Oracle® Process Manufacturing

Production Management and Process Operations Control APIs User's Guide

Release 11*i*

July 2000

Part No. A86086-01

ORACLE

Oracle Process Manufacturing Production Management and Process Operation Control APIs User's Guide, Release 11*i*

Part No. A86086-01

Primary Author: Michele-Andrea Fields

Contributing Authors: Tom Muratore, Glenn Ruhl

Contents

3 Process Operations Control APIs

Send Us Your Comments

Oracle Process Manufacturing Production Management and Process Operation Control APIs User's Guide Release 11*i*

Part No. A86086-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available). You can send comments to us in the following ways:

- FAX: 650-506-7200 Attn: Oracle Process Manufacturing
- Postal service:
  Oracle Corporation
  Oracle Process Manufacturing
  500 Oracle Parkway
  Redwood City, CA 94065
  U.S.A.
- Electronic mail message to appsdoc@us.oracle.com

If you would like a reply, please give your name, address, and telephone number below.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Welcome to the *Oracle Process Manufacturing Production Mangement and Process Operations Control APIs User's Guide*. This user's guide includes the information you need to work with the Oracle Process Manufacturing (OPM) Application Programming Interfaces (APIs).

This preface explains how this user's guide is organized and introduces other sources of information that can help you.

Intended Audience

This guide assumes that you have working knowledge of your business area's processes and tools. It also assumes that you are familiar with APIs and OPM Production Management and Process Operation Control. If you have never used OPM, we suggest you attend one or more of the Oracle Process Manufacturing training classes available through Oracle World Wide Education.

Information Sources

You can choose from many sources of information, including documentation, training, and support services to increase your knowledge and understanding.

Online Documentation

Oracle Applications documentation is available on CD-ROM, except for technical reference manuals. User's guides are available in HTML format and on paper. Technical reference manuals are available on paper only. Other documentation is available on paper and sometimes in PDF format.

The content of the documentation remains the same from format to format. Slight formatting differences could occur due to publication standards, but such differences do not affect content. For example, page numbers are included on paper, but are not included in HTML.

The HTML documentation is available from all Oracle Applications windows. Each window is programmed to start your web browser and open a specific, context-sensitive section. Once any section of the HTML documentation is open, you can navigate freely throughout all Oracle Applications documentation.

Related Documents

Oracle Process Manufacturing shares business and setup information with other Oracle products. You may find the following Oracle Applications user's guides useful:

- *Oracle Applications User's Guide Release 11i*
- *Oracle Application's Flexfields Guide Release 11i*
- *Oracle Workflow User Guide*
- *Oracle Applications System Administrator's Guide Release 11i*
- *Oracle General Ledger User's Guide Release 11i.*
- *Oracle Payables User's Guide Release 11i*
- *Oracle Receivables User's Guide Release 11i*
- *Oracle Human Resources North American User's Guide Release 11i*
- *Oracle Purchasing User's Guide Release 11i*

Oracle Process Manufacturing Guides

The following is a list of documentation in each product group for OPM Release 11i:

Financials

- *Oracle Process Manufacturing Accounting Setup User's Guide*
- *Oracle Process Manufacturing Cost Management User's Guide*
- *Oracle Process Manufacturing Manufacturing Accounting Controller User's Guide*
- *Oracle Process Manufacturing and Oracle Financials Integration User's Guide*

Inventory Control

- *Oracle Process Manufacturing EC Intrastat Reporting User's Guide*

- *Oracle Process Manufacturing Inventory Management User's Guide*
- *Oracle Process Manufacturing Physical Inventory User's Guide*

Logistics

- *Oracle Process Manufacturing Order Fulfillment User's Guide*
- *Oracle Process Manufacturing Purchase Management User's Guide*

Process Execution

- *Oracle Process Manufacturing Process Operation Control User's Guide*
- *Oracle Process Manufacturing Production Management User's Guide*

Process Planning

- *Oracle Process Manufacturing Capacity Planning User's Guide*
- *Oracle Process Manufacturing Capacity Planning with RHYTHM Factory Planner User's Guide*
- *Oracle Process Manufacturing MPS/MRP and Forecasting User's Guide*

Product Development

- *Oracle Process Manufacturing Formula Management User's Guide*
- *Oracle Process Manufacturing Laboratory Management User's Guide*
- *Oracle Process Manufacturing Quality Management User's Guide*

Regulatory

- *Oracle Process Manufacturing Regulatory Management User's Guide*

System Administration and Technical Reference

- *Oracle Process Manufacturing Implementation Guide*
- *Oracle Process Manufacturing System Administration User's Guide*
- Oracle Process Manufacturing Technical Reference Manuals

Training

Oracle offers a complete set of formal training courses to help you master Oracle Process Manufacturing and reach full productivity quickly. We organize these courses into functional learning paths, so you take only those courses appropriate to your area of responsibility.

You have a choice of educational environments. You can attend courses offered by Oracle Education Services at any one of our many Education Centers, or you can arrange for our trainers to teach at your facility. In addition, Oracle Training professionals can tailor standard courses or develop custom courses to meet your needs. For example, you may want to use your organization's structure, terminology, and data as examples in a customized training session delivered at your own facility.

Conventions

The following conventions are used in this guide:

Pending/Completed Transactions

Discussions about processing transactions that use the words pending and completed refer to the status of a transaction. Pending and completed do not refer to the database tables that are updated as a result of transactions (for example, some completed transactions are stored in the Pending Transactions table).

Use of the Word Character

The word character means an alphanumeric character. Characters that are numeric or alphabetic only are referenced specifically. Depending on your system security profile, you many not have access to all of the windows and functions described in this guide. If you do not see a menu option described in this guide, and you want access to it, contact your System Administrator.

Do Not Use Database Tools to Modify Oracle Applications Data

Oracle Applications tables are interrelated. As a result, any change you make using Oracle Applications can update many tables at once. If you modify the Oracle Applications data using anything other than Oracle Applications, you could change a row in one table without making corresponding changes in related tables. If your tables are not synchronized with each other, you risk retrieving erroneous information and receiving unpredictable results throughout Oracle Applications.

When you use Oracle Applications to modify your data, Oracle Applications automatically checks that your changes are valid. Oracle Applications also track who changes information. If you enter information into database tables using database tools, you could store invalid information. You also lose the ability to track who has changed your information because SQL*Plus and other database tools do not keep a record of changes.

Consequently, we strongly recommend that you never use SQL*Plus or any other tool to modify Oracle Applications data unless otherwise instructed by Oracle Support Services.

About Oracle

Oracle Corporation develops and markets an integrated line of software products for database management, applications development, decision support, and office automation, as well as Oracle Applications, an integrated suite of more than 45 software modules for financial management, supply chain management, manufacturing, project systems, human resources, sales, and service management.

Oracle Products are available for mainframes, minicomputers, personal computers, network computers, and personal digital assistants, allowing organizations to integrate different computers, different operating systems, different networks, and even different database management systems, into a single, unified computing, and information resource.

Oracle is the world's leading supplier of software for information management, and the world's second largest software company. Oracle offers its database, tools, and applications products, along with related consulting, education, and support services in over 140 countries around the world.

Thank You

Thank you for choosing Oracle Process Manufacturing and this user's guide.

We value your comments and feedback. At the beginning of this guide is a Reader's Comment Form that you can use to explain what you like or dislike about this user's guide. Mail your comments to the following address or call us directly at 650-506-7000.

Oracle Applications Documentation Manager
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Or, send an electronic mail message to appsdoc@us.oracle.com

1

API Overview

This document describes the Application Program Interfaces (APIs) that support external interfaces to Oracle Process Manufacturing (OPM) Production Management (PM) and Process Operations Control (POC) applications. The topics discussed in this chapter are:

- Introduction to the PM/POC APIs
  - Support Policy
  - Overview of PM/POC APIs
  - Bill of Materials
- Compiling and Linking
  - Compiling Example
- Calling the API Interface Code
  - Calling the API Code Examples
  - User_id Discussion

Also provided is an API flow diagram.

Introduction to the PM/POC APIs

Much of the application is divided into module-specific libraries to allow the user to link OPM functionality into their own programs.

The interfaces can make use of the standard functionality and logic implemented in the OPM PM & POC modules. The APIs provide "hooks" into which customers can interface their own C programs. The standard functionality that can be executed from the APIs include:

PM	POC
Create Batch	Release Step
Release Batch	Certify Step
Reschedule Batch	Close Step
Re route Batch	Post Actual Resource Transaction
Close Batch	Post Incremental Resource Transaction
Cancel Batch	Post Timed Resource Transaction
Manually Release Batch Line (Obsolete)	Start Resource usage
Insert Allocation	End Resource Usage
Insert Detail	
Partial Certification	
Certify Batch	

Support Policy

Supported means:

- Oracle will provide objects and libraries needed to write interfaces to the APIs and the documentation for their use.
- Oracle will ensure that the APIs function as designed.
- Oracle will not support customer generated programs that use the APIs.

