. COMPLETION, 5. CREATION
 1,             --1. pending 2. running, 3. error 4. complete 5. warning
 1318    --HEADER ID
 ,'N'           --ALLOW EXPLOSION
 ,1             --NET QUANTITY
 ,1             --START QUANTITY
 ,323963        --PRIMARY ITEM ID
 ,'FGI'         -- COMPLETION SUB INVENTORY
 ,'Discrete'    --CLASS CODE
 ,'TESTKKWML4'   -- DISCRETE JOB NAME
 ,sysdate       -- FIRST UNIT START DATE
 ,sysdate+1     -- FIRST UNIT COMPLETION DATE
 ,3                 --scheduling method  1- Routing based; 2- Lead Time; 3- Manual
 ,sysdate        --last unit start date
 ,sysdate+1      --last unit completion date
,1                      --STATUS_TYPE
  );

Insert into WIP_JOB_DTLS_INTERFACE
(group_id
,organization_id
,operation_seq_num
,RESOURCE_SEQ_NUM
,RESOURCE_ID_NEW
,USAGE_RATE_OR_AMOUNT
,SCHEDULED_FLAG
,ASSIGNED_UNITS
```

```
,BASIS_TYPE
,AUTOCHARGE_TYPE
,STANDARD_RATE_FLAG
,START_DATE
,COMPLETION_DATE
,load_type
,substitution_type
,process_phase
,process_status
,last_update_date
,last_updated_by
,creation_date
,created_by
,parent_header_id
)
values
(1318     --group id
,207      -- org id
,100     --operation sequnece number
,10      --RESOURCE_SEQ_NUM
,53496      --resource id new(CT)
,10.5       -- usage rate
,2         --SCHEDULE FLAG
,1        --ASSIGNED_UNITS
,1      --BASIS_TYPE
,1 --AUTOCHARGE_TYPE
,1 --STANDARD_RATE_FLAG
,sysdate --START_DATE
,sysdate --COMPLETION_DATE
,1   --load_type
,2  --substitution_type
,2   --process_phase
,1    -- process_status
,sysdate  --last update date
,1318     --last updated by
,sysdate  -- creation date
,1318     --created by
,1318     --parent header id
);
```

