Project and Portfolio Management Center

Software Version: All versions

Reporting Meta Layer Guide and Reference

Go to **HELP CENTER ONLINE**
http://admhelp.microfocus.com/ppm/

Document Release Date: August 2020 | Software Release Date: August 2020

Legal Notices

Disclaimer

Certain versions of software and/or documents (“Material”) accessible here may contain branding from Hewlett-Packard Company (now HP Inc.) and Hewlett Packard Enterprise Company. As of September 1, 2017, the Material is now offered by Micro Focus, a separately owned and operated company. Any reference to the HP and Hewlett Packard Enterprise/HPE marks is historical in nature, and the HP and Hewlett Packard Enterprise/HPE marks are the property of their respective owners.

Warranty

The only warranties for products and services of Micro Focus and its affiliates and licensors (“Micro Focus”) are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Micro Focus shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Restricted Rights Legend

Contains Confidential Information. Except as specifically indicated otherwise, a valid license is required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Contents

Chapter 1: Getting Started with the Reporting Meta Layer

The Reporting Meta Layer (RML) for Project and Portfolio Management Center (PPM) allows customers to use third-party reporting software to define custom reports. Any third-party reporting tool capable of running SQL queries on an Oracle® database can work with PPM reporting capabilities by:

- Using the RML schema in the PPM database as its data source
- Building reports using the standard capabilities of the PPM reporting system

Examples of third-party software being used with PPM include Actuate, Brio, Cognos, Crystal Reports, and Oracle Reports.

Target users of the RML are report designers and administrators responsible for creating business reports about Demand Management and
Deployment Management application usage. Micro Focus assumes that these users have a basic understanding of relational database concepts, Oracle technologies, and PPM applications. However, the RML makes it possible for these users to report on Demand Management and Deployment Management data without understanding the technical complexities of the underlying data model.

The RML Schema

The RML is a schema in the PPM Oracle database that has privileges to view tables in the database schema. The RML:

- Resides in a separate layer from the standard PPM database schema
- Has read-only access to PPM data, so that third-party reporting tools using RML capabilities cannot alter or corrupt the PPM database

To prevent a third-party report from exposing information to people who lack the proper authorization, security views (which can be referenced by any other view) are included in the Reporting Meta Layer.

RML database views, which are created through templates, read and interpret data from the PPM database. Views are created through compilation, in which a view template is read, custom information to be included is calculated, and the final view that resides in the Reporting Meta Layer is generated.

The RML stays up-to-date with the current state of Demand Management and Deployment Management data through synchronization.

During the synchronization procedure, each RML view template is parsed and used as a basis for generating an updated view or set of views in the RML schema. View templates dictate the view's construction. User-specified options control which views are compiled during synchronization.

Note: Micro Focus does not recommend changing or deleting RML view templates.

The RML Views

RML views are representations of logical PPM business or functional entities for Demand Management and Deployment Management. RML views are presented as Oracle views.

Caution: In prior versions of PPM, the RML views supported additional applications, such as Resource Management and Project Management. With version 7.5, RML support is limited to Demand Management and Deployment Management.

Reporting Meta Layer Views

The RML views provide visibility into Demand Management and Deployment Management application areas of the PPM product as described in the following sections. Detailed information on the views and RML messages can be found in the appendix.

Cross-Product Views

"Cross-Product Views" on page 12 relates information across Demand Management and Deployment Management application areas. For example, MWFL_STEP_ACTIVITIES shows statistics about workflow step completion across applications.

Deployment Management Views

"Deployment Management Views" on page 13 provides information specific to Deployment Management. For example, MPKGL_OBJ_TYPE_DEPLOYMENT_D provides summary information for package deployment activity, broken down by object type and calendar day. MPKGL_PACKAGE_LINES provides information about package lines, including global package line user data fields.

Demand Management Views

"Demand Management Views" on page 29 provides information specific to Demand Management. For example, MREQ_OPENED_CLOSED_BY_TYPE_D provides summary information for request submission and completion activity, broken down by request type and by calendar day. MREQ-REFERENCES provides information about references related to Demand Management requests.

Other Views

"Other Views" on page 44 provides information about Demand Management and Deployment Management entities like workflows and security groups. For example, MWFL_STEP_SECURITY_

USERS lists all users with authority to act on a given workflow step through static security group or user linkage, as defined in the workflow step window in the Workflow workbench.

These views are useful to report designers. KRML_CALENDAR_DAYS is a utility table that contains daily date records.

For reporting needs not met by the view in the preceding categories, the RML provides entity-specific views that map to the data shown in the user interface. For example, each request type in Demand Management has a unique view in the RML that presents both request detail fields and user data fields. This allows report writers to devise reports that implement specific customer-oriented business logic contained in customer-defined fields.

Chapter 2: Working with the Reporting Meta Layer

Setting Up the Reporting Meta Layer

The following sections describe the basic structure of the RML, and the behavior and maintenance of its views.

RML views are essentially SQL statements that return specific, useful data from the PPM database, providing direct mapping to the business entities defined in PPM applications.

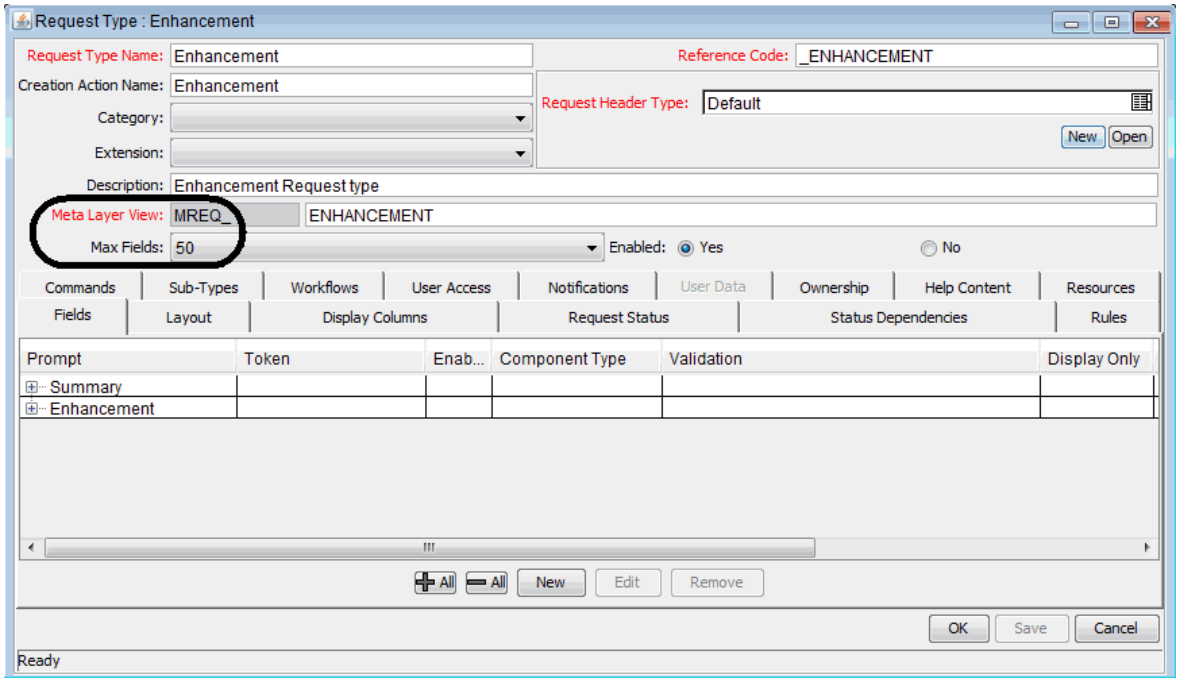
Any third-party reporting software capable of connecting to an Oracle database and running query statements in SQL can use the Reporting Meta Layer. RML views are used by including them in query statements.

Synchronizing the Reporting Meta Layer

PPM transactional entities are particular instances of forms completed for transactions, such as requests and object types, that have their own RML views. Each view is defined by a view template that dictates the view's construction. For example, templates contain markers for entities containing custom fields. When custom fields are encountered during view compilation, the template puts them into the view, using their tokens as column names.

Every time a new entity, such as a request, is created, it must be given a corresponding Meta Layer view name (see " Figure 2-1. Recording Meta Layer view names in entity definitions" on the next page). Each view must have a unique name that cannot be duplicated in the system.

Figure 2-1. Recording Meta Layer view names in entity definitions



Note: RML views are named according to Oracle convention. You must comply with the Oracle naming conventions for your version of the database.

For example, view names may need to be less than 20 characters in length and you should not use Oracle reserved words when naming anything in PPM.

User data fields are also incorporated into many RML views. The types of user data that could be present in one or more RML views includes the following:

- **Package user data:**
  - Contact user data
  - Package line user data
  - Project environment data
  - Request type user data
  - Security group user data
  - Workflow step user data
  - Workflow user data
- **Global and context-sensitive user data:**

  Global user data only.

As part of routine PPM configurations, users can update custom fields, entity names, and other configuration information at any time. Every change has the potential to render existing RML views obsolete, invalidating reports based on these obsolete views. For configuration changes to

be reflected in the Reporting Meta Layer, it must be synchronized to keep RML views current with PPM configurations.

The Synchronization Procedure

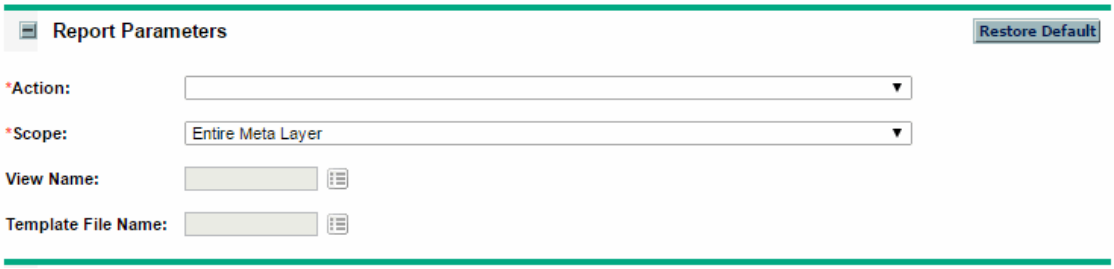
To synchronize the Reporting Meta Layer:

1. In the PPM standard interface, select **Open > Reports > Create Report**.

   The Submit New Report page opens.
2. In the bottom section (Select Report by Category), select **Administrative**.
3. From the list of Administrative reports, select **Synchronize Meta Layer**.

   The Submit Report: Synchronize Meta Layer window opens.

   Figure 2-2. Synchronizing the Meta Layer



4. Select the **Action** you want:
   - To simulate a synchronization, select **Assess**.

     This generates a synchronization report listing the updates that would be made to the Reporting Meta Layer if the synchronization were implemented, allowing the impact of any changes to be assessed.
   - To perform the actual synchronization, select **Synchronize**.

     This compiles all views from existing view templates and generates a report of the updates made, subject to the scope specified (see step 5).
   - To remove views in the Reporting Meta Layer (including the entire RML itself) that are no longer needed, select **Drop.**
5. Select the **Scope** you want:
   - To perform the selected action on the entire Reporting Meta Layer, select **Entire Meta Layer**.
   - To activate the View Name auto-complete list, select **Specific View**.