Overview of PM/POC APIs

The PM and POC APIs are currently written in C programming language. To make use of these API's, you must code your interface in C and include the appropriate header files. Your program will also be responsible for connecting to a database before calling an API function and disconnecting upon return. You may also choose to write log files before calling and after returning from a function. Each function will return an error code < 0 (less than zero) if there is a problem during execution of that call. A return code of 1 denotes success.

Below is a list of the header files that must be included in your interface program to make use of the API functions. For a description of these header files, refer to the *Bill of Materials* section.

- stdio.h
- afpub.h
- sysize.h
- systdsql.h
- pmapi.h (For PM API's) and/or pcapi.h (For POC API's)

The standard construct used and recommended for includes is:

```
#ifndef  AFPUB
#include <afpub.h>
#endif
#ifndef  PMAPI
#include <pmapi.h>
#endif
etc...
```

You will also need to put the following line in your code before any includes:

```
#define CONST_DATA const
```

CONST_DATA is a predefined variable that is needed to compile and therefore must be inserted in your C code interface.

Bill of Materials

For the PM and POC APIs, the following is a list of header files that are delivered with OPM. These must be on your system for your C interface to compile and link properly.

File Name	Description
pcapi.h	OPM POC function prototypes
pmapi.h	OPM PM function prototypes
sysize.h	OPM column size header
systdsql.h	OPM DBDATE definition

Compiling and Linking

File libgme.a is the library which contains all the API code for Process Execution. This is the library that is linked in by your Make script. Other Library files which are necessary and must be linked in are libgmi.a, libgma.a, libgmd.a and libfnd.a. These files come standard with a release or patch.

When using the CC command, you must use the following two switches in addition to all other arguments on the command. This will eliminate inclusion of header files which do not get delivered with the standard software. The -D switch defines the variable that follows it during the execution of the CC command.

cc … -DS -DAFSTD -DGMAO -DSUN_OS5 ...

Compiling Example

```
cc -Aa -g -DS -DAFSTD -DGMAO -DSUN_OS5 -I(directory where the OPM header files are)
-I($FND_TOP)/include -I($ORACLE_HOME)/precomp/public
-I/($ORACLE_HOME)/rdbms/demo  -c myprog.c
```

You should compile and link your API code and generate a standalone executable with a name of your choice. For the link you will need additional standard Oracle libraries and objects in addition to the OPM libraries mentioned in the previous section. For details on the standard Oracle libraries and objects, refer to the *Oracle ProC/ESQL User's Guide*.

Calling the API Interface Code

The following are snippets from our internal C interface code (referred to as pmmain from here on) and are used to test the API code. You will probably want to set up some pre-defined values which represent the particular API action being executed.

```
#define PM_CREATE_BATCH      1
#define PM_RELEASE_BATCH     2
#define PM_RESCHEDULE_BATCH  3
#define PM_REROUTE_BATCH     4
#define PM_CANCEL_BATCH      5
#define PM_CLOSE_BATCH       6
#define PM_MANUAL_RELEASE    7
#define PM_CERTIFY_BATCH     8
#define PM_PARTIAL_CERT      9
```

Our program is defined with a main. We call the APIs directly from a command line since we compile and link a standalone executable. We use a variable argument list and parse out the parameter values which represent different data values depending on which API is being executed.

```
Int main (argc, argv)
int argc;
char *argv[];
```

If you choose not to put a Main into your code, you could compile it as a shared object (.so) and register it in PL/SQL and make direct calls to the API interface code.

Shared Objects are a new Oracle 8 feature. An external procedure call provides the ability for a PL/SQL procedure and/or function to utilize an external C program's functionality and be able to interrogate the return code. for addition information, refer to the Oracle Press *Oracle 8 PL/SQL Programming Guide*.

Variables

The following is a representation of variables you will probably need to use. You can name the variables whatever you choose (ret could be return_code, etc). The types are Oracle types which generally correspond to Ansi C types and must be used.

sword = int sb4 = long text = char

```
sword   ret, i, action;
 text * line_no,
      * line_type,
      * accept_shortage,
      * event_type,
      * accept_zero,
      * accept_incomplete,
      * alloc_type,
      * batchstep_no,
      * qty_type,
      * adjust_cert,
      * check_inv,
      * session_no,
      * fmeff_id,
      * formula_id,
      * routing_id,
      * batch_id,
      * event_id,
      * item_qty,
      * trans_qty,
      * trans_qty2,
      * qty,
      * usage,
      * user_id,
      * item_um,
      * start_date,
      * plan_start_date,
      * expct_cmplt_date,
      * due_date,
      * batch_close_date,
      * batch_no,
      * plant_code,
      * resource,
      * activity,
      * whse_code,
      * lot_no,
      * sublot_no,
      * location,
        new_batch_no[BATCH_NO_SIZE],
        message[MAX_MESSAGE];
```

Sample call 1:pmmain user/pwd@dbname 8 "0" "PM1" "000154" "2036" 1

The code which takes this call and parses it accordingly looks like;

```
/*  Retrieve the 'action' from the parameter list  */
  action = (text *)argv[2];

If (action == PM_CERTIFY_BATCH)  /* action 8 */
{
    accept_zero = (text *)argv[3];
    plant_code = (text *)argv[4];
    batch_no = (text *)argv[5];
    user_id = (text *)argv[6];
    accept_incomplete = (text *)argv[7];
}
else (if action == PM_RELEASE_BATCH)
{
    plant_code = (text *)argv[3];
    batch_no = (text *)argv[4];
    user_id = (text *)argv[5];
    start_date = (text *)argv[6];
    accept_shortage = (text *)argv[7];
}
```

ETC…

Sample call 2

```
pmmain user/pwd@dbname "1" "PM1" "2036" "26" "10" "LB" "0"
                       "09-MAR-1999 12:01:30" "09-MAR-1999 12:01:30"
                       "09-MAR-1999 12:01:30" ""

         plant_code = (text *)argv[2];
         user_id = (text *)argv[3];
         fmeff_id = (text *)argv[4];
         item_qty = (text *)argv[5];
         item_um = (text *)argv[6];
         batch_type = (text *)argv[7];
         plan_start_date = (text *)argv[8];
         expct_cmplt_date = (text *)argv[9];
         due_date = (text *)argv[10];
         strcpy((char *)new_batch_no, (char *)argv[11]);
```

Calling the API Code Examples

This section details how we call the API code within the interface code. The purpose of this is to explain how we call pmmain (the API interface) and in turn it calls a standard PM API. We are using certify batch and release batch as examples.

The first call is to the interface code itself, to certify a batch. The order of these parameters is totally dictated by the interface program. These parameters can be accepted in a totally different order in your own code.

```
pmmain user/pwd@dbname 8 "0" "PM1" "000154" "2036" "1"
```

The first two parameters are exactly the same in all our interface code. That is "user/pwd@dbname" is always first. Pmmain is not a parameter but rather the executable name being run. It is a user defined executable name in the make script. The second parameter, "8", is an internal value we use to represent an action, "certify" in this case, to our interface code. This could be changed in your own interface code to be clearer as to what a particular code represents. The rest of the parameters pertain to each individual API and are documented in the following pages.

It is not critical how the data values get into your customized interface. What is dictated is the order in which those same data values get passed into the standard API calls.

Certify Batch Example

User Defined Executable Name	User, Password and DBname	Internal Action Code	Parameter Values Necessary to Run the Actual API
pmmain	user/pwd@dbname	8	"0" "PM1" "000154 "2036" "1"

Starting at the real parameter 1 after the action code:

Parameter 1	accept_zero	"0"
Parameter 2	plant_code	"PM1"
Parameter 3	batch_no	"000154"
Parameter 4	user_id	"2036"
Parameter 5	accept_incomplete	"1"

Call to actual certify batch API:

```
ret = pm_certify_batch_api (accept_zero, plant_code, batch_no, user_id,
                            accept_incomplete);
```

The parameters are in the same sequence within the call to certify_batch_api. It is recommended that your API routines are coded to accept the parameters in the same order that they are used to call the API function.