The following screen shot shows the updated job



10. Adding a component to existing job

In this scenario we will add a component to an existing job operation. Hence we are choosing the load type as 3 in WIP_JOB_SCHEDULE_INTERFACE which mean we are updating the existing discrete job. In this example in WIP_JOB_DTLS_INTERFACE, I am taking the, LOAD_TYPE as 2 and SUBSTITUTION_TYPE as 2 which means adding a component to the existing job.

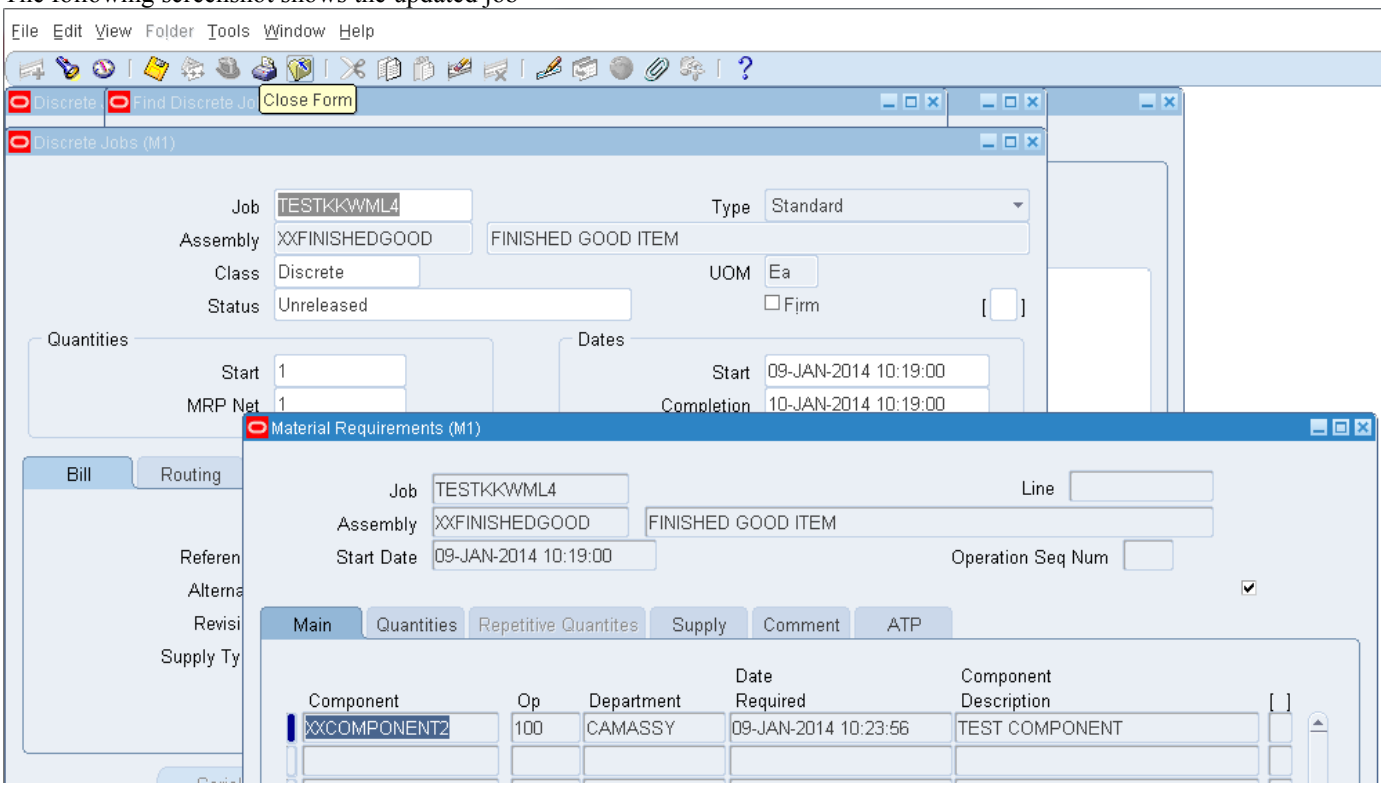
```
INSERT INTO   WIP_JOB_SCHEDULE_INTERFACE
  (last_update_date
,last_updated_by
,creation_date
,created_by
,Last_update_login
,group_id
,organization_id
,load_type
,wip_entity_id
,process_phase
,process_status
,header_id
,allow_explosion
,net_quantity
,start_quantity
```

```
,primary_item_id
,COMPLETION_SUBINVENTORY
,CLASS_CODE
,JOB_NAME
,first_unit_start_date
,FIRST_UNIT_COMPLETION_DATE
,scheduling_method
,last_unit_start_date
,LAST_UNIT_COMPLETION_DATE
,STATUS_TYPE
)
VALUES (
 SYSDATE,       -- LAST UPDATE DATE
 1318,          -- LAST UPDATED BY
 SYSDATE,       -- CREATION DATE
 1318,          -- CREATED BY
 1318,          -- LAST UPDATE LOGIN
 1318,          -- GROUP ID
 207,           -- ORG ID
 3,             --3. update standard or non standard job
 850086,                -- WIP ENTITY ID
 2,             --2. validation, 3.  EXPLOSION, 4. COMPLETION, 5. CREATION
 1,             --1. pending 2. running, 3. error 4. complete 5. warning
 1318    --HEADER ID
 ,'N'           --ALLOW EXPLOSION
 ,1             --NET QUANTITY
 ,1             --START QUANTITY
 ,323963        --PRIMARY ITEM ID
 ,'FGI'         -- COMPLETION SUB INVENTORY