     You must select a view to perform the selected action.
   - To activate the Template File Name auto-complete list, select **Specific Template**.

     You must select a view template to perform the selected action.
6. To run the report, click **Submit**.

The result depends on your choices. If you chose **Synchronize** and **Entire Meta Layer,** running the reports synchronizes the entire meta layer, and the RML view is current with PPM configurations made since the last synchronization.

A window opens, describing the views created by the report. The views in the list may differ from what is generated in your database and from the list of views in "Reporting Meta Layer Views" on page 12. You should consider the list of views in "Reporting Meta Layer Views" on page 12 and in your database to be the most accurate and up-to-date.

Appendix A: Reporting Meta Layer Views

Cross-Product Views

Cross-product views relate information across PPM products. Each view is described in the sections that follow.

For example, MWFL_STEP_ACTIVITIES shows statistics about workflow step completion across applications.

MWFL_STEP_ACTIVITIES

This view contains activity statistics for all workflow steps, including subworkflows. For any given workflow or workflow step, MWFL_STEP_ACTIVITIES can be used to get a quick snapshot of aggregate system activity. It is provided as a general reference for gathering data that is not covered by other product-specific statistical views. The internal ID columns for workflow and workflow step (WORKFLOW_ID and WORKFLOW_STEP_ID) can be used to join this view to other product action or workflow-related views to gather additional information about the records contained therein.

This view can also be used to flag step duration issues by looking at step completion times (AVG_TIME_TO_COMPLETE and AVG_TIME_OPEN), or other exceptions like spikes in the number of cancelled workflow steps for a point in time.

Sample

A report needs to contain summary information for the number of errors for step 2 in the FIN dev-test-prod workflow, broken down by month. The calendar table described in "KRML_CALENDAR_DAYS and KRML_CALENDAR_MONTHS" on page 49 can be used to provide the month-by-month breakdown to join with the ACTIVITY_DATE column in this view:

```
SELECT m.calendar_month MONTH,
       sum(sa.error) NUM_ERRORS
FROM   krml_calendar_months m,
       mwfl_step_activities sa
WHERE  sa.workflow = 'FIN dev-test-prod'
AND    sa.workflow_step_number = 2
AND    sa.activity_date >= m.start_date
AND    sa.activity_date < m.end_date
GROUP BY m.calendar_month
ORDER BY 1;
```

Results

```
    MONTH    NUM_ERRORS
-----------------------
01-APR-01            16
01-MAY-01             4
01-JUN-01             0
01-AUG-01             0
01-SEP-01             1
```

Deployment Management Views

Deployment Management views provide information specific to Deployment Management. For example, "MPKGL_OBJ_TYPE_DEPLOYMENT_D/M" on page 23 provides summary information for package deployment activity, broken down by object type and calendar day. "MPKGL_PACKAGE_LINES" on page 24 provides information about package lines including global package line user data fields.

MPKG_DEPLOYMENT_DETAILS

Provides information on the details of object deployments to environments. MPKG_DEPLOYMENT_DETAILS has a record for each deployment.

This view is based on object deployment history stored in the environment contents tables. As a result, it includes accurate records for deployments even when the destination environment specified on the migration workflow step was overridden during object type command processing.

Sample

The following example reports on all objects deployed to the MFG Prod environment in the last day:

```
SELECT package_number  package,
       line_number     line,
       object_type     object,
       object_name     name,
       object_revision version
FROM   mpkg_deployment_details
WHERE  destination_environment = 'MFG Prod'
AND    deployment_date > sysdate - 1;
```

Results

```
package  line object             name                version
-------- ---- ------------------ ------------------- -------
30023       3 Migrate SQL file   add_user.sql        3.12
30023       5 Migrate SQL file   create_links.sql    8
30121       1 File Migration     runProcess.sh       2.7
30122       1 FIle Migration     runProcess.sh       2.9
```

...

MPKG_NOTES

Provides access to the notes for all packages in Deployment Management.

Notes are stored in an Oracle LONG column; to prevent an overload of information this is presented in a separate Meta Layer view, making it less likely to design a report that inadvertently returns too much data.

To query package notes, join this view with the MPKG_ALL_PACKAGES view.

Sample

To retrieve a list of the notes for all open packages being processed through the FIN dev -> prod workflow, and that have Critical priority, use the following logic in an SQL statement:

```
SELECT p.package_number PKG_NUM,
       n.NOTE_DATA NOTES
FROM   mpkg_packages p,
       mpkg_notes n
WHERE  p.priority = 'Critical'
AND    p.workflow = 'FIN dev -> prod'
AND    p.package_id = n.package_id;
```

MPKG_PACKAGES

The most general view into package transaction data. A blind query (that is, `SELECT * FROM mpkg_packages`) returns one row for each package present in the system, including closed packages. For information about individual package lines, use the other views that provide line detail.

The view columns map to package header fields, such as Priority, Package Group, and Assigned-to User. There are also columns for the package status and the dates on which it was submitted, closed, or cancelled. Because global package user data fields are present on all packages, there is also a view column for each global package user data field that is defined.

The column name for each global package user data field is the same as the token name for that field. Context-sensitive package user data sets have their own views. See "MPKG_UD_<Context Value>" on page 18.

The "MPKGL_PACKAGE_LINES" on page 24 view can be used to query general package line data, including package line user data fields. If it is necessary to report on the activity of specific object types, the set of object type-specific views is more appropriate. See "MPKGL_<Object Type Name>" on page 19.

Sample 1

To determine the number of open packages and to whom they are assigned:

```
SELECT assigned_to_username ASSIGNED_USER,
       COUNT(*) NUM_OPEN
FROM   mpkg_packages
WHERE  close_date IS NULL
AND    cancel_date IS NULL
AND    submission_date IS NOT NULL
GROUP BY assigned_to_username
ORDER BY 1;
```

Results 1

```
ASSIGNED_USER                  NUM_OPEN
---------------------------------------
...
rfrazier                             13
rjeffries                             1
rjones                               28
rnelson                               9
rsmith                                3
...
```

Sample 2

A global package user data field has been defined to capture the username of a backup user responsible for each package. The token name for this field is BACKUP_USERNAME.

```
SQL> desc mpkg_packages;
```

Results 2

```
Name                           Null?    Type
------------------------------ -------- ----
PACKAGE_NUMBER                 NOT NULL VARCHAR2(30)
PACKAGE_DESCRIPTION                     VARCHAR2(240)
...
PACKAGE_TYPE_CODE              NOT NULL DATE
BACKUP_USERNAME                         VARCHAR2(200)
PARENT_REQUEST_ID                       NUMBER
CREATED_BY                     NOT NULL VARCHAR2(30)
...
```

This new column can be used to drive a report. For example, to report on packages that have been open for more than five days and assigned to a particular backup user:

```
SELECT backup_username BACKUP_USER,
       assigned_to_username ASSIGNED_USER,
       COUNT(*) NUM_OLD_REQS
FROM   mpkg_packages
WHERE  backup_username = '<ValidUsername>'
AND    close_date IS NULL
AND    cancel_date IS NULL
```

```
AND    submission_date IS NOT NULL
AND    (sysdate – submission_date) > 5
GROUP BY backup_username, assigned_to_username
ORDER BY 1, 2;
```

This query also displays the original user to whom the package was assigned.

MPKG_PENDING_PACKAGES

Used to create a report that shows the volume of open packages for any given workflow in Deployment Management.

Provides a quick snapshot of ongoing package processing work. It shows:

- a summary of packages currently open for a specific Deployment Management workflow (for example, total number or average age)
- information on how many packages have been opened and closed in the current week and current month
- priority of the packages

  Priority is usually the most important breakdown of load information. Data is grouped into three priority groupings: P1, P2, and P3. These groupings map to the three highest–priority levels defined.

MPKG_PENDING_PACKAGES is aggregated across all packages.

Sample

A project manager has deployments running through three separate workflows in a current project. The manager needs a report that will show current work volume in each of these workflows, to help prioritize work and identify bottlenecks. If the three workflows are named MFG prod deployment, FIN prod deployment, and prod backup, the following SQL query can be used for a report:

```
SELECT workflow              Workflow,
       open_packages         Open_Pkgs,
       avg_age_open_packages Avg Age,
       p1_open_packages      P1 Open Pkgs,
       p2_open_packages      P2 Open Pkgs
FROM   mpkg_pending_packages
WHERE  workflow IN
       ('MFG prod deployment',
       'FIN prod deployment',
       'prod backup');
```

Results

```
                                 P1    P2
                    Open  Avg   Open  Open
WORKFLOW            Pkgs  Age   Pkgs  Pkgs
```

```
-------------------- ----- ---- ----- -----
MFG prod deployment      11    9     3     8
FIN prod deployment      39   16    14    25
prod backup               6   54     5     1
```

This view ignores packages that have not been submitted.

MPKG_REFERENCES

References are used throughout PPM to relate transaction entities together. The MPKG_REFERENCES view can be used to view the references of packages in Deployment Management.

There are several types of references for packages. If a package is part of a release, then there will be a reference for that release. If a package was spawned by a request, then there will be a reference for that request. Packages can be related to other packages through the use of references. References are also used to attach documents to a package.

The RELATIONSHIP column in MPKG_REFERENCES describes the relationship of the referenced item to the package that references it. This view also has columns for each of the entities that can be referenced to a package—other packages, projects, tasks, requests, releases, attachments, and URLs. For each record in MPKG_REFERENCES, only one of these columns will have a value and the others will be NULL.

Sample

The following SQL statement can be used to retrieve a list of all references to a particular package:

```
SELECT referenced_package_id PKG,
       referenced_project_id PROJ,
       referenced_request_id REQ,
       referenced_release_id REL,
       referenced_task_id    TASK,
       attachment_name       ATTACHMENT,
       document_url          URL,
       relationship          RELATIONSHIP
FROM   mpkg_references
WHERE  package_number = '30121';
```

Results

```
PKG  PROJ   REQ  REL  TASK   ATTACHMENT       URL       RELATIONSHIP
---- ----- ----- ---- ----- --------------- ---------- --------------------
                30012                                  Contains this Package
30332                                                  Run after this Package
30043                                                  Run before this Package
30044                                                  Run before this Package
30046                                                  Run before this Package

                           design32_3.doc
```

MPKG_UD_<*Context Value*>

Set of views containing context-sensitive package user data information.

When the Reporting Meta Layer is synchronized, a view is created for every set of context-sensitive package user data fields defined in the system. The name of each view is defined in the User Data window in the Meta Layer View field. It defaults to a prefix MPKG_UD_ and a suffix that defaults to the first 20 alphanumeric characters of the corresponding context value.

For example, if there are two sets of context-sensitive package user data defined in PPM, with a Workflow context field and context values FIN dev -> prod and MFG dev -> prod, then two corresponding Meta Layer views would exist: MPKG_UD_FIN_DEV_PROD and MPKG_UD_MFG_DEV_PROD.

If no context-sensitive package user data has been defined in the User Data window, then no views of this type will exist in the Meta Layer. Global package user data fields are incorporated directly into the package view MPKG_PACKAGES and therefore do not require a separate unique view.