Release Batch Example

User Defined Executable Name	User, Password and DBname	Internal Action Code	Parameter Values Necessary to Run the Actual API
pmmain	user/pwd@dbname	2	"PM1" "000154" "2036" "11-MAR-1999 12:00:00" "1"

Starting at the real parameter 1 after the action code:

Parameter 1	plant_code	"PM1"
Parameter 2	batch_no	"000154"
Parameter 3	user_id	"2036"
Parameter 4	start_date	"11-MAR-1999 12:00:00"
Parameter 5	accept_shortage	"1"

Call to actual release batch API:

```
ret = pm_release_batch_api (plant_code, batch_no,start_date, accept_shortage,
                            user_id, batch_id);
```

Parameters 3 through 5 for the interface call are not in the same sequence with the call to release_batch. In both scenarios the code will work, although it is recommended that your code be consistent and clear.

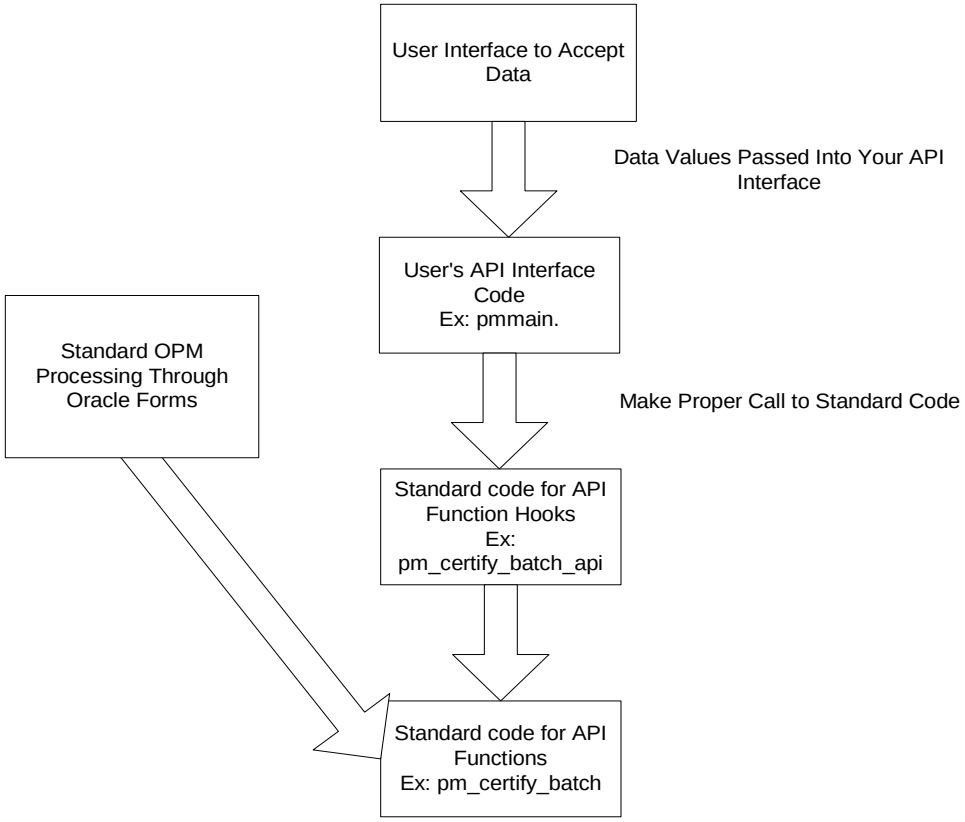
User_id Discussion

Currently, the user_id parameter in our interface code is accepting a string. That value is in turn passed into our API functions (ex. certify_batch) also as a string. At the lowest level of the code which deals with the database directly, this value is stored as a numeric as it's defined in the tables.

In your interface code, you can implement this as it's done in the examples or you could do things differently. You may choose to accept user_name and fetch the user_id from table fnd_user based on the user_name. The API's do validate the user_id to make sure it is a valid entry in fnd_user, regardless.

The important thing to understand is that our code currently accepts the user_id as a string. If you fetch the user_id as previously suggested, it will have to be converted to a string format before calling the API code.

Below is a diagram of the API flow:



PM/POC API Flow Overview

2

Production Management APIs

The following pages detail the APIs currently available for the OPM Production Management (PM) module.

All parameter values should be passed in as character strings to all API's in this manual. The individual parameter lists display the data type that a given character string will be converted to internally.

Note: Oracle Number is a proprietary data type that in essence replaces a double type.

Create Batch

Synopsis

```
sword ret;
ret = pm_create_batch_api (<plant_code>,<user_id>, <fmeff_id>
                           <effective_qty>, <effective_um>, <batch_type>,
                           <plan_start_date>, <expct_cmplt_date>, <due_date>,
                           <batch_no>);

if (ret < 1)
    <error>
```

Description

The Create Batch API may be used to create Batches or Firm Planned Orders. The Save Batch API is obsolete as of 11i. Create Batch encapsulates both creation and save of the batch, whereas in older versions a call to Create Batch required an additional call to Save Batch.

Input Parameters

Parameter	Type string will be converted to	Valid Values
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
user_id	character	fnd_user.user_id
fmeff_id	long	fm_form_eff.fmeff_id
eff_item_qty	Oracle Number	number >= 0
eff_item_um	character	sy_uoms_mst. um_code
batch_type	integer	0 = Batch 10 = FPO
plan_start_date	date	valid date less than or equal to expct_cmplt_date
expct_cmplt_date	date	valid date greater than or equal to plan_start_date
due_date	date	any valid date
batch_no	character	formatted batch number for maual document sequencing plants or NULL for auto doc sequencing plants

Release Batch

Synopsis

```
sword ret;
ret = pm_release_batch_api (<plant_code>, <batch_no>, <start_date>,
                            <accept_shortage>, <user_id>, <batch_id>);
if (ret < 1)
   <error>
```

Description

The Release Batch API loads the specified Batch into memory, validates the batch and releases it. A batch must be in a Pending state to be released by this API. Inventory Shortage Checking will be performed based upon system configuration. Inventory Shortages will either be ignored or they will cause the Release of the Batch to fail (depending upon the value passed in for “accept_shortage”). Releasing the Batch causes the following to occur:

- Batch Status will be changed to WIP
- Actual Start Date will be changed to the start_date passed in or the current system date/time if the value is not passed in
- Automatically Released Ingredients will be consumed from Inventory
- Transactions for Automatically Released Ingredients will be completed
- Actual Quantities will be assigned for the Automatically Released Ingredients

Input Parameters

Parameter	Type string will be converted to	Valid Values
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
start_date	date	valid date
accept_shortage	integer	0 or 1
user_id	character	fnd_user.user_id
batch_id	long	not used for API calls, pass in 0

Partial Certification

Synopsis

```
sword ret;
ret = pm_partial_cert_api (<dtl_index>, <adjust_cert>, <qty>, <qty_type>,
                           <line_no>, <line_type>,<batch_no>,<user_id>,
                           <plant_code>);

if (ret < 1)
    <error>
```

Description

The Partial Certification API is used to record incremental yield or consumption of an item in a WIP or Certified batch. You can pass in either an incremental quantity, a new actual quantity, or a new percent of plan. In addition, the Partial Certification API will backflush other items in the batch which have a release type of Incremental. For lot or location controlled items, the new actual quantities entered or calculated through backflushing will consume from (or yield into) existing pending allocations for those items in the batch. If there are any unallocated actual quantities after backflushing, partial certification will fail.

Input Parameters

Parameter	Type string will be converted to	Valid Values
dtl_index	integer	always "-1". Not for API use.
adjust_cert	integer	adjust certification if batch is certified 0 = no 1 = yes
qty	Oracle Number	quantity which represents 3 possible entities
qty_type	integer	0 = incremental 1 = % complete 2 = new actual
line_no	integer	line no which will drive partial certification
line_type	integer	- 1 = ingredient 1 = product 2 = byproduct
batch_no	character	pm_btch_hdr.batch_no for given plant_code
user_id	character	fnd_user.user_id
plant_code	character	sy_orgn_mst.orgn_code where plant code = 1

Reschedule Batch

Synopsis

```
sword ret;
ret = pm_reschedule_batch_api (<plan_start_date>, <expct_cmplt_date>,
                               <batch_type>, <plant_code>, <batch_no>,
                               <user_id>);
if (ret < 1)
    <error>
```

Description

The Reschedule Batch API will modify the Planned Start and Expected Completion Dates of a Pending Batch or the Expected Completion Date of a WIP Batch. When the Planned Start Date is modified all Ingredient Transactions will be similarly modified. That is to say that the difference between the original Planned Start Date and the new Planned Start Date is applied to every Ingredient Transaction. Modifying the Expected Completion Dates affects Product and Byproduct Transactions in the same way.