 ,'Discrete'    --CLASS CODE
 ,'TESTKKWML4'   -- DISCRETE JOB NAME
 ,sysdate       -- FIRST UNIT START DATE
 ,sysdate+1     -- FIRST UNIT COMPLETION DATE
 ,3                 --scheduling method  1- Routing based; 2- Lead Time; 3- Manual
 ,sysdate        --last unit start date
 ,sysdate+1      --last unit completion date
,1               --STATUS_TYPE
  );

Insert into WIP_JOB_DTLS_INTERFACE
(group_id
,organization_id
,OPERATION_SEQ_NUM
,DATE_REQUIRED
,INVENTORY_ITEM_ID_NEW
,MRP_NET_FLAG
,QUANTITY_ISSUED
,QUANTITY_PER_ASSEMBLY
,REQUIRED_QUANTITY
,WIP_SUPPLY_TYPE
,load_type
,substitution_type
,process_phase
,process_status
,creation_date
,created_by
,parent_header_id
,last_update_date
```

```
,last_updated_by
)
values
(1318     --group id
,207      -- org id
,100     --operation sequnece number
,sysdate  --date required
,323965      --inventory item id new
,1       -- mrp net flag
,1         --quantity issued
,1        --quantity per assembly
,1      --required quantity
,2      --wip supply type
,2      --load_type
,2      --substitution_type   1 Delete; 2 Add ; 3 Change
,2   --process_phase
,1    -- process_status
,sysdate  -- creation date
,1318     --created by
,1318     --parent header id
,sysdate    --last update date
,1318   --last updated by
);
```

The following screenshot shows the updated job

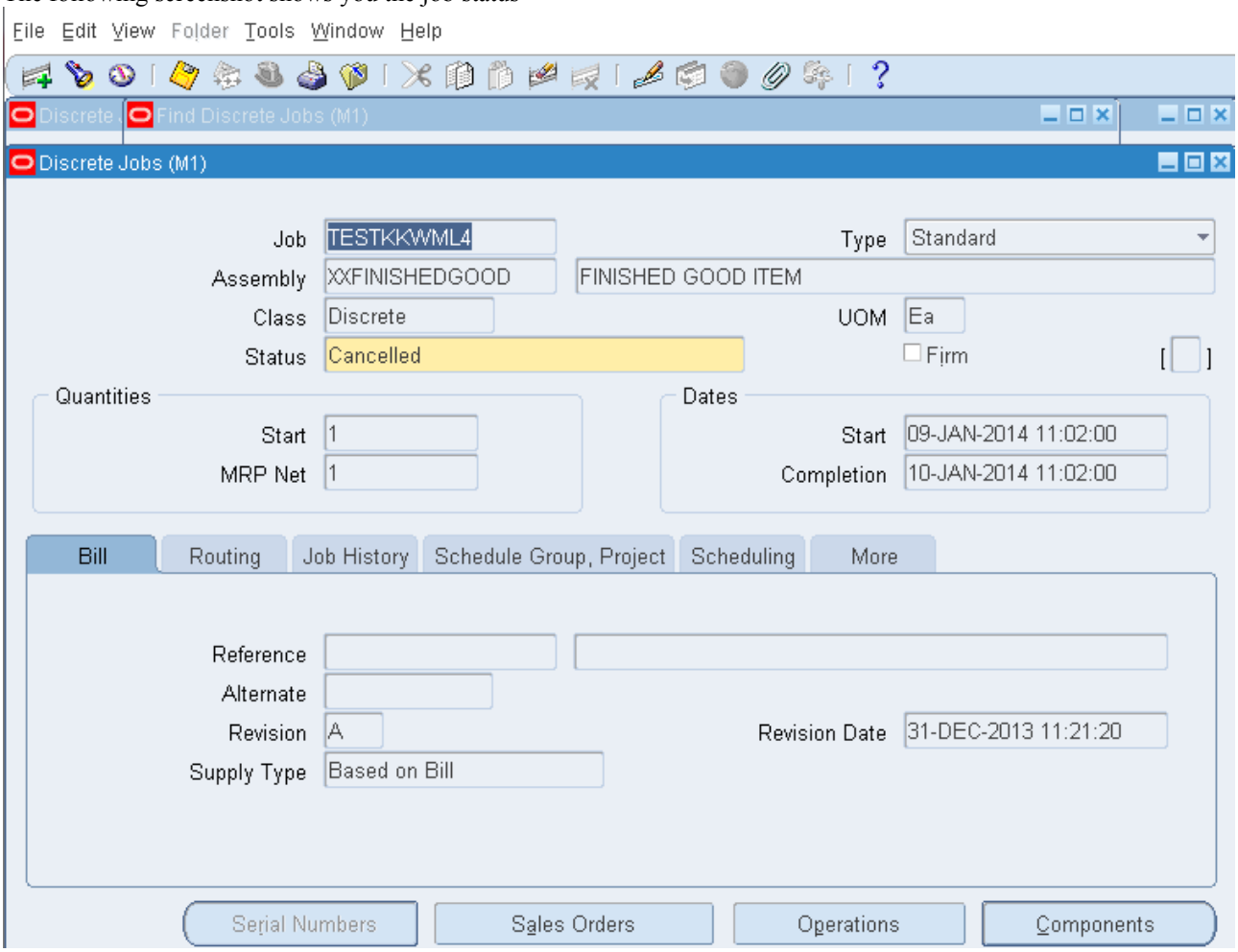


11. Cancelling existing discrete Job

In this scenario I am planning to cancel an existing job. Hence I choose the load type as 3 and STATUS_TYPE as 7 in WIP_JOB_SCHEDULE_INTERFACE which mean we are updating the existing discrete job with job status as cancelled.