If context-sensitive package user data has been defined, only new packages with this user data and existing packages that have been edited will appear in the views.

Sample 1

Continuing the example, there are two package user data fields defined for the FIN dev -> prod workflow context, with tokens named VERSION_CTL_PROJECT and VERSION_CTL_ENV.

In the corresponding view MPKG_UD_FIN_DEV_PROD, two columns named the same as the token names would be present:

```
SQL> desc mpkg_ud_fin_dev_prod;
```

Results 1

```
Name                           Null?    Type
------------------------------ -------- ----
PACKAGE_NUMBER                 NOT NULL VARCHAR2(30)
PACKAGE_TYPE                   NOT NULL VARCHAR2(80)
CONTEXT_FIELD                           VARCHAR2(80)
CONTEXT_VALUE                           VARCHAR2(200)
CONTEXT_CODE                            VARCHAR2(200)
VERSION_CTL_PROJECT                     VARCHAR2(200)
VERSION_CTL_ENV                         VARCHAR2(200)
CREATION_DATE                  NOT NULL DATE
CREATED_BY_USERNAME            NOT NULL VARCHAR2(30)
LAST_UPDATE_DATE               NOT NULL DATE
PACKAGE_ID                     NOT NULL NUMBER
```

Sample 2

A report is needed that shows the number of open packages that are being processed through the FIN dev -> prod workflow, broken down by VERSION_CTL_PROJECT and priority:

```
SELECT f.version_ctl_project PROJECT,
       p.priority PRIORITY,
       COUNT(*) NUM_OPEN_PKGS
FROM   mpkg_ud_fin_dev_prod f,
       mpkg_packages p
WHERE  p.close_date IS NULL
AND    p.cancel_date IS NULL
AND    p.submission_date IS NOT NULL
AND    p.package_id = f.package_id
GROUP BY f.version_ctl_project, p.priority
ORDER BY 1, 2;
```

Results 2

```
PROJECT                          PRIORITY        NUM_OPEN_PKGS
-------------------------------- --------------- -------------
Rel 3.0                          High                        2
                                 Normal                     12
                                 Low                        32

Rel 2.1.2                        Critical                    1
                                 High                        1
                                 Normal                      8
                                 Low                         3

Rel 2.1                          Low                        23
...
```

MPKGL_<*Object Type Name*>

Set of views containing object type-specific package line information.

When the Reporting Meta Layer is synchronized, a view is created for every object type defined in the system. The name of each view is defined on the object type screen in the Meta Layer View field. It defaults to a prefix MPKGL_ and a suffix that defaults to the first 20 alphanumeric characters of the corresponding object type name.

If there are three object types defined in Deployment Management named Java File Migration, SQL Script Migration, and Forms 4.5 Migration, then three corresponding Meta Layer views would exist: MPKGL_JAVA_FILE_MIGRATION, MPKGL_SQL_SCRIPT_MIGRATION, and MPGKL_FORMS_45_MIGRATION.

The view columns are identical to those of the general MPKGL_PACKAGE_LINES view (including the package line user data fields), and they include additional columns for each custom field for the object type. This allows a report designer to create a report that implements business logic that drives off of customer-defined object type fields.

For example, consider the Java File Migration object type. This object type might have custom fields with tokens such as FILE_NAME, FILE_LOCATION, and SUB_PATH. The corresponding view MPKGL_JAVA_FILE_MIGRATION would contain columns with these names.

Sample 1

```
SQL> desc mpkgl_java_file_migration;
```

Results 1

```
Name                           Null?    Type
------------------------------ -------- ----
PACKAGE_NUMBER                 NOT NULL VARCHAR2(40)
LINE_NUMBER                    NOT NULL NUMBER
...
CANCEL_DATE                             DATE
FILE_NAME                               VARCHAR2(200)
SUB_PATH                                VARCHAR2(200)
FILE_LOCATION                           VARCHAR2(200)
CREATION_DATE                  NOT NULL DATE
CREATED_BY_USERNAME            NOT NULL VARCHAR2(30)
...
```

Sample 2

To continue the example, a report is needed that will list the PPM user who is assigned to open packages containing one or more package lines that are Java File Migration objects, and that are eligible for migration.

A SQL query such as the following might handle this:

```
SELECT p.workflow,
       p.assigned_to_username ASSIGNED_USER,
       COUNT(UNIQUE(p.package_id)) NUM_ELIGIBLE
FROM   mpkg_packages p,
       mpkgl_package_line_actions pla,
       mpkgl_java_file_migration j
WHERE  j.close_date IS NULL
AND    j.cancelled_flag = 'N'
AND    j.submission_date IS NOT NULL
AND    j.package_line_id = pla.package_line_id
AND    pla.status_type = 'ELIGIBLE'
AND    j.package_id = p.package_id
GROUP BY p.workflow, p.assigned_to_username
ORDER BY 1, 2;
```

MPKGL_APP_DEPLOYMENT_D/M

Summary information for package deployment activity, broken down by application, environment, and calendar day for MPKGL_APP_DEPLOYMENT_D and month for MPKGL_APP_

DEPLOYMENT_M.

This information can be used to quickly assess regular package throughput for each application managed by the IT department, and can help indicate trends in package processing over time for a specified application. An application corresponds to an app code designated in environment definitions.

Besides just the number or packages which were deployed on a given day or month, these views also contain columns to show the number of packages and package lines that were involved in listed deployments, and the number of different object types that were used.

Results from a query of one of these views contain records only for days or months on which deployments occurred for each application.

Sample

The following SQL query can be used as a basis for a report that summarizes all package deployment activity, per day, for a specified application, over a range of dates:

```
SELECT app_code          Application,
       environment       Dest_Env,
       deployment_date   Date,
       total_deployments Total_Deployed,
       unique_obj_types  Num_Obj_Types
FROM   mpkgl_app_deployment_d
WHERE  deployment_date BETWEEN '01-APR-01' AND '05-APR-01'
AND    app_code = 'FINAPP02'
ORDER BY deployment_date;
```

To get a breakdown by month, replace `deployment_date` with `deployment_month` and `mpkgl_app_deployment_d` with `mpkgl_app_deployment_m`.

Results

```
                                                 Num
                                       Total     Obj
Application  Dest_Env         Date     Deployed  Types
------------ ---------------- -------- -------- -------
FINAPP02     FIN Test 1       01-APR-01       42       4
FINAPP02     FIN Test 2       01-APR-01       12       2
FINAPP02     FIN Prod         01-APR-01        2       1
FINAPP02     FIN Test 1       02-APR-01        3       1
FINAPP02     FIN Test 2       02-APR-01       55       3
FINAPP02     FIN Prod         02-APR-01       39       3
FINAPP02     FIN Test 1       03-APR-01       18       4
FINAPP02     FIN Test 2       03-APR-01       22       3
FINAPP02     FIN Prod         03-APR-01       11       2
...
```

MPKGL_ENV_DEPLOYMENT_D/M

The Reporting Meta Layer views MPKGL_ENV_DEPLOYMENT_D and MPKGL_ENV_DEPLOYMENT_M give summary information for package deployment activity, broken down by environment and calendar day or month.

These views can be used to assess regular package throughput for each environment managed by the IT department, and can help indicate trends in package processing over time for a specified environment.

Besides just the number of packages which were deployed on a given day or month, these views also contain columns to show the number of packages and package lines that were involved in listed deployments, and the number of different object types that were used.

Results from a query of one of these views contain records only for days or months on which deployments occurred for each environment.

Sample

The following SQL query can be used as a basis for a report that summarizes all package deployment activity, per day, for a specified environment, over a range of dates:

```
SELECT environment       Dest_Env,
       deployment_date   Date,
       total_deployments Total_Deployed,
       unique_obj_types  Num_Obj_Types
FROM   mpkgl_env_deployment_d
WHERE  deployment_date BETWEEN '01-APR-01' AND '10-APR-01'
AND    environment = 'FIN Test 2'
ORDER BY deployment_date;
```

To get a breakdown by month, replace `deployment_date` with `deployment_month` and `mpkgl_env_deployment_d` with `mpkgl_env_deployment_m`.

Results

```
                                    Num
                             Total  Obj
Dest_Env         Date      Deployed Types
---------------- --------- -------- -------
FIN Test 2       01-APR-01       12       2
Fin Test 2       02-APR-01       55       3
FIN Test 2       03-APR-01       22       3
FIN Test 2       04-APR-01        3       1
FIN Test 2       05-APR-01       18       4
FIN Test 2       06-APR-01       39       3
FIN Test 2       07-APR-01       18       4
FIN Test 2       09-APR-01       22       3
FIN Test 2       10-APR-01        3       1
```

MPKGL_OBJ_TYPE_DEPLOYMENT_D/M

The views MPKGL_OBJ_TYPE_DEPLOYMENT_D and MPKGL_OBJ_TYPE_DEPLOYMENT_M give summary information for package deployment activity, broken down by object type and calendar day or month.

These views can be used to:

- assess regular package throughput for each object type used by the IT department
- help indicate trends in package processing over time for a specified object type
- show the number of packages and package lines that were involved in listed deployments, and the number of different environments to which they were deployed

Results from a query of one of these views contain records only for days or months on which deployments occurred for each object type.

Sample

The following SQL query can be used as a basis for a report that summarizes all package deployment activity, per month, for a specified object type, over a range of dates:

```
SELECT object_type         Object_Type,
       deployment_month    Month,
       total_deployments   Total_Deployed,
       unique_environments Num_Envs
FROM   mpkgl_obj_type_deployment_m
WHERE  deployment_month BETWEEN '01-MAR-01' AND '01-AUG-01'
AND    object_type = 'File Migration'
ORDER BY deployment_date;
```

To get a breakdown by day, replace `deployment_month` with `deployment_day` and `mpkgl_obj_type_deployment_m` with `mpkgl_obj_type_deployment_d`.

Results

```
                              Total     Num
Object_Type      Date      Deployed    Envs
---------------- --------- -------- -------
File Migration   01-MAR-01      122      12
File Migration   01-APR-01      104      12
File Migration   01-MAY-01       87      15
File Migration   01-JUN-01      156      16
File Migration   01-JUL-01      263      22
File Migration   01-AUG-01      290      23
```

MPKGL_PACKAGE_LINES

View into package line transaction data. A blind query (`SELECT * FROM mpkgl_package_lines;`) will return one row for each package line present in the system, including closed lines. The view columns map to common package line fields like Sequence, Object Type Name, Object Revision, and App Code. There are also columns for the dates on which it was submitted, closed, or cancelled, and for each package line user data field that is defined.

The column name for each package line user data field is the same as the token name for that field.

This view does not contain an indication of workflow status. Because workflows may be branched and multiple steps might be active at one time, the workflow status is not necessarily a single piece of information that can be represented in a view column. Instead, the report designer must also reference the "MPKGL_PACKAGE_LINE_ACTIONS" below view for workflow step statuses.