Input Parameters

Parameter	Type string will be converted to	Valid Values
plan_start_date	date	valid date less than or equal to expct_cmplt_date
expct_cmplt_date	date	valid date greater than or equal to plan_start_date
batch_type	integer	0 = batch 10 = FPO
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
user_id	character	fnd_user.user_id

Reroute Batch

Synopsis

```
sword ret;
ret = pm_reroute_batch_api (<batch_type>, <plant_code>, <batch_no>,
                            <user_id>, <fmeff_id>);

if (ret < 1)
   <error>
```

Description

The Reroute Batch API assigns a new Routing to the given Batch. The new Routing must be effective for the same Formula that was used to create the Batch. If POC data exists for the original Batch Routing, Rerouting the Batch will delete all associated POC data. POC data will be recreated for the new Routing.

Input Parameters

Parameter	Type string will be converted to	Valid Values
batch_type	integer	0 = batch 10 = FPO
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
user_id	character	fnd_user.user_id
fmeff_id	long	fm_form_eff.fmeff_id

Close Batch

Synopsis

```
sword ret;
ret = pm_close_batch_api (<plant_code>, <batch_no>, <batch_close_date>,
                          <user_id>);

if (ret < 1)
    <error>
```

Description

The Close Batch API updates the Batch Status to Closed. No further modifications can be made to a Batch once it has been Closed. Only Certified Batches can be Closed. Closing the Batch causes the following to occur:

- Batch Status will be changed to Closed
- Batch Close Date will be changed to the closed_date passed in or the current system date/time if the value is not passed in

Input Parameters

Parameter	Type string will be converted to	Valid Values
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
batch_close_date	date	valid date
user_id	character	fnd_user.user_id

Cancel Batch

Synopsis

```
sword ret;
ret = pm_cancel_batch_api (<plant_code>, <batch_no>, <user_id>);

if (ret < 1)
  <error>
```

Description

The Cancel Batch API updates the Batch Status to Cancelled. All Transactions (Product, Byproduct and Ingredient) will be backed out of the system. No further modifications can be made to a Batch once it has been Cancelled. Only Pending Batches can be Cancelled.

Input Parameters

Parameter	Type string will be converted to	Valid Values
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
user_id	character	fnd_user.user_id

Manually Release Material (Obsolete in 11.5.2)

Synopsis

```
sword ret;
ret = pm_manual_release_api (<plant_code>, <batch_no>, <line_no>, <line_type>,
                             <user_id>, <trans_qty>, <trans_qty2>, <whse_code>,
                             <lot_no>, <sublot_no>, <location>);
if (ret < 1)
  <error>
```

Description

The Manual Release API is used to charge material to a batch or yield product. The Batch must be in a WIP or Certified state and the material detail line must have a Release Type of Manual or Incremental.

Note: Pm_manual_release is obsolete as of 11.5.2.

Pm_insert_allocation_api replaces this API for 11.5.2.

Input Parameters

Parameter	Type string will be converted to	Valid Values
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
line_no	integer	pm_matl_dtl.line_no for given plant_code and batch_no
line_type	integer	- 1 = ingredient 1 = product 2 = byproduct
user_id	character	fnd_user.user_id
trans_qty	Oracle Number	greater than zero if line_type = 1 or 2; less than zero if line_type = -1
trans_qty2	Oracle Number	same sign as trans_qty; non-zero if Item is dual UM controlled
whse_code	character	ic_whse_mst.whse_code
lot_no	character	ic_lots_mst.lot_no where item_id = pm_matl_dtl.item_id for given plant_code, batch_no and line_no if Item is lot-controlled; null if Item is not lot-controlled
sublot_no	character	ic_lots_mst.sublot_no where item_id = pm_matl_dtl.item_id for given plant_code, batch_no and line_no and lot_no = given lot_no if Item is sublot-controlled; null if Item is not sublot-controlled
location	character	ic_loct_mst.location where whse_code = given whse_code if Item is location-controlled; null if Item is not location-controlled

Insert Allocation

The pm_insert_allocation_api allows the user to insert an allocation against any detail line of any line_type for pending, wip and certified batches. This new functionality is an extension of what pm_manual_release did. Pm_manual_release allowed insertion for manual or incremental release detail lines in wip and certified batches only. It also assumed that it was a completed transaction.

To implement this new functionality, the following was done:

- pm_manual_release_api is now obsolete and the API is now called pm_insert_allocation_api.
- An additional parameter was added, completed indicator, which gives the user control over the status of the allocation in a wip state for manual/incremental release items (see Completed Indicator section).
- Additional validations were put into place for the new parameter.

Completed Indicator

The valid values for completed indicator are zero (pending) and one (completed). The API will validate the value passed in for this set of valid values and will return an error if it is invalid. This value will only be used for manual or incremental detail lines in a wip batch. This is the only time (even when using the forms interface) that the user can enter an allocation and determine if it is complete or pending. The rest of the scenarios are defined as follows:

- All allocations in a pending batch are pending (0).
- All allocations in a certified batch are complete (1).

In a wip batch

- All auto release ingredient transactions are complete.
- All auto release product/byproduct transactions are pending

  For manual/incremental release detail lines use completed indicator value passed into API.

> **Note:** This API is for 11i only. Pm_manual_release is obsolete as of 11.5.2

Synopsis

```
sword ret;
ret = pm_insert_allocation_api (<plant_code>, <batch_no>, <line_no>,
                                <line_type>, <user_id>, <trans_qty>,
                                <trans_qty2>, <whse_code>, <lot_no>,
                                <sublot_no>, <location>, <completed_ind>);
if (ret < 1)
  <error>
```

Description

The Insert Allocation API allows the user to create allocations, pending or complete, for a given detail line in a given batch. The Batch can be in a Pending, WIP or Certified state and the material detail line can have any Release Type. This API supersedes the Manual Release API. It is more robust and has made the Manual Release API obsolete. For users who wish to use the Manual Release API functionality, simply call the new function and pass in the additional parameter value for completed indicator (which should be 1 to mimic Manual Release API).

Input Parameters

Parameter	Type string will be converted to	Valid Values
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
line_no	integer	pm_matl_dtl.line_no for given plant_code and batch_no
line_type	integer	- 1 = ingredient 1 = product 2 = byproduct
user_id	character	fnd_user.user_id
trans_qty	Oracle Number	greater than zero if line_type = 1 or 2; less than zero if line_type = -1
trans_qty2	Oracle Number	same sign as trans_qty; non-zero if Item is dual UM controlled
whse_code	character	ic_whse_mst.whse_code
lot_no	character	ic_lots_mst.lot_no where item_id = pm_matl_dtl.item_id for given plant_code, batch_no and line_no if Item is lot-controlled; null if Item is not lot-controlled
sublot_no	character	ic_lots_mst.sublot_no where item_id = pm_matl_dtl.item_id for given plant_code, batch_no and line_no and lot_no = given lot_no if Item is sublot-controlled; null if Item is not sublot-controlled
location	character	ic_loct_mst.location where whse_code = given whse_code if Item is location-controlled; null if Item is not location-controlled
completed_ind	character	0 = pending; 1 = completed. Must be zero or one but this value will be disregarded and set automatically in many cases. The user value is only considered for manual/incremental detail lines in a wip batch.

Validations Overview

This API code has a few new validations. This was done to accommodate opening of the function to scenarios which were not handled before. The code now allows entry for pending batches in addition to wip and certified. It also allows entry for auto release detail lines.

In addition to the valid values information listed above, the following are more validations and behaviors that you should be aware of.

- The API logic does not allow allocations for the primary product in a phantom batch.
- Allocations for an unexploded phantom ingredient are not allowed.

- The user can now create a lot on the fly for phantom ingredients. Prior to this, it could only be done for a product or byproduct.
- The completed indicator is dealt with as follows.

Disregard value of completed_ind if we know the rules otherwise use it.	Pending batch = use 0 Certified batch = use 1 Wip batch / auto release line Ingred = use 1 Prod / Byp = use 0 Wip batch / Manual or incremental release use value passed in by user

Insert Detail

Synopsis

```
sword ret;
ret = pm_insert_detail_api (<plant_code>, <batch_no>, <line_no>, <line_type>,
                            <qty>, <item_no>, <dlt_uom>, <release_type>,
                            <scale_type>, <phantom_type>, <scrap_factor>,
                            <user_id>);
if (ret < 1)
  <error>
```

Description

This function provides the user with the capability of inserting detail lines into a given batch. The batch must be either pending or wip. Under certain conditions, data passed in by the user is not used when it's not valid for a particular business scenario. This is explained in the Validations Overview section.