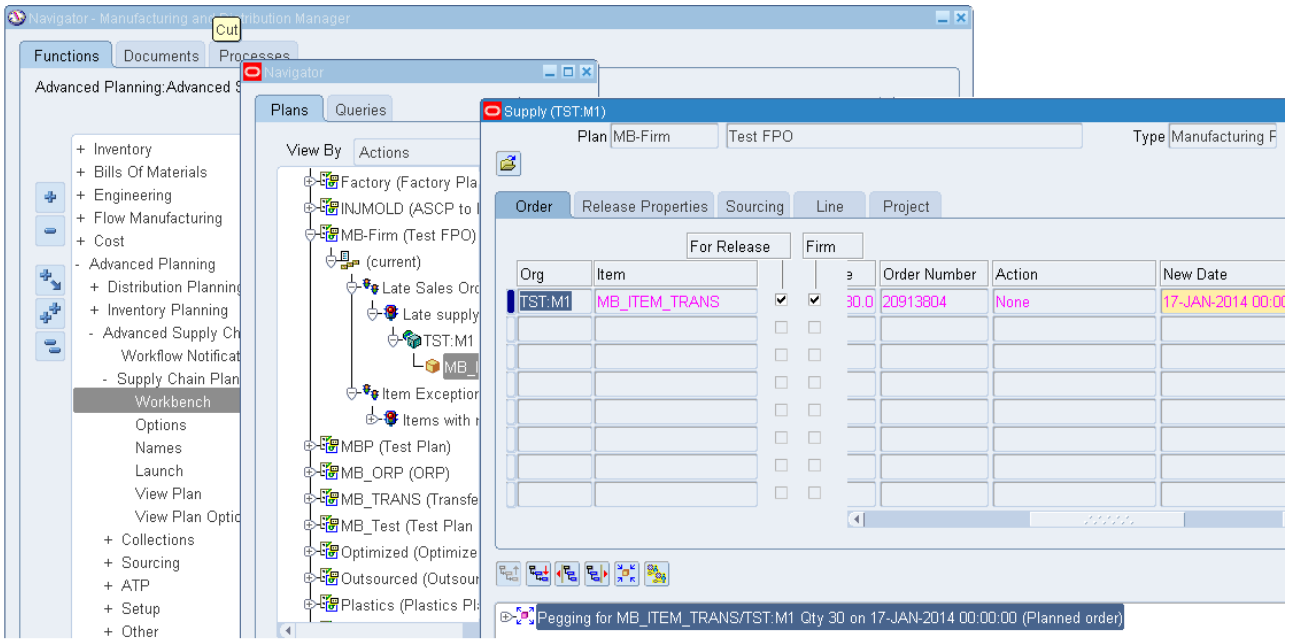
```
INSERT INTO   WIP_JOB_SCHEDULE_INTERFACE
  (last_update_date
,last_updated_by
,creation_date
,created_by
,Last_update_login
,group_id
,organization_id
,load_type
,wip_entity_id
,process_phase
,process_status
,header_id
,allow_explosion
,net_quantity
,start_quantity
,primary_item_id
,COMPLETION_SUBINVENTORY
,CLASS_CODE
,JOB_NAME
,first_unit_start_date
,FIRST_UNIT_COMPLETION_DATE
,scheduling_method
,last_unit_start_date
,LAST_UNIT_COMPLETION_DATE
,STATUS_TYPE
)
VALUES (
  SYSDATE,        -- LAST UPDATE DATE
  1318,           -- LAST UPDATED BY
  SYSDATE,        -- CREATION DATE
  1318,           -- CREATED BY
  1318,           -- LAST UPDATE LOGIN
  1318,           -- GROUP ID
  207,            -- ORG ID
  3,              --1. create standard DJ, 2.creative pending repetitive schedule, 3. update standard or non standard job,
                      --4. create non standard job
  850086,                 -- WIP ENTITY ID
  2,              --2. validation, 3.  EXPLOSION, 4. COMPLETION, 5. CREATION
  1,              --1. pending 2. running, 3. error 4. complete 5. warning
  1318    --HEADER ID
  ,'N'            --ALLOW EXPLOSION
  ,1              --NET QUANTITY
  ,1              --START QUANTITY
  ,323963         --PRIMARY ITEM ID
  ,'FGI'          -- COMPLETION SUB INVENTORY
  ,'Discrete'     --CLASS CODE
  ,'TESTKKWML4'   -- DISCRETE JOB NAME
  ,sysdate        -- FIRST UNIT START DATE
  ,sysdate+1      -- FIRST UNIT COMPLETION DATE
  ,3                  --scheduling method  1- Routing based; 2- Lead Time; 3- Manual
  ,sysdate         --last unit start date
  ,sysdate+1       --last unit completion date
  ,7       --STATUS_TYPE
   );
```

The following screenshot shows you the job status



12. Releasing a Planned Order from ASCP

The following screenshot shows the planned order that is released from ASCP. When a planned order is released from ASCP, the planning work bench the records will get populated in to MSC_WIP_JOB_SCHEDULE_INTERFACE and MSC_WIP_JOB_DTLS_INTERFACE tables from there system will populate the records into the WIP Interface tables. After inserting the data in to the interface tables, the WIP Mass Load concurrent program will be triggered automatically to create a discrete job based on the planned order requirement.