Sample

The package line ID is provided as a key column to join "MPKGL_PACKAGE_LINE_ACTIONS" below with MPKGL_PACKAGE_LINES. For example, to list all workflow steps that a particular PPM user is eligible to act on:

```
SELECT p.package_number PKG_NUM,
       pl.line_number LINE_NUM,
       pl.object_name OBJECT,
       pla.workflow_step_number STEP_NUM
FROM   mpkg_packages p,
       mpkgl_package_lines pl,
       mwfl_step_security_users ssu,
       mpkgl_package_line_actions pla
WHERE  pla.status_type = 'ELIGIBLE'
AND    ssu.workflow_step_id = pla.workflow_step_id
AND    ssu.username = 'FJOHNSON'
AND    pla.package_line_id = pl.package_line_id
AND    pla.package_id = p.package_id
ORDER BY 1,2,4;
```

The view column PACKAGE_LINE_ID was used to join MPKGL_PACKAGE_LINES with MPKGL_PACKAGE_LINE_ACTIONS. MWFL_STEP_SECURITY_USERS (see "MWFL_STEP_SECURITY_GROUPS and MWFL_STEP_SECURITY_USERS" on page 44) is used to determine if a specified user is authorized for a specified workflow step.

MPKGL_PACKAGE_LINE_ACTIONS

Used to gather transaction details for any given package line in Deployment Management. Contains columns to display the current status of a step, how long that step has been in the

current status, whether the step is complete or resulted in an error, details about the step (source and destination environment), and other relevant details.

To relate information from this view with detail information from related packages or package lines, the report designer can use the package and package line identifiers (PACKAGE_ID and PACKAGE_LINE_ID columns) to join with other standard views such as MPKG_PACKAGES and "MPKGL_PACKAGE_LINES" on the previous page.

Sample

A report is needed that shows the number of package lines that have had certain actions taken for each calendar week in the last month, broken down by object type, for a customer's Dev-Test-Prod workflow:

```
SELECT  trunc(eligible_date,'WW')                       Week,
        line_object_type                                Object_Type,
        sum(decode(action_name,'Open',1,0))             Opened,
        sum(decode(action_name,'Migrate to Test',1,0)) Into_Test,
        sum(decode(action_name,'Migrate to Prod',1,0)) Into_Prod,
        sum(decode(action_name,'Close',1,0))            Closed
FROM    mpkgl_package_line_actions
WHERE   package_workflow = 'Dev - Test - Prod'
AND     eligible_date > sysdate - 30
GROUP BY trunc(eligible_date,'WW'),
         line_object_type;
```

The column STATUS is the status name that is displayed for lines in the status tab of packages in the Deployment Management application. The internal code STATUS_TYPE is provided to group these status names into logical groupings.

For example, there may be many different statuses that all represent a COMPLETE status type. For example, the result value of any workflow step, such as Approved, Succeeded, Rejected, Failed QA Test.

While STATUS may have many different possible values, STATUS_TYPE has any of the following possible values:

- SUBMITTED
- IN_PROGRESS
- CLOSED_SUCCESS
- ELIGIBLE
- ERROR
- CLOSED_FAILURE
- PENDING
- COMPLETE
- CANCELLED

MPKGL_PENDING_DEPLOYMNT_BY_ENV/APP/OT

Summarizes the number of open packages and package lines that are currently pending deployment into environments.

The deployment information is broken down into a different category for each view:

- To see the distribution of the number of objects pending deployment across environments, use the view MPKGL_PENDING_DEPLOYMNT_BY_ENV.
- To see the same information distributed across applications, use MPKGL_PENDING_DEPLOYMNT_BY_APP.
- To see the same deployment information distributed across object types, use MPKGL_PENDING_DEPLOYMENT_BY_OT.

Sample

To obtain a quick look at the volume of deployments queued up at each environment defined in the system (for those with one or more pending deployments):

```
SELECT environment,
       total_count,
       unique_pkgs,
       unique_pkg_lines,
       unique_obj_types
FROM   mpkgl_pending_deploymnt_by_env;
```

The internal ID columns for Environments and Object Types (ENVIRONMENT_ID and OBJECT_TYPE_ID) can be used to link this view with other relevant views (for example, "MPKGL_PACKAGE_LINES" on page 24) to provide additional information in a report built from these views.

This view will not capture processes where the package line is waiting at an approval step that fires an immediate execution step.

MREL_DISTRIBUTIONS

Used to gather information about distributions of releases in Deployment Management. Contains columns to display the workflow used by a distribution, a distribution's status, whether a distribution has provided a feedback value to contained packages, and other information.

To relate information from this view with information from related views, the report designer can use the release identifier RELEASE_ID and distribution identifier DISTRIBUTION_ID to join with other views like "MREL_RELEASES" on page 29 and "MREL_DISTRIBUTION_ACTIONS" on the next page.

Also provided is the DIST_WORKFLOW_ID, which can be useful in joining to workflow views such as "MWFL_WORKFLOWS" on page 45 to include information about the workflows being used by a distribution.

MREL_DISTRIBUTION_ACTIONS

Used to gather information about current workflow steps for any given release distribution in Deployment Management. Contains columns to display the current status of a step, how long that step has been in the current status, whether the step is complete or resulted in an error, details about the step (source and destination Environment), and other relevant details.

To relate information from this view with information from related releases or release distributions, the report designer can use the release and distribution identifiers (RELEASE_ID and DISTRIBUTION_ID columns) to join with other standard views like "MREL_RELEASES" on page 29 and "MREL_DISTRIBUTIONS" on the previous page.

Sample

A report is needed that takes a release name input from the user running the report, and shows the details of all open distributions of the release:

```
SELECT release_name                              RELEASE_NAME,
       distribution_name                         DISTRIBUTION_NAME,
       dist_workflow_step_label || ': '|| action_name
                                                 ELIGIBLE_STEP,
       duration                                  DAYS_ELIGIBLE
FROM   mrel_distribution_actions
WHERE  status_type = 'ELIGIBLE'
GROUP BY release_name,
         distribution_name,
         dist_workflow_step_label || ': '|| action_name,
         duration
ORDER BY 1,2;
```

The column STATUS is the status name that is displayed in the status tab of distributions in the Deployment Management application.

The internal code STATUS_TYPE is provided to group these status names into logical groupings. For example, there may be many different statuses that all represent a COMPLETE status type. For example, the result value of any workflow step like Approved, Succeeded, Rejected, or Failed QA Test.

While STATUS may have many different possible values, STATUS_TYPE has only the following possible values:

- SUBMITTED
- IN_PROGRESS
- CLOSED_SUCCESS

- ELIGIBLE
- ERROR
- CLOSED_FAILURE
- PENDING
- COMPLETE
- CANCELLED

MREL_REFERENCES

Used to view the references of releases in Deployment Management.

There are several types of references for releases. If a package is part of a release, then there will be a reference for that package. Similarly, if a request is part of a release, then there will be a reference for that request. Releases can be designated as children or parents of other releases through the use of references. References are also used to attach documents to a release.

The RELATIONSHIP column in MREL_REFERENCES describes the relationship of the referenced item to the release that references it. This view also has columns for each of the entities that can be referenced to a release: other releases, requests, packages, attachments, and URLs.

For each record in MREL_REFERENCES, only one of these columns will have a value and the others will be NULL.

Sample

To retrieve a list of all references to a particular release:

```
SELECT referenced_release_id REL,
       referenced_package_id PKG,
       referenced_request_id REQ,
       attachment_name       ATTACHMENT,
       document_url          URL,
       relationship          RELATIONSHIP
FROM   mrel_references
WHERE  release_name = 'FIN Apps Prod Release';
```

Results

```
REL   PKG   REQ   ATTACHMENT           URL           RELATIONSHIP
----- ----- ----- -------------------- ------------- --------------------
30012                                                Parent Release
                  finAppsRelease.doc
42764                                                Contained in this Release
42765                                                Contained in this Release
42772                                                Contained in this Release
42773                                                Contained in this Release
42774                                                Contained in this Release
42778                                                Contained in this Release
```

MREL_RELEASES

Used to gather information about releases in Deployment Management. Contains columns to display the current status of a release, the number of distributions that have been deployed for a release, the manager, team, and group of a release, and other information.

To relate information from this view with information from related distributions, use the release identifier RELEASE_ID to join with "MREL_DISTRIBUTIONS" on page 26 or "MREL_REFERENCES" on the previous page.

Sample

To show details about releases that are part of the release team FIN Apps Prod Release, including all packages by relevant releases, and their statuses:

```
SELECT r.release_name    RELEASE,
       r.release_status  REL_STATUS,
       p.package_number  PKG_NUMBER,
       p.package_status  PKG_STATUS
FROM   mpkg_packages p,
       mrel_references ref,
       mrel_releases r
WHERE  r.release_team = 'FIN Apps Prod Release'
AND    r.release_id = ref.release_id
AND    p.package_id = ref.referenced_package_id
ORDER BY r.release_name, p.package_number;
```

Results

```
RELEASE        REL_STATUS         PKG_NUMBER      PKG_STATUS
-------------- ------------------ --------------- -------------------
Apply to Test  Code Freeze        43002           Ready for Release
Apply to Test  Code Freeze        43005           In Progress
Apply to Test  Code Freeze        43007           Ready for Release
```

The column RELEASE_STATUS in MREL_RELEASES is the status displayed in the releases screen in the Deployment Management application. The RELEASE_STATUS column has the following possible values:

- New
- Code freeze
- Open
- Closed

Demand Management Views

Demand Management views provide information specific to Demand Management. For example, MREQ_OPENED_CLOSED_BY_TYPE_D provides summary information for request submission

and completion activity, broken down by request type and by calendar day. MREQ-REFERENCES provides information about references related to Demand Management requests.

MREQ_<*Request Type Name*>

Contains request type-specific information.

When the Reporting Meta Layer is synchronized, a view is created for every request type defined in the system. The name of each view is defined on the request type screen in the Meta Layer View field. It defaults to a prefix MREQ_ and a suffix that defaults to the first 20 alphanumeric characters of the corresponding request type name. For example, if there are three request types defined in Demand Management named Support Ticket, Bug, and Work Order, then three corresponding Meta Layer views would exist: MREQ_SUPPORT_TICKET, MREQ_BUG, and MREQ_WORK_ORDER.

The view columns are identical to those of the general MREQ_ALL_REQUESTS view (including the global request user data fields), and they also include additional columns for each custom request detail field for the request type. This allows a report designer to create a report that implements business logic based on customer-defined request detail fields.

Sample 1

For example, consider the Work Order request type. This request type might have custom detail fields with tokens like CUSTOMER, TIME_ESTIMATE, and ACTUAL_TIME. The corresponding view MREQ_WORK_ORDER would contain columns with these names:

```
SQL> desc mreq_work_order;
```

Results 1

```
Name                           Null?    Type
------------------------------ -------- ----
REQUEST_ID                     NOT NULL NUMBER
REQUEST_STATUS                 NOT NULL VARCHAR2(80)
...
CUSTOMER                                VARCHAR2(200)
TIME_ESTIMATE                           VARCHAR2(200)
ACTUAL_TIME                             VARCHAR2(200)

...
```

Sample 2

A report is needed that will list information about work order requests in which the actual time was more than one day longer than the estimated time.