Input Parameters

Parameter	Type string will be converted to	Valid Values
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
line_no	integer	see line number discussion in the Validations Overview section
line_type	integer	Valid values: - 1 = ingredient 1 = product 2 = byproduct
qty	Oracle Number	Represents plan or actual depending on the batch status and line type being inserted. Must be positive or zero.
item_no	character	Valid item number from ic_item_mst
dtl_uom	character	Valid uom from sy_uoms_mst

Parameter	Type string will be converted to	Valid Values
release_type	character	Valid values; 0 = Auto 1 = Manual 2 = Incremental If Null ("") passed in, FM$DEFAULT_RELEASE_TYPE value used.
scale_type	integer	0 = Fixed 1 = Linear
phantom_type	integer	Valid values; 0 = Not a phantom 1 = Auto Phantom 2 = Manual Phantom **Note:** This is ignored if the batch_type is WIP.
scrap_factor	Oracle Number	Defaults to zero if no value passed in. Any value passed in must be less than 10000 and greater than or equal to zero. This number is divided by 100 to come up with the factor (same as the form). If user wants 10 % scrap, pass in integer 10.
user_id	character	fnd_user.user_id

Validations Overview

In addition to the standard validations above there are a number of additional business rule validations. Many of these situations are deduced by the logic and therefore allow us to disregard data being passed in by the user when it's invalid or unnecessary. To make this more flexible and user friendly we chose to override invalid data wherever feasible instead of passing an error code back. These business rules are defined here.

1. Batch status must be pending or wip.
2. Phantom type value is only used for ingredients in a pending state. This has no meaning for products and byproducts, and unexploded phantom ingredients are not valid in a wip state, therefore the value is defaulted to zero for these cases.
   - If a phantom type value, other than zero, is passed in, for a product line, it is disregarded. Phantom type is not used with products, and on the forms you cannot even access this field when initiated from a product. The code simply puts a zero in for the new line.
3. When inserting a product into a phantom batch, you cannot specify a line number that is a lower than the primary product's line number.
4. The dtl uom has to have a valid conversion set up to the item's primary uom.
5. Scrap factor is defaulted to zero for line types other than ingredients. Example:
   - If a scrap factor value, other than zero, is passed in, for a product line, it is disregarded. Scrap factor is not used with products, and on the forms you cannot even access this field when initiated from a product. The code simply puts a zero in for the new line.
6. Line Number
   a. Passing a NULL, -1 or 0 for line number will insert the detail at the next available line number within the line type given.
   b. If a line number is passed in, the code will do a variety of validations;
      - If the line number given already exists (including blank lines), the API will insert the new line into that spot and shift all the others down, starting from the line number given.
      - If the line number reflects a position currently not occupied, a threshold check is made to make sure this new line will not generate more than three blank lines from the current highest line number within that type.

For example: If the highest line number ingredient is 5, valid line numbers to pass into the API are 0 thru 8 inclusive… 9 or higher will return an error.

- If this is the first line number within a given line type, a threshold check is made to make sure the new line will not generate more than three blank lines. The line number cannot be greater than 4.
- An additional special check is done on line number if the line type being inserted is a product. If the batch in question is a phantom batch, the API will not allow the user to insert a product line with a lower line number than the primary product.

7. Qty value

a. If the batch status is pending, then the qty passed in is used for plan_qty regardless of line_type.

b. If the batch status is WIP and the item is an auto release ingredient, then the qty passed in is used for actual_qty.

c. In all other cases, the qty passed in is ignored and both plan_qty and actual_qty are set to 0.

Certify Batch

Synopsis

```
sword ret;
ret = pm_certify_batch_api (<accept_zero>, <plant_code>, <batch_no>, <user_id>,
                            <accept_incomplete>);

if (ret < 1)
    <error>
```

Description

The Certify Batch API will set the Batch status to certified and the actual completion date to system date. All auto release products will be produced and inventory will be updated. If you are direct certifying a pending batch, all auto release ingredients will be consumed in the same manner as when a batch is released.

Note: For manual/incremental detail lines, incomplete pending transactions can be disregarded by passing "1" in for accept_incomplete.

Input Parameters

Parameter	Type string will be converted to	Valid Values
accept_zero	integer	0 = do not accept actual quantity of 0 upon certification 1 = accept actual quantity of 0 upon certification
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
user_id	character	fnd_user.user_id
accept_incomplete	integer	0 = do not accept any incomplete transactions upon certification 1 = accept any incomplete transactions upon certification (pending transactions are deleted)

3

Process Operations Control APIs

The following pages detail the APIs currently available for the OPM Process Operations Control (POC) module.

All parameter values should be passed in as character strings to all API's in this manual. The individual parameter lists display the data type that a given character string will be converted to internally.

> **Note:** Oracle Number is a proprietary data type that in essence replaces a double type.

Release Step

Synopsis

```
sword ret;
ret = pc_release_step_api (<dtl_index>, <batchstep_no>, <plant_code>,
                           <batch_no>, <user_id>, <start_date>);
if (ret < 1)
    <error>
```

Description

Releasing a Batch Step updates the Step Status of the specified Batch Step to WIP. The Batch must be in WIP status in order to release a Batch Step. The actual start date of the step will be changed to the start_date passed in or the current system date/time if the value is not passed in.

Input Parameters

Parameter	Type string will be converted to	Valid Values
dtl_index	integer	always "-1", not for API use
batchstep_no	integer	pm_rout_dtl.batchstep_no for given plant_code and batch_no
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
user_id	character	fnd_user.user_id
start_date	date	valid date

Certify Step

Synopsis

```
sword ret;
ret = pc_certify_step_api (<dtl_index>, <batchstep_no>, <plant_code>,
                           <batch_no>, <user_id>);

if (ret < 1)
    <error>
```

Description

Certify Step updates the step status to certified and sets the step actual start date to the current date / time. It also sets actual usage and quantity to the planned values if actual values were not previously entered, and it creates completed resource transactions if transactions had not be previously created.

Input Parameters

Parameter	Type string will be converted to	Valid Values
dtl_index	integer	Always "-1", not for API use.
batchstep_no	integer	pm_rout_dtl.batchstep_no for given plant_code and batch_no
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
user_id	character	fnd_user.user_id

Close Step

Synopsis

```
sword ret;
ret = pc_close_step_api (<dtl_index>, <batchstep_no>, <plant_code>, <batch_no>,
                         <user_id>);

if (ret < 1)
    <error>
```

Description

Close Step updates the step status to Closed and sets the step closed date to the current date/time. Closed steps can no longer be editted.

Input Parameters

Parameter	Type string will be converted to	Valid Values
dtl_index	integer	always 1
batchstep_no	integer	pm_rout_dtl.batchstep_no for given plant_code and batch_no
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
user_id	character	fnd_user.user_id

Post Actual Resource Transaction

Synopsis

```
sword ret;
ret = pc_post_tran_actual_api (<usage>, <batchstep_no>, <resource>, <activity>,
                               <trans_date>, <start_date>, <end_date>,
                               <plant_code>, <batch_no>, <user_id>);

if (ret < 1)
    <error>
```

Description

Creates a completed resource transaction and associated actual usage, replacing (deleting) any other resource transactions and actual usage for the resource / activity. Compare with Post Incremental. The step must be in a WIP or Certified state.

Input Parameters

Parameter	Type string will be converted to	Valid Values
usage	Oracle Number	any number greater than or equal to zero
batchstep_no	integer	pm_rout_dtl.batchstep_no for given plant_code and batch_no
resource	character	pm_oprn_dtl.resource for given plant_code, batch_no and batchstep_no
activity	character	pm_oprn_dtl.activity for given plant_code, batch_no, batchstep_no and resource
trans_date	date	valid date
start_date	date	greater than or equal to Operation Detail Actual Start Date
end_date	date	valid date
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
user_id	character	fnd_user.user_id

Post Incremental Resource Transaction

Synopsis

```
sword ret;
ret = pc_post_tran_incr_api (<usage>, <batchstep_no>, <resource>, <activity>,
                             <trans_date>, <start_date>, <end_date>,
                             <plant_code>, <batch_no>, <user_id>);
if (ret < 1)
    <error>
```

Description

Creates a completed resource transaction and adds this to any previous actual usage. Does not delete other existing resource transactions. The step must be in a WIP or Certified state.

Input Parameters

Parameter	Type string will be converted to	Valid Values
usage	Oracle Number	any number greater than or equal to zero
batchstep_no	integer	pm_rout_dtl.batchstep_no for given plant_code and batch_no
resource	character	pm_oprn_dtl.resource for given plant_code, batch_no and batchstep_no
activity	character	pm_oprn_dtl.activity for given plant_code, batch_no, batchstep_no and resource
trans_date	date	valid date
start_date	date	greater than or equal to Operation Detail Actual Start Date
end_date	date	valid date
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
user_id	character	fnd_user.user_id

Post Timed Resource Transaction

Synopsis

```
sword ret;
ret = pc_post_tran_timed_api (<batchstep_no>, <resource>, <activity>,
                              <trans_date>, <start_date>, <end_date>,
                              <plant_code>, <batch_no>, <user_id>);

if (ret < 1)
    <error>
```

Description

Creates a completed resource transaction, using resource start and end date to calculate usage. The step must be in a WIP or Certified state.