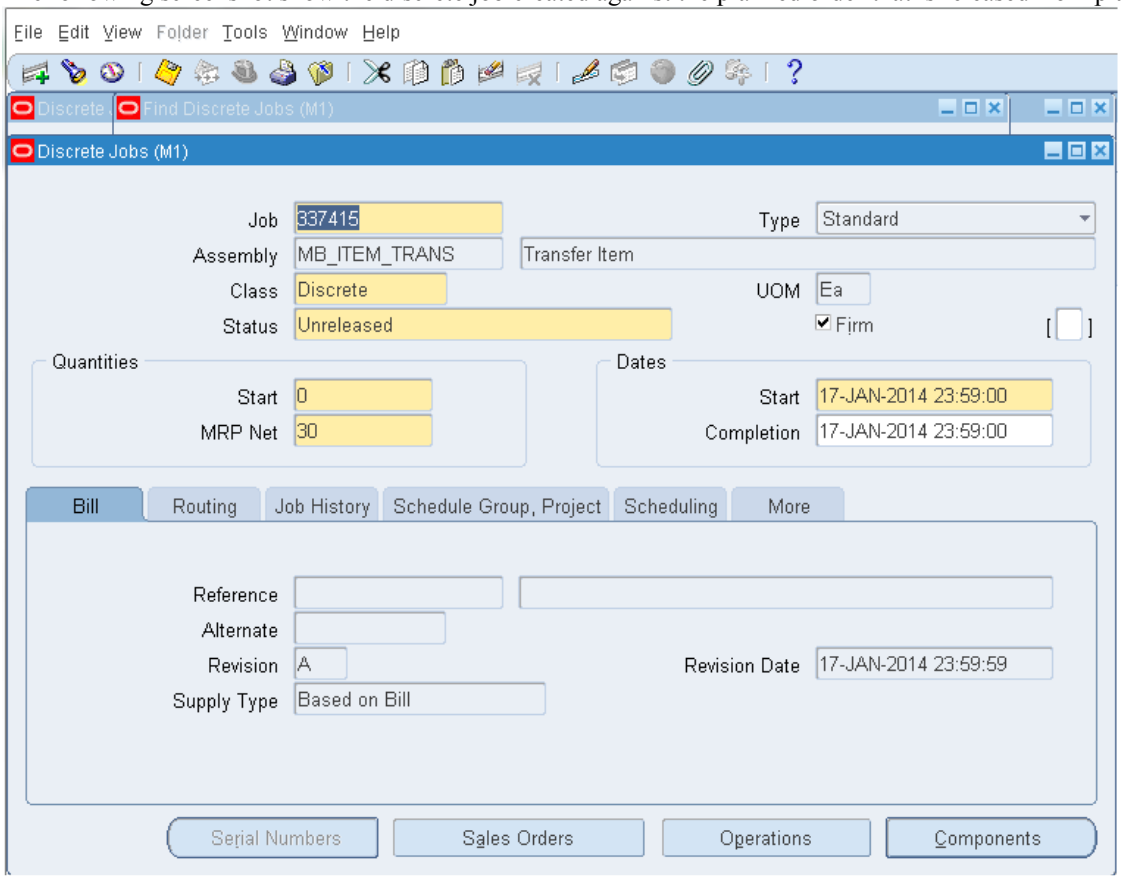


The following data was inserted in WIP_JOB_SCHEDULE_INTERFACE table for a sample planned order released from ASCP as per the above screenshot.

```
Insert into wip_job_schedule_interface
(LAST_UPDATE_DATE
,LAST_UPDATED_BY
,CREATION_DATE
,CREATED_BY
,LAST_UPDATE_LOGIN
,GROUP_ID
,SOURCE_CODE
,SOURCE_LINE_ID
,ORGANIZATION_ID
,LOAD_TYPE
,STATUS_TYPE
,LAST_UNIT_COMPLETION_DATE
,PRIMARY_ITEM_ID
,CLASS_CODE
,JOB_NAME
,FIRM_PLANNED_FLAG
,START_QUANTITY
,PROCESS_PHASE
,PROCESS_STATUS
,SCHEDULING_METHOD
,NET_QUANTITY
,DUE_DATE
,ALLOW_EXPLOSION
,HEADER_ID
)
values
(to_date('17-JAN-14','DD-MON-RR')
,1318
,to_date('17-JAN-14','DD-MON-RR')
,1318
,6302038
,4086722
,'MSC'
```

```
,20913804
,207
,1
,1
,to_date('17-JAN-14','DD-MON-RR')
,311963
,'Discrete'
,'337415'
,1
,0
,2
,1
,1
,30
,to_date('17-JAN-14','DD-MON-RR')
,'Y'
,20913804
);
```

The following screenshot show the discrete job created against the planned order that is released from planning



13. Common Errors:

The following are some of the general errors we will be getting while running the WIP Mass Load

1. *Invalid or missing WIP Accounting Class*

ANS): Set a default WIP accounting class in the WIP parameters for the organization.