An SQL query such as the following would handle this:

```
SELECT request_number REQUEST_NUM,
       status_name CURRENT_STATUS,
       customer CUSTOMER,
```

```
       (actual_time – time_estimate) EXTRA_DAYS_WORKED
FROM   mreq_work_order
WHERE  time_estimate IS NOT NULL
AND    actual_time IS NOT NULL
AND    (actual_time – time_estimate) > 1
ORDER BY request_number;
```

MREQ_CONTACTS

Contains all fields for contacts defined in Demand Management. Contains all relevant pieces of information about a contact, including a denormalized username (if present) and a column for each Contact User Data field defined in the system. The column name for each Contact User Data field is the same as the token name for that field.

A subset of the information provided here is also present in the request views MREQ_REQUESTS and MREQ_<*Request Type Name*>.

Sample 1

```
SELECT full_name NAME,
       phone_number PHONE_NUMBER,
       email_address EMAIL
FROM   mreq_contacts
WHERE  enabled_flag = 'Y';
```

Sample 2

If there are Contact User Data fields defined, the token for each field will appear as a separate column in MREQ_CONTACTS.

For example, two Contact User Data fields have been defined to track additional contact information, with tokens PAGER_NUMBER and HOME_PHONE_NUMBER. Two columns with the same names would be present in MREQ_CONTACTS:

```
SQL> desc mreq_contacts;
```

Results 2

```
Name                           Null?    Type
------------------------------ -------- ----
LAST_NAME                      NOT NULL VARCHAR2(30)
FIRST_NAME                     NOT NULL VARCHAR2(30)
...
PAGER_NUMBER                            VARCHAR2(200)
HOME_PHONE_NUMBER                       VARCHAR2(200)
ENABLED_FLAG                   NOT NULL VARCHAR2(1)
CREATION_DATE                  NOT NULL DATE
...
```

Sample 3

Building on Sample 1 by using "MREQ_REQUESTS" on page 37, consider designing a report to print the full name, pager, and work numbers of all users who are assigned as backup users on requests that have been open for more than 5 days.

An SQL statement to achieve this type of information might look as follows:

```
SELECT r.backup_username USERNAME,
       c.full_name NAME,
       c.pager_number PAGER_NUMBER,
       c.phone_number WORK_NUMBER,
FROM   mreq_contacts c,
       mreq_requests r
WHERE  c.enabled_flag = 'Y'
AND    r.backup_username = c.username (+)
AND    r.close_date IS NULL
AND    r.cancel_date IS NULL
AND    r.submission_date IS NOT NULL
AND    (sysdate – r.submission_date) > 5;
```

MREQ_CHANGES

When a field is being audited, a record is stored in the PPM database every time the value in that field changes on any open Request. This audit history can be important to business decision-making.

MREQ_CHANGES allows a report to display and drive off of changes to request fields. This view exposes the audit trail for the request header and detail fields. It contains columns for the old and new values, and the field prompts and tokens.

Sample

To report on the frequency at which the request priority is changed from any value to Critical, an SQL statement such as the following can be used:

```
SELECT m.calendar_month MONTH,
       c.old_field_value OLD_VALUE,
       count(*) NUM_CHANGED
FROM   mreq_changes c,
       krml_calendar_months m
WHERE  c.field_prompt = 'Priority'
AND    c.new_field_code = 'C'
AND    c.change_date >= m.start_date
AND    c.change_date < m.end_date
GROUP BY m.calendar_month, c.old_field_value
ORDER BY 1, 2;
```

In the WHERE clause of this statement that we are testing, the NEW_FIELD_CODE is used instead of the NEW_FIELD_VALUE. Either would work.

C is the code for the Critical priority; this statement could also have been written `WHERE c.new_field_value = 'Critical'`.

The validation for the request priority field contains the hidden and visible values for this field. Consult this validation in the Validations window for verification of these values.

Consider a slight extension to the previous SQL statement. If it was necessary to limit this information to a specific request type, an additional AND condition could be used: AND `c.request_type = '<Name>'`.

MREQ_NOTES

Provides access to the notes for all requests in Demand Management.

Notes are stored in an Oracle LONG database column; to prevent an overload of information this was presented in a separate Meta Layer view, making it less likely to design a report that inadvertently returns too much data.

To query request notes, join this view with a request view ("MREQ_REQUESTS" on page 37, or a request type-specific view "MREQ_<Request Type Name>" on page 30).

Sample

To retrieve a list of the notes for all open requests of the Bug request type, that have Critical priority, use the following logic in an SQL statement:

```
SELECT r.request_number REQ_NUM,
       n.notes NOTES
FROM   mreq_bug r,
       mreq_notes n
WHERE  r.priority = 'Critical'
AND    r.request_id = n.request_id;
```

MREQ_OPENED_CLOSED_BY_DETAIL_D/M

Use this view to assess daily request throughput, and to help indicate trends in open requests over time. These views provide information for request submission and completion activity (throughput), broken down by:

- day or month
- combinations of request type, application, department, priority, and assigned-to user

Thus allowing access to more information than "MREQ_OPENED_CLOSED_BY_DETAIL_D/M" above or "MREQ_OPENED_CLOSED_BY_TYPE_D/M " on the next page.

Results from a query of this view contain records only for days or months on which there were requests opened or closed.

Sample

This creates a report to examine throughput of all work order request types for the IT development department:

```
SELECT activity_date,
       application,
       priority,
       total_opened,
       total_closed,
       num_still_open,
       avg_comp_time_opened,
       avg_comp_time_closed
FROM   mreq_opened_closed_by_detail_d
WHERE  activity_date BETWEEN '01-APR-01' AND '05-APR-01'
AND    request_type_name = 'Work Order'
AND    department = 'Development'
ORDER BY activity_date;
```

To get a breakdown by month, replace `activity_date` with `activity_month` and `mreq_opened_closed_by_detail_d` with `mreq_opened_closed_by_detail_m`.

Results

Date	Application	Priority	Total Open	Total Closed	Avg Num Still Open	Avg Comp Time Open	Avg Comp Time Closed
01-APR-01	Manufacturing	Normal	0	2	0		26.06
01-APR-01	Financials	Normal	0	2	0		31.07
01-APR-01	Work-in-process	Normal	0	2	0		22.74
02-APR-01	Documentation	Normal	0	1	0		21.78
03-APR-01	Bill-of-materials	Low	0	1	0		41.01
03-APR-01	Bill-of-materials	Normal	0	1	0		26.09
04-APR-01	Bill-of-materials	Low	0	1	0		47.35
04-APR-01	Bill-of-materials	Normal	0	2	0		20.60
04-APR-01	Configuration	Normal	0	1	0		63.18
04-APR-01	Workflow	Low	0	2	0		20.70
05-APR-01	Manufacturing	Low	0	2	0		36.90
05-APR-01	Work-in-process	Normal	0	1	0		4.35

MREQ_OPENED_CLOSED_BY_TYPE_D/M

The Reporting Meta Layer views MREQ_OPENED_CLOSED_BY_TYPE_D and MREQ_OPENED_CLOSED_BY_TYPE M provide summary information for request submission and completion activity, broken down by request type and by calendar day or month.

Use to assess daily or monthly request throughput, and to help indicate trends in request processing over time.

These views contain columns such as:

- the net change in number of open requests during that day or month
- the number of requests still open at the end of the day or month
- the average time to completion, in days or months

for requests opened in that day or month and which have already been closed.

Results from a query of this view contain records only for days or months in which there were requests opened or closed.

Sample

The following SQL query can be used as a basis for a report that summarizes all request submission and completion activity, per month, over a range of dates:

```
SELECT *
FROM   mreq_opened_closed_by_type_m
WHERE  activity_month BETWEEN '01-MAR-01' AND '01-APR-01'
ORDER BY activity_month;
```

To get a breakdown by day, replace `activity_month` and `mreq_opened_closed_by_type_m` with `activity_date` and `mreq_opened_closed_by_type_d`.

Results

```
                                                                  Avg      Avg
                                                          Num    Comp     Comp
                                    Tot    Total     Net Still    Time     Time

REQUEST_TYPE_NAME           Month  Open  Closed  Change  Open    Open  Closed
--------------------------- ------ ----- ------- ------- ----- ------- ---
Customer Access             01-MAR-01    53      52       1     0     .07     .01
HR Job Requisition          01-MAR-01    16      17      -1     6   38.84   48.93
HR New Hire Process         01-MAR-01    13      10       3     1   40.35   25.61
Product Bug                 01-MAR-01    83     232    -149    60    7.64  299.71
Product Patch               01-MAR-01     8       0       8     8
Purchase Request            01-MAR-01    18      24      -6     0   13.72    9.13
Services Work Order         01-MAR-01    17       3      14    17     .00   33.59
Training Approval Request   01-MAR-01   336     369     -33     9    8.46   19.59
Vacation Request            01-MAR-01   115      72      43    33   25.87   27.84
Customer Access             01-APR-01    15       6       9    12   11.63  150.55
HR Job Requisition          01-APR-01     5       6      -1     0     .81   36.94
HR New Hire Process         01-APR-01    27       6      21    27    9.87  255.96
Product Bug                 01-APR-01    36      35       1     2     .21     .29
...
```

For more detailed request information filtered by common request header fields like Application, Department, Priority, and Assigned-to User, use the detail summary views MREQ_OPENED_CLOSED_BY_DETAIL_D and MREQ_OPENED_CLOSED_BY_DETAIL_M.

MREQ_PENDING_REQUESTS

Used to create a report that shows the volume of open requests for any given request type in Demand Management.

This report can be used to get information about ongoing request processing work. It shows a summary of requests currently open for a specific Demand Management request type (for example, total number or average age), and information showing how many requests have been opened and closed in the current week and current month.

MREQ_PENDING_REQUESTS is aggregated across all requests.

In addition to overall totals of open requests, this view breaks down the information by priority (using the Priority header field). This is done because priority is usually the most important breakdown of load information. Data is grouped into three priority groupings—P1, P2, and P3, which map to the three highest-priority levels defined.

Sample

A QA manager has three types of requests to handle, running through three separate processes. The manager needs a report that will show current work volume for each of these request types, to help prioritize work and identify bottlenecks.

If the three request types are named MFG bug report, FIN bug report, and APPS enhancement request, the following SQL query can be used as a basis for a report to display the desired information:

```
SELECT request_type          Request_Type,
       open_requests         Open_Reqs,
       avg_age_open_requests Avg Age,
       p1_open_requests      P1 Open Reqs,
       p2_open_requests      P2 Open Reqs
FROM   mreq_pending_requests
WHERE  process_name IN
       ('MFG bug report',
       'FIN bug report',
       'APPS enhancement request');
```

Results

```
                                      P1    P2
                           Open  Avg  Open  Open
REQUEST_TYPE               Reqs  Age  Reqs  Reqs
------------------------- ----- ---- ----- -----
MFG bug report               98    3    21    77
FIN bug report               39    4    14    25
APPS enhancement request    140   12     8   132
```

This view ignores requests that have not been submitted.

MREQ_REQUESTS

The most general view into request transaction data.

A blind query (`SELECT * FROM mreq_requests;`) will return one row for each request present in the system, including closed requests.

The view columns map to the request fields that are common to all request types (for example, priority, department, application, assigned-to user, and contact information). There are also columns for the status of a request and the dates on which it was submitted, closed, or cancelled.

Because global request user data fields are present on all requests, there is also a view column for each global request user data field that is defined. The column name for each global request user data field is the same as the token name for that field.