Input Parameters

Parameter	Type string will be converted to	Valid Values
batchstep_no	integer	pm_rout_dtl.batchstep_no for given plant_code and batch_no
resource	character	pm_oprn_dtl.resource for given plant_code, batch_no and batchstep_no
activity	character	pm_oprn_dtl.activity for given plant_code, batch_no, batchstep_no and resource
trans_date	date	valid date
start_date	date	valid date
end_date	date	valid date
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
user_id	character	fnd_user.user_id

Start Resource Usage

Synopsis

```
text trans_id [ITEM_ID_SIZE];
sword ret;
ret = pc_post_tran_start_api (<trans_id>, <batchstep_no>, <resource>,
                              <activity>, <trans_date>, <start_date>,
                              <plant_code>, <batch_no>, <user_id>);

if (ret < 1)
    <error>
```

Description

Used with the End Resource Usage API to create a completed resource transaction. The Start Resource Usage API accepts the start date. The resource transaction usage will be zero until the end date is posted. The step must be in a WIP or Certified state.

Input Parameters

Parameter	Type string will be converted to	Valid Values
&trans_id	address of long	valid address
batchstep_no	integer	pm_rout_dtl.batchstep_no for given plant_code and batch_no
resource	character	pm_oprn_dtl.resource for given plant_code, batch_no and batchstep_no
activity	character	pm_oprn_dtl.activity for given plant_code, batch_no, batchstep_no and resource
trans_date	date	valid date
start_date	date	greater than or equal to Operation Detail Actual Start Date
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
user_id	character	fnd_user.user_id

End Resource Usage

Synopsis

```
sword ret;
ret = pc_post_tran_end_api (<trans_id>, <batchstep_no>, <resource>, <activity>,
                            <trans_date>, <end_date>, <plant_code>, <batch_no>,
                            <user_id>);
if (ret < 1)
    <error>
```

Description

Used with the Start Resource Usage API to create a completed resource transaction and associated actual usage. The usage is calculated by comparing the start and end dates. The step must be in a WIP or Certified state.

Input Parameters

Parameter	Type string will be converted to	Valid Values
trans_id	long	trans_id of 'started' Resource Transaction
batchstep_no	integer	pm_rout_dtl.batchstep_no for given plant_code and batch_no
resource	character	pm_oprn_dtl.resource for given plant_code, batch_no and batchstep_no
activity	character	pm_oprn_dtl.activity for given plant_code, batch_no, batchstep_no and resource
trans_date	date	valid date
end_date	date	valid date
plant_code	character	sy_orgn_mst.orgn_code where plant_code = 1
batch_no	character	pm_btch_hdr.batch_no for given plant_code
user_id	character	fnd_user.user_id

4
Error Codes

This chapter numerically lists the error PM and POC API error codes.

(-100) COPY_TO_SDA_ERR

Unable to update data currently resident in memory. This indicates a serious problem at the system level.

(-103) SYTYPE_LOAD_ERR

Could not retrieve a type defined in table sy_type_mst. Ex. Could not find proper batch status during create batch.

(-108) DEFAULT_TRANS_LOST

Default transaction for a specific line cannot be located. Batch is probably corrupt

(-110) TRANS_FOR_LINE_LOST

Unable to locate list for Transactions for Material Detail line. This indicates a serious internal error that cannot be resolved. Please contact OPM Support.

(-112) FETCH_FROM_SDA_ERR

Failure to access Batch currently resident in memory. This may indicate a problem at the system level.

(-114) PMCOMMON_ACCEPT_SHORTAGE_ERR

Accept shortage parameter for release batch API is not a "0" or "1", Invalid value.

(-116) UOM_CONV_ERR

Unable to perform unit of measure conversion.

Verify that a unit of measure conversion exists from the source to the target unit of measure. In the PM module this may be the Item and Material Detail units of measure.

(-117) BATCH_LOCK_ERR

Problem when attempting to lock the batch during load.

(-118) MEM_ALLOC_ERR

A problem occurred when trying to allocate memory. This could happen in many different areas.

(-119) LOT_STATUS_MISSING

Lot status for a particular lot could not be located.

(-123) PMCOMMON_SDA_ACCESS_ERR

Code could not extract information from memory about the batch.

(-138) PMCOMMON_SYCONST_ERR

Error attempting to retrieve system constant from database.

Verify that the constants PM$AUTO_REL_ALLOC_ONLY, PM$CHECK_INV_RELEASE and PM$CHECK_LOT_STATUS exist using the Constants form located on the OPM System menu.

(-139) PMCOMMON_BAD_ARG

Required input parameter not passed.

Verify that all the necessary input parameters are being passed to the given function.

(-146) PMCOMMON_CLOSE_STATUS_ERR

Current Batch status is invalid for close.

(-147) PMCOMMON_CANCEL_STATUS_ERR

Current Batch Status is invalid for Cancellation.

Verify that the current Batch Status is Pending.

(-149) PMCOMMON_IN_USE

The Batch being Released is currently In Use locked by another user and cannot be Released at this time.

(-150) PMCOMMON_CERTIFY_STATUS_ERR

Current Batch status is invalid for certification.

(-151) PMCOMMON_RELEASE_STATUS_ERR

Current Batch status is invalid for release.

(-154) PMCOMMON_NO_MATL_DTL

No Material Details for the Batch were retrieved from the database.

This indicates that an unforeseen error has occurred. Either the Material Details for the Batch were deleted from the database independently of the OPM application or some other failure at the system level.

(-156) PMCOMMON_BATCH_CANCELLED

The Batch has been Canceled and cannot be edited.

(-157) PMCOMMON_BATCH_CLOSED

The Batch has been Closed and cannot be edited.

(-161) PMCOMMON_INVENTORY_SHORT

Inventory Shortages exist for one or more Ingredients. The Accept Shortage input parameter indicates whether this condition should be an error or should be ignored.

(-162) PMCOMMON_LINE_NOT_FOUND

Material Detail Line not found for given Batch.

Verify that the specified Material Detail Line exists for the given Batch.

(-163) PMCOMMON_LOT_NOT_FOUND

Lot Number does not exist.

(-164) PMCOMMON_RELEASE_TYPE_ERR

Auto release detail lines invalid for manual release API.

(-165) PMCOMMON_TRANS_QTY_ERR

Invalid Transaction Quantity passed as input parameter.

Verify that the Transaction Quantity input parameter is greater than or equal to zero for Products and Byproducts and less than or equal to zero for Ingredients.

(-166) PMCOMMON_INVALID_PARAMETER

Invalid values are being passed into the API parameters.

(-167) PMCOMMON_DATE_CONV_ERR

Invalid dates being passed into the API.

(-168) PMCOMMON_RESCHEDULE_DATE_ERR

Invalid date ranges being passed into reschedule batch API.

(-169) PMCOMMON_INVALID_EFF

The given Effectivity exists but is not valid for the attempted Operation.

When Rerouting a Batch verify that the given Effectivity is for the same Formula as the original Effectivity and a different Routing.

(-170) PMCOMMON_BATCH_TYPE_ERR

The given Batch is not a Batch or an FPO.

Verify that the Batch Type parameter is set correctly. Please refer to the Production Management module documentation for the valid values.

(-178) PMCOMMON_INVALID_FOR_PHANTOM

This API cannot be used directly on a phantom batch.

(-180) PMCOMMON_UNEXPLODED_PHANTOMS

The current Batch is a Phantom Ingredient in another Batch. The Release Batch function must be performed on the Parent Batch.

(-181) PMCOMMON_DATE_REQD

One of the date input parameters is missing or blank.

Verify that Planned Start Date, Expected Completion Date and Due Date are valid dates.

(-182) PMCOMMON_RESCH_STATUS_ERR

Current Batch status is invalid for Reschedule.

(-184) PMCOMMON_ITEM_QTY_ERR

The Item Quantity input parameter is invalid.

Verify that the Item Quantity is greater than or equal to zero.

(-185) PMCOMMON_START_CMPLT_ERR

The Expected Completion Date is earlier than the Planned Start Date.

Verify that the Expected Completion Date is greater than or equal to the Planned Start Date.

(-188) PMCOMMON_ACCEPT_ZERO_ERR

The Accept Zero input parameter is invalid.

Verify that the Accept Zero input parameter is zero or one.

(-189) PMCOMMON_ORGN_NOT_PLANT

The Organization input parameter is valid but is not a Plant.

Verify that the Organization is a Plant.