2. *Invalid value for Scheduling Method*

ANS): Normally you will get this kind of error when the ALLOW_EXPLOSION is set to No and no value was assigned to SECHDULING_METHOD. Routing based scheduling cannot be done if BOM explosion is not allowed. Please change the scheduling method to manual, and provide start and completion dates, or set the ALLOW_EXPLOSION to Y.

3. *System cannot add or change operation because requested department_id does not exist in bom_departments table*.

ANS): We will get this kind of error in case we missed to provide department information while adding the operation or the department we have provided is not in the same organization.

4. *Invalid resource. Check Bom_Resources table. Resource does not exist in Bom_resources table*.

ANS): We will get this kind of error in case the resource you are adding is not in the operation department. You need to make sure that the resource is already attached to the department which is available for the operation.

5. *System cannot add resource or material because the requested operation_seq_num does not exist*

ANS): You will be getting this error incase if you are trying to enter the resource/component to an operations which is not exists against the discrete job.

6. *System cannot add or change material requirement because requested job/operation_seq_num /inventory_item_id_new combo has already existed*

ANS): If a particular component is already exists and you are trying to add same component is again using detail interface, you will be getting this error. You should be careful enough, such that you should not add a component with the job, operation_seq_num, and inventory_item_id_new combination if the same is already exists in the discrete job.

7. *Cannot read value for profile option RPM_CONC_PROCESS_ID in routine &ROUTINE*.

ANS): There are multiple possibilities for getting such errors. Make sure that all the following setups are place

a. When updating a job through interface by inserting rows in the table WIP_JOB_DTLS_INTERFACE ensures that there is a corresponding operation_seq_num in WIP_OPERATIONS. When updating the components of the current job, the operation_seq_num should exist for that job in the table WIP_OPERATIONS.
b. Please make sure that you have done proper WIP setup
c. Check whether the current accounting period is open or not. If not please open the accounting periods where the job dates falls. The navigation for checking the accounting period status is Inventory - Accounting Close Cycle - Inventory Period

8. *Exiting with warning! calling mrepgrm_end_program 1*

ANS): There are multiple possibilities for getting this type of errors. Make sure that all the following precautions were taken care.

a. Please check the profile option INV: Dynamic Precision Option for Quantity on Reports to a valid value at site level. This profile should not have null value.
b. If you are releasing the planned order from ASCP, please check whether the default job class is set or not. You can use the following steps for achieving this.