Use this view when writing a report to present general request information without respect to a particular request type.

To build reports that make use of custom detail fields of a particular request type, see "MREQ_<Request Type Name>" on page 30.

Sample 1

To get information about the number of open requests in the system and to whom they are assigned:

```
SELECT assigned_to_username ASSIGNED_USER,
       COUNT(*) NUM_OPEN
FROM   mreq_requests
WHERE  close_date IS NULL
AND    cancel_date IS NULL
AND    submission_date IS NOT NULL
GROUP BY assigned_to_username
ORDER BY 1;
```

Results 1

```
ASSIGNED_USER                  NUM_OPEN
---------------------------------------
...
rfrazier                             13
rjeffries                             1
rjones                               28
rnelson                               9
rsmith                                3
...
```

Sample 2

Or consider a similar query with the results grouped by the request type, to see how many requests of each type are open:

```
SELECT request_type_name REQUEST_TYPE,
       COUNT(*) NUM_OPEN
FROM   mreq_requests
WHERE  close_date IS NULL
AND    cancel_date IS NULL
AND    submission_date IS NOT NULL
GROUP BY request_type_name
ORDER BY 1;
```

Results 2

```
REQUEST_TYPE                        NUM_OPEN
------------------------------------------
HR Job Requisition                        37
HR New Hire Process                       11
Press Release                              3
Product Patch                             33
Purchase Request                          11
Services Work Order                       81
Training Approval Request                115
Vacation Request                          56
```

Sample 3

Consider the case where a global request user data field has been defined to capture the username of a backup user responsible for each request.

The token name for this field is BACKUP_USERNAME. Therefore, in this view there would be a column named BACKUP_USERNAME:

```
SQL> desc mreq_requests;
```

Results 3

```
Name                           Null?    Type
------------------------------ -------- ----
REQUEST_ID                     NOT NULL NUMBER
REQUEST_DESCRIPTION            NOT NULL VARCHAR2(240)
SUBMISSION_DATE                NOT NULL DATE
REQUEST_STATUS                 NOT NULL VARCHAR2(80)
...
CANCEL_DATE                    NOT NULL DATE
BACKUP_USERNAME                         VARCHAR2(200)
REQUEST_TYPE_NAME                       VARCHAR2(80)
REQUEST_SUBTYPE_NAME                    VARCHAR2(80)
...
```

Sample 4

The new column can be used to drive a report, if necessary. For example, to report on requests that have been open for more than five days and assigned to a particular backup user:

```
SELECT backup_username BACKUP_USER,
       assigned_to_username ASSIGNED_USER,
       COUNT(*) NUM_OLD_REQS
FROM   mreq_requests
WHERE  backup_username = '<ValidUsername>'
AND    close_date IS NULL
AND    cancel_date IS NULL
AND    submission_date IS NOT NULL
AND    (sysdate – submission_date) > 5
GROUP BY backup_username, assigned_to_username
ORDER BY 1, 2;
```

This query also displays the name of the original user to whom the request was assigned.

MREQ_REQUEST_ACTIONS

Used to gather information about all workflow actions for any given request in Demand Management. Contains columns to display the result status of each step, how long it took to complete, details about the step (for example, source and destination environment), and other relevant details. It also adds the submission (Process Open) and completion (Process Close) of a request as pseudo workflow step actions, displaying the entire life cycle of the request in a single view.

This view can be used directly to view the full transaction history of a request, or it can be used as a basis for more complex reports showing, for example, throughput at specific request steps.

To relate information from this view with information from relevant requests, use the request identifier REQUEST_ID to join with "MREQ_REQUESTS" on page 37 or "MREQ_<Request Type Name>" on page 30.

Sample 1

Consider a report that takes a request ID as input from the person running the report, and shows all transactions for that request. To include the name of the step, the date an action was taken, the result, and how long the step stayed active before the action was taken, you could write a query similar to:

```
SELECT action_name,
       action_date,
       action_result,
       duration
FROM   mreq_request_actions
WHERE  request_id = <DesiredID>
ORDER BY action_date;
```

Results 1

```
                              Action    Action
Process Step                  Date      Result            Duration
----------------------------- --------- ----------------- --------
Open                          26-APR-01 Released               .00
Check Priority                26-APR-01 Normal                 .00
SA - Check Prodcut            26-APR-01 NULL result            .00
CL - Check issue assignment   26-APR-01 aaslani                .00
Work In Progress              15-MAY-01 Resolved             18.72
Feedback                      20-MAY-01 Timeout               5.00
Close                         20-MAY-01 Closed [Success]       .00
Request resolved              20-MAY-01 Succeeded              .00
```

Sample 2

Consider a work order request type that has a Customer field with token CUSTOMER.

The name of the corresponding request view will be MREQ_WORK_ORDER based on the general view "MREQ_<Request Type Name>" on page 30. A report is needed to show all work order requests that are eligible for fjohnson to act on, broken down by customer:

```
SELECT wo.customer                CUSTOMER,
       wo.request_id              REQ_NUM,
       ra.request_workflow_step_label || ': '|| ra.action_name
                                  ELIGIBLE_STEP,
       ra.duration                DAYS_ELIGIBLE
FROM   mreq_work_order wo,
       mwfl_step_security_users ssu,
       mreq_request_actions ra
WHERE  ra.status_type = 'ELIGIBLE'
AND    ssu.workflow_step_id = ra.workflow_step_id
AND    ssu.username = 'fjohnson'
AND    ra.request_id = wo.request_id
ORDER BY 1,2,3,4;
```

The format of the ELIGIBLE_STEP column being selected, which will return a value similar to `12.3.1: Review by Lead`. MWFL_STEP_SECURITY_USERS (see "MWFL_STEP_SECURITY_GROUPS and MWFL_STEP_SECURITY_USERS" on page 44) is used to determine if a specified user is authorized for a specified workflow step.

Additional considerations:

- The column STATUS is the status name that is displayed in the status tab of requests in the Demand Management application.

  The internal code STATUS_TYPE is provided to group these status names into logical groupings. For example, there may be many different statuses that all represent a COMPLETE status type (the result value of any workflow step—Approved, Succeeded, Rejected, or Failed QA Test). While STATUS may have many different possible values, STATUS_TYPE has only the following possible values:

- SUBMITTED
- IN_PROGRESS
- CLOSED_SUCCESS
- ELIGIBLE
- ERROR
- CLOSED_FAILURE
- PENDING
- COMPLETE
- CANCELLED

MREQ_REFERENCES

Used to view the references of requests in Demand Management.

There are several types of references for requests. If a request is part of a release, then there will be a reference for that release. If a request is a parent or child of another request, then there will be a reference for that request. References are also used to attach documents to a request.

The RELATIONSHIP column in MREQ_REFERENCES describes the relationship of the referenced item to the request that references it. This view also has columns for each of the entities that can be referenced to a request—other requests, packages, projects, tasks, releases, attachments, and URLs.

For each record in MREQ_REFERENCES, only one of these columns will have a value and the others will be NULL.

Sample

The following SQL can be used to retrieve a list of all references to a particular request:

```
SELECT referenced_package_id PKG,
       referenced_project_id PROJ,
       referenced_request_id REQ,
       referenced_release_id REL,
       referenced_task_id    TASK,
       attachment_name       ATTACHMENT,
       document_url          URL,
       relationship          RELATIONSHIP
FROM   mreq_references
WHERE  request_number = '54872';
```

Results

```
PKG  PROJ  REQ   REL  TASK   ATTACHMENT       URL         RELATIONSHIP
--- ----- ----- ----- ----- ------------ ---------- --------------------------
                     43301                            Contains this Request
                  43304                             Contains this request
```

```
30043                                                   Child of this Request
         52383                                          Parent of this Request
                          screenShot.doc
```

MREQ_REQUEST_HEADER_TYPES

Accesses configuration details of request header types in Demand Management.

In some cases a report designer might need to include request header type information in a report, and can join the REQUEST_HEADER_TYPE column in this view with the same column in the MREQ_REQUEST_TYPES view, and in general request views ("MREQ_REQUESTS" on page 37 and "MREQ_<Request Type Name>" on page 30).

PPM supports user data on request header types. All defined request header type user data fields are represented in MREQ_REQUEST_HEADER_TYPES view; there is a column for each request header type user data field.

The column name for each request header type user data field is the same as the token name for that field.

Sample

A user data field with token name OWNER is defined for request header types, to keep track of a PPM administrator responsible for maintaining each request header type configuration.

A corresponding view column named OWNER will be present in MREQ_REQUEST_HEADER_TYPES view:

```
SQL> desc mreq_request_header_types;
```

Results

```
Name                            Null?    Type
------------------------------- -------- ----
REQUEST_HEADER_TYPE             NOT NULL VARCHAR2(80)
REQUEST_HEADER_TYPE_DESC                 VARCHAR2(240)
...
ACCELERATOR_NAME                NOT NULL VARCHAR2(80)
OWNER                                    VARCHAR2(200)
CREATION_DATE                   NOT NULL DATE
CREATED_BY_USERNAME             NOT NULL VARCHAR2(30)
LAST_UPDATE_DATE                NOT NULL DATE
```

MREQ_REQUEST_TYPES

Accesses configuration details of request types in Demand Management.

You can include request type information in a report by joining the REQUEST_TYPE column in this view with the same column in the general request views ("MREQ_REQUESTS" on page 37 and "MREQ_<Request Type Name>" on page 30).

PPM supports user data on request types. All defined request type user data fields are represented in MREQ_REQUEST_TYPES view; there is a column for each request type user data field. The column name for each request type user data field is the same as the token name for that field.

Sample 1

A user data field with token name OWNER is defined for request types, to keep track of a PPM administrator responsible for maintaining each request type configuration.

A corresponding view column named OWNER will be present in MREQ_REQUEST_TYPES view:

```
SQL> desc mreq_request_types;
```

Results 1

```
Name                           Null?    Type
------------------------------ -------- ----
REQUEST_TYPE                   NOT NULL VARCHAR2(30)
REQUEST_TYPE_DESCRIPTION       NOT NULL VARCHAR2(240)
...
INITIAL_STATUS                 NOT NULL VARCHAR2(80)
RESTRICTION                    NOT NULL VARCHAR2(30)
OWNER                                   VARCHAR2(200)
CREATION_DATE                  NOT NULL DATE
CREATED_BY_USERNAME            NOT NULL VARCHAR2(30)
...
```

Sample 2

An SQL query based on this view might be used to determine how many requests were created prior to a configuration change for a particular request type.

For example, a request type named Work Order has undergone a significant configuration change that invalidates open work order requests that were created before the change. Therefore, a report is needed to determine the status of open work order requests that were created before the changes:

```
SELECT wo.request_id          REQUEST_NUM,
       wo.request_status      CURRENT_STATUS,
       wo.request_description DESCRIPTION
FROM   mreq_work_order wo,
       mreq_request_types rt
WHERE  wo.creation_date < rt.last_update_date
AND    rt.request_type = 'Work Order'
ORDER BY 1;
```

Notice that we do not have to join the explicit request type name to the view MREQ_WORK_ORDER, because it is already implicit in the view definition—only work order requests are returned from that view.