(-193) PMCOMMON_PTCT_ADJ_CERT_FLAG

The adjust certification flag being passed into partial certification API is invalid or 0 (zero = do not adjust certification) was passed in for a batch that is in a certified state.

(-194) PMCOMMON_PTCT_QTY_ERR

Zero quantity invalid for partial certification API.

(-195) PMCOMMON_PTCT_DTL_LINE_ERR

Invalid line to drive partial certification.

(-196) PMCOMMON_PTCT_STATUS_ERR

Current Batch status is invalid for Partial certification.

(-197) PMCOMMON_PTCT_NOT_POSITIVE

Quantities less than or equal to zero are invalid when doing partial certification with a new actual quantity.

(-198) PMCOMMON_NO_PROD

There are no products in this batch.

(-199) PMCOMMON_INVALID_USER_ID

User ID is invalid.

(-200) MSDA_ACTUAL_START_REQD

Actual Start date required for release batch API.

(-201) PMSDA_TRANS_SURG_ERR

Error obtaining Surrogate Key for Transaction.

Verify that the Surrogate Control table, sy_surg_ctl, contains a row for the trans_id surrogate.

(-207) PMSDA_DEFAULT_LOCT_ERR

Unable to determine the Default Lot and / or Location.

Verify that IC$DEFAULT_LOT and IC$DEFAULT_LOCT are defined by using the Constants form on the OPM System menu.

(-210) (PMCOMMON_UNALLOC_ITEMS)

Batch contains unallocated items.

Allocate all items before certifying a batch.

(-213) PMSDA_RELEASE_STATUS_ERR

Current Batch status is invalid for release.

(-215) PMSDA_WHSE_SELECT_ERR

Unable to assign default Warehouses to Material Detail lines.

Verify that Plant-Warehouse Effectivities exist for the given Plant using Plant-Warehouse on the OPM MPS menu.

(-220) PMSDA_WHSE_LOAD_ERR

Unable to load Warehouse Effectivities.

Verify that Plant-Warehouse Effectivities exist for the given Plant using the Plant-Warehouse form on the OPM MPS menu.

(-222) PMSDA_INVALID_LINE_TYPE

Invalid Line Type passed as input parameter.

Verify that the Line Type passed is valid. See the Production Management module documentation for more information.

(-224) PMSDA_INVENTORY_SHORT

Inventory shortages exist for details on this batch.

(-227) PMSDA_DEFAULT_TRANS_LOST

Unable to locate default Transaction for Material Detail line.

This indicates that the data has been corrupted. Please contact OPM Support.

(-230) PMSDA_INDIV_ALREADY_ALLOCD

Indivisible: Lot is already allocated.

(-232) PMSDA_WIP_WHSE_REQD

Required field WIP Warehouse has no value.

The attempted operation requires that the Batch currently loaded into memory has a valid WIP Warehouse.

(-233) PMSDA_INV_NOT_USABLE

Inventory selected is not usable for production.

(-236) PMSDA_UOM_ITEM_CNV_ERR

Error performing unit of measure conversion.

Verify that the Formula Detail units of measure can be converted to the corresponding Item Secondary unit of measure (if applicable). Also verify that the Batch can be successfully Scaled.

(-238) PMSDA_INCOMPLETE_MANUAL

Pending transactions exist for manual or incremental detail lines.

(-241) PMSDA_QTY_NEGATIVE

Formula contains Ingredient, Product or Byproduct with a negative quantity.

Correct the Formula using Formulas on the OPM Formula menu.

(-244) PMSDA_INDIV_NOINV

Inventory does not exist for this indivisible item. This will appear for ingredient consumption detail lines.

(-245) PMSDA_INDIV_QUANTITIES_EXIST

Inventory already exists for this indivisible item. This will appear for production replenishment detail lines.

(-301) PMMTLDTL_CURSOR_ERR

Error opening/closing cursor for PM Material Detail table.

Verify that the table does exist and that you have access to it. If so then a problem may exist at the system level.

(-302) PMMTLDTL_RESIZE_ERR

Unable to dynamically allocate memory.

Verify that the system is configured with sufficient memory. If so then a problem may exist at the system level.

(-303) PMMTLDTL_INSERT_ERR

Error encountered inserting a detail line in pm_matl_dtl.

(-304) PMMTLDTL_UPDATE_ERR

Error encountered updating a detail line in pm_matl_dtl.

(-305) PMMTLDTL_FETCH_ERR

Error retrieving Material Detail Line or the Line does not exist.

Verify that the given Material Detail line exists on the Batch. If so, then an unknown error has occurred retrieving from the PM Material Detail table, pm_matl_dtl. Please contact OPM Support.

(-350) PMBTCHDR_NOT_FOUND

Batch Header not found.

Verify that the given Batch exists for the specified Plant, that it is indeed a Batch and not an FPO and that it has not been marked for deletion.

(-351) PMBTCHDR_FETCH_ERR

Error retrieving Batch Header.

An unknown error has occurred retrieving from the Batch Header table, pm_btch_hdr. Please contact OPM Support.

(-352) PMBTCHDR_INSERT_ERR

Error encountered inserting a line in pm_btch_hdr.

(-353) PMBTCHDR_UPDATE_ERR

Error updating Batch Header.

An unknown error has occurred updating the Batch Header table, pm_btch_hdr. Please contact OPM Support.

(-354) PMBTCHDR_PLANT_CODE_REQD

Required parameter Plant Code not found.

Verify that the Plant Code is being passed if the Batch Surrogate is not being used.

(-355) PMBTCHDR_BATCH_NO_REQD

Required parameter Batch Number not found.

Verify that the Batch Number is being passed if the Batch Surrogate is not being used.

(-356) PMBTCHDR_IN_USE

Batch is currently In-Use locked by another user.

(-359) PM_STEPS_IN_USE

Some steps on this batch are in use by another user.

(-362) PMBTCHDR_SELECT_ERR

Batch number, plant code or both are not a unique entity in pm_btch_hdr.

(-367) PM_DUPLICATE_BATCH_NO_ERR

Batch numbers must be unique.

Verify that the batch number does not already exist in the database for the specified plant.

(-625) PMOPNDTL_CURSOR_ERR

Error opening / closing cursor for PM Operation Detail table.

Verify that the table does exist and that you have access to it. If so then a problem may exist at the system level.

(-626) PMOPNDTL_RESIZE_ERR

Unable to dynamically allocate memory.

Verify that the system is configured with sufficient memory. If so then a problem may exist at the system level.

(-627) PMOPNDTL_INSERT_ERR

Error encountered inserting a detail line into pm_oprn_dtl.

(-628) PMOPNDTL_UPDATE_ERR

Error encountered updating a detail line into pm_oprn_dtl.

(-630) PMOPNDTL_DELETE_ERR

Error encountered deleting a detail line into pm_oprn_dtl.

(-650) PMRTEDEP_CURSOR_ERR

Error opening/closing cursor for PM Batch Step Dependencies table.

Verify that the table does exist and that you have access to it. If so then a problem may exist at the system level.

(-651) PMRTEDEP_RESIZE_ERR

Unable to dynamically allocate memory.

Verify that the system is configured with sufficient memory. If so then a problem may exist at the system level.

(-652) PMRTEDEP_INSERT_ERR

Error encountered inserting a detail line into pm_rout_dep.

(-653) PMRTEDEP_UPDATE_ERR

Error encountered updating a detail line into pm_rout_dep.

(-654) PMRTEDEP_DELETE_ERR

Error encountered deleting a detail line into pm_rout_dep.

(-670) PMRTEDTL_DELETE_ERR

Error encountered deleting a detail line into pm_rout_dtl.

(-675) PMRTEDTL_INSERT_ERR

Error encountered inserting a detail line into pm_rout_dtl.

(-676) PMRTEDTL_UPDATE_ERR

Error encountered updating a detail line into pm_rout_dtl.

(-678) PMRTEDTL_CURSOR_ERR

Error opening / closing cursor for PM Batch Step Detail table.

Verify that the table does exist and that you have access to it. If so then a problem may exist at the system level.

(-679) PMRTEDTL_RESIZE_ERR

Unable to dynamically allocate memory.

Verify that the system is configured with sufficient memory. If so then a problem may exist at the system level.

(-725) PMRTEMTL_INSERT_ERR

Error encountered inserting a detail line into pm_rout_mtl.

(-726) PMRTEMTL_UPDATE_ERR

Error encountered updating a detail line into pm_rout_mtl.

(-727) PMRTEMTL_CURSOR_ERR

Error opening / closing cursor for PM Batch Step / Item Associations table.

Verify that the table does exist and that you have access to it. If so then a problem may exist at the system level.

(-728) PMRTEMTL_RESIZE_ERR

Unable to dynamically allocate memory.

Verify that the system is configured with sufficient memory. If so then a problem may exist at the system level.