   ==>Navigate: Adv Supply Chain Planner -> Supply Chain Plan -> Workbench
   ==>Once the organization is selected, go Tools > Preferences
   ==>Set a Default Job Class
c. Check whether the accounting period is open or not.

14. APPENDIX A

WIP_JOB_SCHEDULE_INTERFACE

The following table lists the columns in the WIP_JOB_SCHEDULE_ INTERFACE table and provides their load/update type and validation information.

NOTE: R—Required; **O/D**—Optional/Derived if NULL; **O**—Optional; **D/I**—Derived or Ignored

LOAD_TYPE ==>	1	2	3	4
COLUMN NAME				
ALLOW_EXPLOSION	O/D	--	O/D	O/D
ALTERNATE_BOM_DESIGNATOR	O	D/I	D/I	O
ALTERNATE_ROUTING_DESIGNATOR	O	D/I	D/I	O
ATTRIBUTE1-15	O	O	O	O
ATTRIBUTE_CATEGORY	O	O	O	O
BOM_REFERENCE_ID	D/I	D/I	D/I	O
BOM_REVISION	O/D	O/D	D/I	O/D
BOM_REVISION_DATE	O/D	O/D	D/I	O/D
BUILD_SEQUENCE	O	D/I	O	O
CLASS_CODE	O/D	D/I	D/I	R
COMPLETION_LOCATOR_ID	O/D	D/I	D/I	O/D
COMPLETION_LOCATOR_SEGMENTS	O/D	D/I	D/I	O/D
COMPLETION_SUBINVENTORY	O/D	D/I	D/I	O/D
CREATED_BY	R	R	R	R
CREATED_BY_NAME	R	R	R	R
CREATION_DATE	R	R	R	R
DAILY_PRODUCTION_RATE	D/I	R	D/I	D/I
DEMAND_CLASS	O	O	D/I	O
DESCRIPTION	O/D	O/D	O	O/D
DUE_DATE	O	--	O	O
DUE_DATE_PENALTY	O	--	O	O
DUE_DATE_TOLERANCE	O	--	O	O
FIRM_PLANNED_FLAG	O/D	O/D	O	O/D
FIRST_UNIT_COMPLETION_DATE	O/D	O/D	O	O/D
FIRST_UNIT_START_DATE	O/D	O/D	O	O/D
GROUP_ID	R	R	R	R
HEADER_ID	R	R	R	R
INTERFACE_ID	D/I	D/I	D/I	D/I
JOB_NAME	O/D	D/I	D/I	O/D
KANBAN_CARD_ID	O	O	O	O
LAST_UNIT_COMPLETION_DATE	R	R	O	R
LAST_UNIT_START_DATE	R	R	O	R
LAST_UPDATED_BY	R	R	R	R
LAST_UPDATED_BY_NAME	R	R	R	R
LAST_UPDATE_DATE	R	R	R	R
LAST_UPDATE_LOGIN	D/I	D/I	D/I	D/I
LINE_CODE	O	R	O	O
LINE_ID	O	R	O	O
LOAD_TYPE	R	R	R	R
LOT_NUMBER	O/D	D/I	O	O/D
NET_QUANTITY	O	D/I	O	O
OVERCOMPLETION_TOLERANCE_TYPE	O	O	O	O
OVERCOMPLETION_TOLERANCE_VALUE	O	O	O	O
PRIMARY_ITEM_ID	R	R	D/I	O
PRIORITY	O	--	O	O
PROCESSING_WORK_DAYS	D/I	R	D/I	D/I
PROCESS_PHASE	R	R	R	R
PROCESS_STATUS	R	R	R	R
PROGRAM_APPLICATION_ID	D/I	D/I	D/I	D/I
PROGRAM_ID	D/I	D/I	D/I	D/I
PROGRAM_UPDATE_DATE	D/I	D/I	D/I	D/I
PROJECT_ID	O	D/I	O/D	O
PROJECT_NUMBER	O	D/I	O/D	O
REPETITIVE_SCHEDULE_ID	D/I	D/I	D/I	D/I
REQUEST_ID	D/I	D/I	D/I	D/I
ROUTING_REFERENCE_ID	D/I	D/I	D/I	O

ROUTING_REVISION	O/D	O/D	D/I	O/D
ROUTING_REVISION_DATE	O/D	O/D	D/I	O/D
SCHEDULE_GROUP_ID	O	D/I	O	O
SCHEDULE_GROUP_NAME	O	D/I	O	O
SCHEDULING_METHOD	O	D/I	O	O
SOURCE_CODE	O	O	O	O
SOURCE_LINE_ID	O	O	O	O
START_QUANTITY	O	D/I	O	O
STATUS_TYPE	O/D	D/I	O	O/D
TASK_ID	O/D	O	O/D	O
TASK_NUMBER	O/D	O	O/D	O
WIP_ENTITY_ID	D/I	D/I	R	D/I
WIP_ENTITY_NAME	D/I	D/I	R	D/I
WIP_SUPPLY_TYPE	O/D	D/I	D/I	O/D

15. APPENDIX B

WIP_JOB_DTLS_INTERFACE

The following table lists the columns in the WIP_JOB_DTLS_INTERFACE table and provides their load/update type and validation information.

Column Name	Add Operation	Change Operation	Add Component	Change Component	Add Resource	Change Resource	Scheduled
ACTIVITY_ID	N	N	N	N	O	O	N
APPLIED_RESOURCE_UNITS	N	N	N	N	N	N	N
APPLIED_RESOURCE_VALUE	N	N	N	N	N	N	N
ASSIGNED_UNITS	N	N	N	N	R	O	N
ATTRIBUTE_CATEGORY	O	O	O	O	O	O	N
ATTRIBUTE1-15	O	O	O	O	O	O	N
AUTO_REQUEST_MATERIAL	N	N	N	N	N	N	N