MREQ_TABLE_COMPONENT

Contains table component data for request detail fields with validations.

Other Views

Other views provide information about PPM entities like workflows and security groups. For example, MWFL_STEP_SECURITY_USERS lists all users with authority to act on a given workflow step through static security group or user linkage, as defined in the workflow step window in the Workflow workbench.

MWFL_STEP_SECURITY_GROUPS and MWFL_STEP_SECURITY_USERS

Used to get information about PPM users or security groups linked to Workflow steps.

- **MWFL_STEP_SECURITY_USERS** lists all users with authority to act on a given workflow step through static security group or user linkage, as defined in the workflow step dialog in the Workflow workbench.
- **MWFL_STEP_SECURITY_GROUPS** lists all security groups with authority to act on a step through static security group linkage.

These views can be useful for reporting on specific key workflow steps to show more detailed information that may not be available in the more general activity management views.

Sample

A report is needed to show all requests in Demand Management belonging to a given user who is eligible for one or more approval workflow steps. The view "MWFL_WORKFLOW_STEPS" on page 46 can be used to show the workflow steps that are approval steps, and the view "MREQ_REQUEST_ACTIONS" on page 39 will provide the request information for eligible steps:

```
SELECT ssu.username                ELIGIBLE_USER,
       ra.request_id              REQUEST_NUM,
       ra.request_workflow_step_label || ': '|| ra.action_name
                                  ELIGIBLE_STEP,
       ra.duration                DAYS_ELIGIBLE
FROM   mwfl_step_security_users ssu,
       mwfl_workflow_steps ws,
       mreq_request_actions ra
WHERE  ra.status_type = 'ELIGIBLE'
AND    ws.step_type = 'Approval'
AND    ra.workflow_step_id = ws.workflow_step_id
```

```
AND    ssu.workflow_step_id = ra.workflow_step_id
ORDER BY 1,2,3,4;
```

In this query, the workflow step identifier WORKFLOW_STEP_ID was used to join MWFL_STEP_SECURITY_USERS with the view "MREQ_REQUEST_ACTIONS" on page 39, to relate request workflow step information.

Dynamic workflow step security defined by tokens is not included in these views.

MWFL_WORKFLOWS

Use to access basic configuration details of workflows.

To include workflow information in a report, you can join the WORKFLOW_ID column in this view with the same column in Workflow transaction views (for example, "MREQ_REQUEST_ACTIONS" on page 39). The view MWFL_WORKFLOW has columns for the main workflow definition fields present on the first tab of the workflow detail window in the PPM Workbench, and also includes a column for each workflow user data field defined in the system.

Sample

If the system has three workflow user data fields defined, this view will contain three columns that use the user data fields' token names as view column names. If these three user data fields have the tokens DEPARTMENT, ADMINSTRATOR_USERNAME, and WORKFLOW_MANAGER, then the MWFL_WORKFLOWS view would contain three columns with these names:

```
SQL> desc mwfl_workflows;
```

Results

```
Name                           Null?    Type
------------------------------ -------- ----
WORKFLOW                       NOT NULL VARCHAR2(80)
WORKFLOW_DESCRIPTION                    VARCHAR2(240)
...
SUB_WORKFLOW_FLAG                       VARCHAR2(1)
DEPARTMENT                              VARCHAR2(200)
ADMINISTRATOR_USERNAME                  VARCHAR2(200)
WORKFLOW_MANAGER                        VARCHAR2(200)
CREATED_BY_USERNAME            NOT NULL VARCHAR2(30)
CREATION_DATE                  NOT NULL DATE
...
```

By default this view returns both reference and non-reference workflows in the system.

PPM provides reference copies of some workflows, which are disabled and not usable by PPM transactions, and as such are rarely of reporting interest. The view column REFERENCE_FLAG can be used to filter results. To only show active, non-reference workflows while using MWFL_WORKFLOWS view, include `REFERENCE_FLAG = 'N'` in the query.

MWFL_WORKFLOW_STEPS

Provides configuration details of workflow steps.

In some cases a report designer might need to present workflow step information in a report. The report designer can join this view with other workflow views through the key values WORKFLOW_STEP_ID and WORKFLOW_ID.

This view also includes a column for each workflow step user data field defined in the system.

Sample 1

A user data field has been defined for workflow steps to provide a categorization. This Step Category field, has a token CATEGORY. Therefore, in this view there will be a CATEGORY column:

```
SQL> desc mwfl_workflow_steps;
```

Results 1

```
Name                            Null?    Type
------------------------------- -------- ----
WORKFLOW_STEP                   NOT NULL VARCHAR2(80)
WORKFLOW_STEP_NUMBER            NOT NULL NUMBER
...
PARENT_REQUEST_TYPE_STATUS               VARCHAR2(30)
CATEGORY                                 VARCHAR2(200)
CREATED_BY_USERNAME             NOT NULL VARCHAR2(30)
CREATION_DATE                   NOT NULL DATE
...
```

This type of information can be used to drive reports built using the Meta Layer.

Sample 2

To continue the example, the CATEGORY user data field has values of Normal, Test Gate, and Prod Gate to give an indication of the nature of each step. A report is needed to show if fjohnson is eligible for any Deployment Management workflow steps that are critical gateways to production (that is, in the Prod Gate category), and how long they have been eligible:

```
SELECT pla.package_number         PACKAGE_NUM,
       pla.line_number            LINE_NUM,
       pla.line_workflow_step_label || ': '|| pla.action_name
                                  ELIGIBLE_STEP,
       pla.duration               TIME_ELIGIBLE,
       ws.workflow                WORKFLOW
FROM mwfl_step_security_users ssu,
     mwfl_workflow_steps ws,
     mpkgl_package_line_actions pla
WHERE  pla.status_type = 'ELIGIBLE'
AND    ws.category = 'Prod Gate'
```

```
AND    ws.workflow_step_id = pla.workflow_step_id
AND    ws.workflow_step_id = ssu.workflow_step_id
AND    ssu.username = 'fjohnson';
```

In this example, MWFL_WORKFLOW_STEPS was joined to the view "MPKGL_PACKAGE_LINE_ACTIONS" on page 24 with the WORKFLOW_STEP_ID column.

Additional information:

- By default this view returns both reference and non-reference workflow steps in the system. PPM provides reference copies of some workflow steps, which are disabled and not usable by PPM transactions, and as such are rarely of reporting interest. The view column REFERENCE_FLAG can be used to filter results. To show only active, non-reference workflow steps while using the MWFL_WORKFLOW_STEPS view, include `REFERENCE_FLAG = 'N'` in the query.
- The type of each workflow step is accessible through the column STEP_TYPE. The following types of workflow steps are available:
  - Condition
  - Decision
  - Execution
  - Workflow

KCRT_PARTICIPANT_CHECK_V

Used to enforce request participant security in the data presented in reports.

A query of KCRT_PARTICIPANT_CHECK_V will return the requests in Demand Management in which a particular PPM user is a participant. This view can be joined into report queries to check whether the user running the report is a participant of requests that are enforcing participant-only viewing restriction.

Sample

Consider a report that is to return the description of open requests in Demand Management. To restrict reported information to only those requests in which the user running the report is a participant, so that it requires a valid username as an input field.

Note: The methodology and support for this type of report input will vary between reporting systems. Consult the documentation for the reporting system you are using for specific instructions.

Assuming the input username was available as REPORT_USER, you can include the following SQL fragment in the report query:

```
...
FROM   kcrt_participant_check_v kpc
WHERE  kpc.username = REPORT_USER
```

```
AND    kpc.request_id = ...
...
```

Including this fragment in the full SQL statement might look as follows:

```
SELECT r.request_id,
       r.request_status,
       r.request_description
FROM   mreq_requests r,
       kcrt_participant_check_v kpc
WHERE  r.request_status not in ('Cancelled','Closed')
AND    kpc.username = 'fjohnson'
AND    kpc.request_id = r.request_id;
```

If a request type does not enforce request participant security, then all requests of this request type will be returned by KCRT_PARTICIPANT_CHECK_V as viewable.

KDLV_PARTICIPANT_CHECK_V

Used to enforce package participant security in the data presented in reports.

A query of KDLV_PARTICIPANT_CHECK_V will return the packages in Deployment Management of which a particular user is a participant. This view can be joined into report queries to check whether the user running the report is a participant of packages that are enforcing participant-only viewing restriction.

Sample

Consider a report that is to return the description of open packages in Deployment Management. To restrict reported information to only those packages that the user running the report is a participant of, you must design the report so that it requires a valid username as an input field.

Note: The methodology and support for this type of report input will vary between reporting systems. Consult the documentation for the reporting system you are using for specific instructions.

Assuming the input username is available as REPORT_USER, include the following SQL fragment in the report query:

```
...
FROM   kdlv_participant_check_v kpc
WHERE  kpc.username = REPORT_USER
AND    kpc.package_id = ...
...
```

Including this fragment in the full SQL statement might look as follows (with an example username of fjohnson):

```
SELECT p.package_id,
       p.package_status
```

```
       p.package_description
FROM   mpkg_packages p,
       kdlv_participant_check_v kpc
WHERE  p.package_status not like 'Closed%'
AND    kpc.username = 'fjohnson'
AND    kpc.package_id = p.package_id;
```

If a Deployment Management workflow does not enforce package participant security, then all packages using this workflow will be returned by KDLV_PARTICIPANT_CHECK_V as viewable.

KRML_CALENDAR_DAYS and KRML_CALENDAR_MONTHS

These tables, included in the RML, contain sequential dates. KRML_CALENDAR_DAYS contains a record for every day from January 1, 1998, to mid-2011. KRML_CALENDAR_MONTHS contains a record for every month from January 1998 to mid-2011.

These tables can be used to provide a date for organizing and grouping the results of queries.

Sample

A report needs to contain summary information for the number of errors for step 2 in the FIN dev-test-prod workflow, broken down by month. The calendar table KRML_CALENDAR_MONTHS can be used to provide the month-by-month breakdown to join with the ACTIVITY_DATE column in the view "MWFL_STEP_ACTIVITIES" on page 12:

```
SELECT m.calendar_month MONTH,
       sum(sa.error)    NUM_ERRORS
FROM   krml_calendar_months m,
       mwfl_step_activities sa
WHERE  sa.workflow = 'FIN dev-test-prod'
AND    sa.workflow_step_number = 2
AND    sa.activity_date >= m.start_date
AND    sa.activity_date < m.end_date
GROUP BY m.calendar_month
ORDER BY 1;
```

The comparison of ACTIVITY_DATE to the START_DATE and END_DATE of the calendar month. This can be very useful for grouping discrete activity dates into aggregate time buckets.

Appendix B: Synchronization Messages

"Table B-1. RML synchronization messages, continued" on page 59 lists and describes the possible RML synchronization messages.