(-729) PMRTEMTL_DELETE_ERR

Error encountered deleting a detail line into pm_rout_mtl.

(-1025) ICCTRUP_SURKEY_ERR

Error obtaining Surrogate Key for Transaction.

Verify that the Surrogate Control table, sy_surg_ctl, contains a row for the trans_id surrogate.

(-1026) ICCTRUP_INSERT_ERR

Error inserting Completed Transaction into ic_tran_pnd.

This error may occur if the Surrogate Control table is updated incorrectly outside the OPM application. Please contact OPM Support.

(-1101) ICPTRUP_TRANS_ID_ERR

Error obtaining Surrogate Key for Transaction.

Verify that the Surrogate Control table, sy_surg_ctl, contains a row for the trans_id surrogate.

(-1103) ICPTRUP_INSERT_TRAN_ERR

Error inserting Transaction into ic_tran_pnd.

This error may occur if the Surrogate Control table is updated incorrectly outside the OPM application. Please contact OPM Support.

(-1106) ICPTRUP_UPDATE_TRAN_ERR

Error updating Transaction in ic_tran_pnd.

This indicates corrupt data. Please contact OPM Support.

(-1110) ICPTRUP_TRANS_ID_MISSING

Required parameter trans_id missing.

The Transaction Processor will allocate Surrogates for any Transactions if desired. However, if it does not it expects the trans_id to be passed in. If this is not the case then an error will result.

(-1117) ICPTRUP_TRANS _FETCH _ERR

Error retrieving row from ic_tran_pnd.

This indicates that a Transaction no longer exists were one was expected. Verify that the Transactions for the current document are correct in the database.

(-1119) ICPTRUP_NO_ROWS_FOUND

Error retrieving row from ic_tran_pnd.

This indicates that a Transaction no longer exists were one was expected. Verify that the Transactions for the current document are correct in the database.

(-1402) ICVALID_DELETED

Formula contains an Item which has been marked as Deleted.

Remove the Item from the Formula or use Items on the OPM Inventory menu to 'undelete' the Item.

(-1403) ICVALID_INACTIVE_ITEM

Formula contains an Item which has been marked as Inactive.

Remove the Item from the Formula or use Items on the OPM Inventory menu to the Item as Active.

(-1404) ICVALID_EXPER_ITEM

Formula contains an Item which has been marked as Experimental.

Remove the Item from the Formula or use Items on the OPM Inventory menu to mark the Item as not experimental.

(-4001) PCSDA_RESIZE_ERR

Unable to dynamically allocate memory.

Verify that the system is configured with sufficient memory. If so then a problem may exist at the system level.

(-4007) PCSDA_INVALID_DTL_INDEX

Invalid index being passed in for a given api. Ex. Invalid step number.

(-4008) PCSDA_INVALID_OPRN_INDEX

Invalid index for operation.

(-4009) CSDA_INVALID_TRANS_INDEX

Invalid transaction index.

(-4011) CSDA_SURKEY_ERR

Problem occurred trying to assign surrogates.

(-4012) CSDA_STEPS_NOT_LOADED

No steps loaded for this batch. Either this is not a POC batch or there was a problem loading into memory.

(-4013) PCSDA_NO_TRANS_FOUND

No Resource Transaction were loaded for a given Operation Detail line.

Although there may be no Completed Transactions for a given Operation Detail line there must exist a single Pending Transaction which tracks the unallocated quantity. This error indicates that data corruption may have occurred. Please contact OPM Support.

(-4021) PCSDA_PEND_RSRC_TRANS_ERR

Invalid Resource Transaction found

Resource Transactions must be Completed. If a Pending Transaction is encountered while load them into memory an error will be reported.

(-4024) PCSDA_CIRCULAR_DEPEND

Circular reference detected when creating batch steps during batch create.

(-4026) PCSDA_UOM_CONV_ERR

Error performing unit of measure conversion.

Very that the Primary Product unit of measure can be converted to the Routing unit of measure.

(-4053) PCCOMMON_STEP_STATUS_ERR

Invalid Batch Step Status for attempted operation. Please refer to the POC module documentation for more information.

(-4060) PCCOMMON_OP_CODE_REQD

Required parameter Operator Code not passed.

(-4061) PCCOMMON_USAGE_ERR

Actual Usage is less than or equal to zero.

Verify that the Actual Usage is greater than zero.

(-4065) PCCOMMON_BATCH_STATUS_ERR

Invalid Batch Status for attempted operation. Please refer to the POC module documentation for more information.

(-4076) PCCOMMON_NO_POC_DATA

POC data does not exist for this batch.

(-4079) PCCOMMON_TRANS_START_END_ERR

Transaction Start and / or End Dates are invalid.

Verify that the Transaction Start Date is greater than or equal to the Resource Actual Start Date and the Transaction End Date is greater than or equal to the Transaction Start Date.

(-4085) PCCOMMON_RESOURCE_ACTIVITY_ERR

Specified Operation not found.

Verify that the specified Operation exists for the given Batch Step. If so then this error indicates a memory problem which may indicate a system error.

(-4086) PCCOMMON_TRANS_NOT_FOUND

Transaction requested does not exist in these steps.

(-4087) PCCOMMON_ORGN_NOT_PLANT

Invalid plant code.

(-4150) PCTRNPND_INSERT_ERR

Error encountered inserting a detail line into pc_tran_pnd.

(-4151) PCTRNPND_OPEN_CURSOR_ERR

Error opening cursor for POC Transaction table.

Verify that the table does exist and that you have access to it. If so then a problem may exist at the system level.

(-4152) PCTRNPND_CLOSE_CURSOR_ERR

Error closing cursor for POC Transaction table.

Verify that the table does exist and that you have access to it. If so then a problem may exist at the system level.

(-4153) PCTRNPND_RESIZE_ERR

Unable to dynamically allocate memory.

Verify that the system is configured with sufficient memory. If so then a problem may exist at the system level.

(-4154) PCTRNPND_DELETE_ERR

Error encountered deleting a detail line into pc_tran_pnd.

(-4155) PCTRNPND_UPDATE_ERR

Error encountered updating a detail line into pc_tran_pnd.

5
API Tables

The following tables are updated by the API's:

Table	Description
pm_btch_hdr	Production batch header
pm_matl_dtl	Production batch details (ingredients, products, co/by products)
pm_hist_hdr	History of batch changes that potentially have a Financial impact
ic_tran_pnd	Pending and completed inventory transactions
ic_summ_inv	Inventory Summary table
ic_loct_inv	On-hand Inventory balances
ic_lots_mst	Lot Master table
pc_tran_pnd	Resource transactions in POC
pm_rout_dtl	Batch steps in POC
pm_oprn_dtl	Batch operation details in POC
sy_docs_seq	Document Sequencing table

All other tables, referenced by the Production Batch processing, are within the Inventory and Formula Management modules.

6

Transitioning Existing API Code

The following is a guideline to help customers with existing API interfaces to migrate the code to Release 11i (from 11.0 or 4.1). If the current release you're on is checked then you need to perform the action described.

Action	11.0	4.1	Comments
Include pmapi.h instead of pmcommon.h in your C interface code.	x	x	pcapi.h for POC code
Change function names to suffix them with "_api". Ex. pm_close_batch becomes pm_close_batch_api.	x	x	Same holds true for POC.
Drop the batch_handle and session number parameters from all api function calls. They are now obsolete.	x	x	All other parameters should remain in place after dropping the two that are obsolete. Same holds true for POC.
Pass all parameter values in as character strings including quantities, dates, etc.	x	x	Same holds true for POC.
Save batch api is now obsolete, remove it from your code where it's used in tandem with create batch.	x	x	
Create batch has an additional parameter, batch number, for plants using manual doc number sequencing. Change existing call to conform. Pass in null for auto doc numbering plants.	x	x	
Convert your char variable data types to Oracle text data types.		x	**Note:** Inclusion of afpub.h will give you access to other Oracle data types such as sb4 or sword should you choose to use them.
Function sy_connect no longer exists. You will have to code your own connect to the database.		x	Users on 11.0 using the API's would have done this already.

4.10 Customers

Please note that sample interface source code files pmmain.c/pcmain.c are no longer shipped. The 11i documentation has accounted for some of the information typically extracted by end users directly from the sample program. Once your existing program has been modified to meet 11i specifications you can compile it. The compile command will require some additional switches ("-D") to run properly. This is discussed under the compiling and linking overview in this manual.

Also, in 4.10, four separate libraries were shipped for your use during linking. They were; pmapp.a, pmsql.a, pcapp.a and pcsql.a. The functionality represented by these four libraries are now in one library, libgme.a. This library and other necessary ones are also discussed under the compiling and linking section.

Index