AUTOCHARGE_TYPE	N	N	N	N	R	O	N
BACKFLUSH_FLAG	R	O	N	N	N	N	N
BASIS_TYPE	N	N	N	N	R	O	N
COMPLETION_DATE	N	N	N	N	R	O	N
COMPONENT_YIELD_FACTOR	N	N	O	O	N	N	N
COUNT_POINT_TYPE	R	O	N	N	N	N	N
CREATED_BY	R	R	R	R	R	R	R
CREATION_DATE	R	R	R	R	R	R	R
DATE_REQUIRED	N	N	R	O	N	N	N
DEPARTMENT_ID	R	O	O	O	N	N	N
DESCRIPTION	O	O	O	O	O	O	O
FIRST_UNIT_COMPLETION_DATE	R	O	N	N	N	N	N
FIRST_UNIT_START_DATE	R	O	N	N	N	N	N
GROUP_ID	R	R	R	R	R	R	R
INTERFACE_ID	N	N	N	N	N	N	N
INVENTORY_ITEM_ID_NEW	N	N	R	O	N	N	N
INVENTORY_ITEM_ID_OLD	N	N	N	R	N	N	N
ITEM_SEGMENTS	O	O	O	O	O	O	O
LAST_UNIT_COMPLETION_DATE	R	O	N	N	N	N	N
LAST_UNIT_START_DATE	R	O	N	N	N	N	N
LAST_UPDATE_DATE	R	R	R	R	R	R	R
LAST_UPDATE_LOGIN	R	N	N	N	N	N	N
LAST_UPDATED_BY	R	R	R	R	R	R	R
LOAD_TYPE	3	3	2	2	1	1	4
MINIMUM_TRANSFER_QUANTITY	R	O	N	N	N	N	N
MPS_DATE_REQUIRED	N	N	O	O	N	N	N
MPS_REQUIRED_QUANTITY	N	N	O	O	N	N	N
MRP_NET_FLAG	N	N	R	O	N	N	N
OPERATION_SEQ_NUM	R	R	R	R	R	R	R
ORGANIZATION_ID	N	N	N	N	N	N	N
PARENT_HEADER_ID	R	R	R	R	R	R	R
PROCESS_PHASE	R	R	R	R	R	R	R
PROCESS_STATUS	R	R	R	R	R	R	R
PROGRAM_APPLICATION_ID	N	N	N	N	N	N	N
PROGRAM_ID	N	N	N	N	N	N	N
PROGRAM_UPDATE_DATE	N	N	N	N	N	N	N

QUANTITY_ISSUED	N	N	R	O	N	N	N
QUANTITY_PER_ASSEMBLY	N	N	R	O	N	N	N
REQUEST_ID	N	N	N	N	N	N	N
REQUIRED_QUANTITY	N	N	R	O	N	N	N
RESOURCE_ID_NEW	N	N	N	N	R	O	N
RESOURCE_ID_OLD	N	N	N	N	N	R	N
RESOURCE_INSTANCE_ID	N	N	N	N	N	N	N
RESOURCE_SEQ_NUM	N	N	N	N	R	R	R
SCHEDULE_SEQ_NUM	N	N	N	N	N	N	N
SCHEDULED_FLAG	N	N	N	N	R	O	N
STANDARD_OPERATION_ID	O	O	N	N	N	N	N
STANDARD_RATE_FLAG	N	N	N	N	R	O	N
START_DATE	N	N	N	N	R	O	N
SUBSTITUTION_TYPE	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3
SUPPLY_LOCATOR_ID	N	N	O	O	N	N	N
SUPPLY_SUBINVENTORY	N	N	O	O	N	N	N
USAGE_BLOCK	N	N	N	N	N	N	R
UOM_CODE	N	N	N	N	R	O	N
USAGE_RATE_OR_AMOUNT	N	N	N	N	R	O	N
WIP_ENTITY_ID	N	N	N	N	N	N	N
WIP_SUPPLY_TYPE	N	N	R	O	N	N	N

15. References

1. Work Order Interface from Oracle® Supply Chain Management APIs and Open Interfaces Guide. You can download the same from http://docs.oracle.com/cd/B40089_10/current/acrobat/120scmapi.pdf.
2. WICMLP: WIP Mass Load: Error: 'Must only provide one of the p_startDate and p_endDate for routing based scheduling' (Doc ID 1439531.1)
3. Changing One Column In One Row Through WIP Mass Load Resulting In Many MLOG Changes (Doc ID 1548444.1)
4. WIP Mass Loads Explodes BOM when Releasing Job Even Material Requirements Were Changed in Unreleased Status (Doc ID 1502423.1)
5. Wip Mass Load Does Not Create Jobs In Wip. Error In Log File: Cannot Read Value For Profile Option (Doc ID 1101695.1)
6. Wip Mass Load Program Error Cannot Read Value For Profile Option RPM_CONC_PROCESS_ID In Routine &ROUTINE (Doc ID 1353766.1)
7. WIP Mass Load (WICUMI) error APP-25039 Unable to Find Open Period (Doc ID 111020.1)
8. Wip Mass Load Does Not Pick The Wip Accounting Class (Doc ID 165226.1)
9. WICMLP/WICMLX: WIP Mass Load Error: System Cannot Add Resource Or Material (Doc ID 736758.1)
10. Discrete Job Completion Date is Moved from Monday to Saturday (Doc ID 1498069.1)
11. Clarification on Profile Option WIP: ALIGN JOB DATES WITH SHIFT CALENDAR (Doc ID 1580391.1)