Table B-1. RML synchronization messages

Message ID	Display Text	Cause	Action/Solution
KNTA-10504	No description available for this view. View description is not provided in template <*TemplateName*>.	The template file for the specified RML view does not contain a description. (For Micro Focus-supplied templates, this should never occur.)	RML view templates are stored on the PPM Server file system in the $knta_home/rml/templates directory. Edit the template file with a text editor and add the text "[VIEW_DESCRIPTION =]" to the template header.
KNTA-10505	Could not determine context type for Meta Layer view template file <*TemplateName*>.	None of [VIEW_NAME_PREFIX] and [STATIC_VIEW_NAME] is specified in the template.	Specify [STATIC_VIEW_NAME] for a template with static context or specify [VIEW_NAME_PREFIX] for a template with dynamic context.
KNTA-10506	Could not determine Meta Layer view name. <*ViewName*>	Either the system could not find the corresponding parameter set or the Meta Layer view name root is not specified.	Make sure the "Meta Layer View" field is completed for the corresponding Request Type, Object Type, or User Data.
KNTA-10507	Fail to create comment for "<*EntityType1*>". <*EntityID1*> DDL statement: <*statement*>.	The system grant is granted to the base PPM schema at the time of installation and upgrade.	Varies depending on what is causing the particular issue.

Table B-1. RML synchronization messages, continued

Message ID	Display Text	Cause	Action/Solution
KNTA-10509	View ID is not returned after assessment.	Internal error. An unexpected error occurred.	Contact Software Support to report the problem.
KNTA-10510	"Create or replace view..." statement failed.	The SQL statement to create an RML view failed.	Examine the Oracle error. There may be several reasons why the SQL statement would fail. Perhaps the original SQL specified in the RML template has syntax errors, or some parameters in the SQL were either missing or resulted in an invalid SQL statement.
KNTA-10511	Oracle error when dropping view <*ViewName*>.	The specified view could not be dropped.	Examine the Oracle error to determine why the view could not be dropped, and consult the DBA for the PPM database to help fix the problem.
KNTA-10512	The configuration that corresponds to PARAMETER_SET_CONTEXT_ID = <*entityID*> no longer exists.	Either a request type, object type, or context-sensitive user data field set has been deleted.	The corresponding view will be dropped from the RML.
KNTA-10513	A new RML view template, <*TemplateName*>, was found.	The specified new RML view template was found in the $knta_home/rml/templates directory on the PPM Server.	New RML views will be generated based on this template.

Table B-1. RML synchronization messages, continued

Message ID	Display Text	Cause	Action/Solution
KNTA-10514	Custom field definitions were updated since the last RML synchronization.	Modifications to the custom field definitions for the corresponding request type, object type, or user data field set were made since the last RML synchronization event.	The associated RML views will be updated with the current field information.
KNTA-10515	View name changed from <*OldViewName*> to <*NewViewName*> since the last synchronization.	The RML view name was changed from <*OldViewName*> to <*NewViewName*> since the last synchronization.	The corresponding view will be recreated with the new name.
KNTA-10516	A new configuration with PARAMETER_SET_ CONTEXT_ID = <*fieldSet*> was found for the template.	Since the last synchronization, a new request type, object type, or context-sensitive user data field set was created.	A corresponding view will be generated in the RML.
KNTA-10517	User has requested to drop view <*ViewName*> from the RML.	User has requested to drop the specified view from the RML.	View will be dropped.
KNTA-10518	User requested to drop all RML views generated from template <*TemplateName*>.	User has requested to drop the template from the RML.	All RML views generated from the template will be dropped.

Table B-1. RML synchronization messages, continued

Message ID	Display Text	Cause	Action/Solution
KNTA-10543	Duplicate view name. The RML view name <*ViewName*> you have specified for <*EntityType1*> <*EntityID1*> conflicts with the name of the existing RML view for <*EntityType2*> <*EntityID2*>. Please choose another name for the view.	The same RML view name has been specified for multiple request types, object types, or context-sensitive user data field sets.	Update one of the entities with the duplicate RML view name and change it to a unique name.
KNTA-10545	Cannot change the view name for built-in static context template <*TemplateName*>. Please restore the original view name <*TemplateName2*>.	The static view names defined in RML view templates that come with PPM cannot be changed.	Restore the original view name <*ViewName*>.
KNTA-10546	Driving context changed for template <*TemplateName*>. It is not allowed to change driving context set for Micro Focus PPM built-in template. Please restore original template. Original driving context set is <*ContextID*>.	Some unsupported customization has been done for the name template.	Be sure to undo those changes.
KNTA-10547	Cannot change view name prefix to <*NewViewPrefix*> for built-in template <*TemplateName*>. Please restore original [VIEW_NAME_PREFIX= <*OldViewPrefix*>].	The Token [VIEW_NAME_PREFIX=...] must be defined for specific entity-based RML view templates. The definition is not allowed to change for those view templates that come with PPM.	Restore the view to the original.

Table B-1. RML synchronization messages, continued

Message ID	Display Text	Cause	Action/Solution
KNTA-10548	Driving context changed from <*OldContext*> to <*NewContext*> for template <*TemplateName*>. Cannot continue assessment. To change driving context set for the template, you must drop the template first and then re-assess the template.	For specific entity-based RML view templates, [DRIVING_ PARAMETER_SET = parameter_set_id] is specified.	Drop the template and synchronize again to make the change take effect.
KNTA-10549	Cannot find view <*ViewName*> to drop; no action taken.	The RML view to drop does not exist.	No action was taken during RML synchronization; the view was previously dropped.
KNTA-10550	Drop operation complete.	The operation requested by the user to drop specific RML views has been completed.	No action needed.
KNTA-10551	Since the last RML synchronization there have been some configuration changes that impact views generated from template <*TemplateName*>.	Several types of configuration events since the last RML synchronization could lead to this message: A new request type, object type, or context-sensitive user data field set was created; a field definition was added, modified, or removed in an existing request type, object type, or user data field set; a request type, object type, or context-sensitive user data field set was deleted.	Impacted views will be dropped and recreated, as applicable.

Table B-1. RML synchronization messages, continued

Message ID	Display Text	Cause	Action/Solution
KNTA-10552	Dependent context sets changed for template <*TemplateName*>. Could not continue process. To make this kind of change, the template must be dropped and re-assessed.	All RML templates that have custom data fields in them have associated contexts. Any changes to the dependent contexts invalidate all views and the templates that have previously been assessed or synchronized.	Perform a drop operation on the name template and synchronize the template again.
KNTA-10562	A new RML view will be generated for the template <*TemplateName*>.	The RML view for the specified template did not exist prior to this synchronization.	The view will be created during the RML synchronization.
KNTA-10591	Existing view name is restored: <*ExistingView Name*>.	The operation requested by the user to restore a specific RML view has been completed.	View will be restored.
KNTA-10620	Another RML assessment or synchronization process is already running. There can be only one RML assessment or synchronization process running at the same time.	Another RML assessment or synchronization process is running at the same time this one was submitted.	Try again later.
KNTA-10621	Meta Layer Synchronization Process can only run from the original report type <*reporttype*>. Currently running report type is <*reporttype*>.	Incorrect report type.	Rerun the synchronization process from the correct report type.

Table B-1. RML synchronization messages, continued

Message ID	Display Text	Cause	Action/Solution
KNTA-10622	No Change Detected.	The view or template specified for synchronization was not updated, because it is already up-to-date.	No further action is needed.
KNTA-10629	The RML view name is not defined for <*EntityType*> <*EntityID*>.	All configurations that drive a dynamic RML view (request types, object types, and context-sensitive user data field sets) must have an RML view name specified. This one did not.	Specify an RML view name for the corresponding configuration.
KNTA-10675	Value <*ViewName*> is too long for [VIEW_NAME] in template <*TemplateName*>. Value for [VIEW_NAME] must be no more than 30 characters.	The length of the supplied RML view name was too long.	Shorten the value of the RML view name to less than 30 characters.
KNTA-10678	Value <*ViewName*> is too long for [VIEW_NAME_PREFIX] in template <*TemplateName*>. Value for [VIEW_NAME_PREFIX] must be no more than 10 characters.	The view name prefix cannot be more than 10 characters long.	Change the length of the view name prefix.
KNTA-10680	Could not resolve driving parameter set context id in template <*TemplateName*>. Please verify the template.	The RML system could not resolve the specific entity on which the name template is based.	Contact Software Support to report the problem.

Table B-1. RML synchronization messages, continued

Message ID	Display Text	Cause	Action/Solution
KNTA-10681	The RML view name <*ViewName*> specified for <*EntityType*> <*EntityID*> conflicts with the name of another RML view that is either currently pending to be dropped, or was supposed to have been dropped but failed.	The RML synchronization is attempting to create a new view with the same name as an existing view that should have been dropped.	Make sure the existing view <*ViewName*> has been successfully dropped before trying to create the new one.
KNTA-10698	No views will be generated as no driving contexts were resolved for the template <*TemplateName*>.	The specified template will generate a view for each configuration instance (request type, object type, or user data field set), but no configurations were found for the template.	Verify that the driving contexts were correct. If they were, no further action is needed. Otherwise, provide the revised driving contexts.
KNTA-10714	The Meta Layer view name <*ViewName*> you have specified for <*EntityType*> <*EntityID*> conflicts with the name of another Meta Layer view that was previously synchronized, but was not successfully created. To clean up from the previous failed synchronization, please run a Meta Layer Synchronization report to explicitly drop the view <*ViewName*> before trying to create the new one.	An error occurred during a previous run of the synchronization report.	Drop the view.

Table B-1. RML synchronization messages, continued

Message ID	Display Text	Cause	Action/Solution
KNTA-10715	The Meta Layer view name <*ViewName*> you have specified for <*EntityType*> <*EntityID*> conflicts with the name of another Meta Layer view that was previously synchronized, but was not successfully replaced. To clean up from the previous failed synchronization, please run a Meta Layer Synchronization report to explicitly drop the view <*ViewName*> before trying to create the new one.	A failed synchronization has caused a conflict in view names.	Drop the views causing the problem.
KNTA-10863	The Meta Layer view name <*ViewName*> you have specified for <*EntityType*> <*EntityID*> conflicts with one of the reserved Meta Layer views. Please choose another name for the view.	Conflicting value specified.	Choose a different name for the new Reporting Meta Layer View.
KNTA-10864	The Meta Layer view name <*ViewName*> you have specified for <*EntityType*> <*EntityID*> conflicts with the name of an existing Meta Layer view for Request User Data. Please choose another name for the view.	Conflicting value specified.	Choose a different name for the new Reporting Meta Layer View.
KNTA-10865	Template parse error: Unable to parse multi-value-expandable token. No value provided for multi-value variable: <*variableName*>.	RML parser cannot locate the values for the specified multi-value-variable when it parses a multi-value-expandable token.	Make sure the multi-value-variable is defined in the template and values are properly assigned to the variable.

Table B-1. RML synchronization messages, continued

Message ID	Display Text	Cause	Action/Solution
KNTA-10866	'=' is missing for multi-value-assignment token.	Multi-value-assignment token has the following syntax: [MULTI_VALUE_ASSIGNMENT:VAR={sql query}] where {sql query} can contain additional global assignment tokens.	Fix the syntax of the token.
KNTA-10867	Missing multi-value variable 'VAR' for Multi-value expandable token.	Missing value.	Fix the syntax of the token.
KNTA-10868	Missing divider 'DIVIDER' for Multi-value expandable token.	Missing value.	Fix the syntax of the token.

Send Us Feedback

Let us know how we can improve your experience with the Reporting Meta Layer Guide and Reference.
Send your email to: docteam@microfocus.com