Oracle® Agile Product Lifecycle Management for Process Reporting Guide

Feature Pack 4.1

E64712-01

July 2015

ORACLE®

Copyrights and Trademarks

Agile Product Lifecycle Management for Process

Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface

Audience

This guide is intended for client programmers involved with integrating Oracle Agile Product Lifecycle Management for Process. Information about using Oracle Agile PLM for Process resides in application-specific user guides. Information about administering Oracle Agile PLM for Process resides in the *Oracle Agile Product Lifecycle Management for Process Administrator User Guide*.

Variability of Installations

Descriptions and illustrations of the Agile PLM for Process user interface included in this manual may not match your installation. The user interface of Agile PLM for Process applications and the features included can vary greatly depending on such variables as:

- Which applications your organization has purchased and installed
- Configuration settings that may turn features off or on
- Customization specific to your organization
- Security settings as they apply to the system and your user account

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Software Availability

Oracle Software Delivery Cloud (OSDC) provides the latest copy of the core software. Note the core software does not include all patches and hot fixes. Access OSDC at:

http://edelivery.oracle.com

Chapter 1—PLM for Process Reporting Introduction

Purpose

This guide describes how to configure custom reports for the Oracle Agile Product Lifecycle Management (PLM) for Process Reporting module.

Overview

All reports will use a reporting service like Oracle BI Publisher or SQL Server Reporting Services. However, PLM for Process allows you to execute these reports in different ways. Depending on your usability needs, environment requirements and security requirements around parameters, either choice is valid.

1. **Custom Report Configuration** – This approach allows you to execute the report from within PLM for Process. You can use the PLM for Process custom report UI interface and the report can be returned to the user in PDF format. While this involves more complexity during the configuration of the report, this keeps the user within the PLM for Process application and security filtering can be applied to report parameters. For example, if a specification is required as a parameter, PLM for Process UI tools can be used to filter the user's available choices using core application security rules like GSM BU Security. You would also use this approach when you don't want to set up user access in the reporting application. More information around configuring this approach can be found in the Custom Report Configuration Section of this guide.

2. **BI Publisher Integration** – This approach allows the user to execute reports within a reporting application. We recommend Oracle BI Publisher and offer out of the box reports that can be leveraged. If the user has appropriate roles in BI publisher they can customize the report layouts as well as create new reports based on the provided data models. More information around configuring this approach can be found in the BI Publisher Integration Section of this guide.

As you can see in the example click streams below, PLM for Process offers navigation extensibility that allows both approaches to be contextually called from a specific object. For example, if the user is on Material Specification ABC they can launch a report by clicking on a link within PLM for Process. "Material Specification ABC" can be sent as a report parameter to the reporting method of choice.

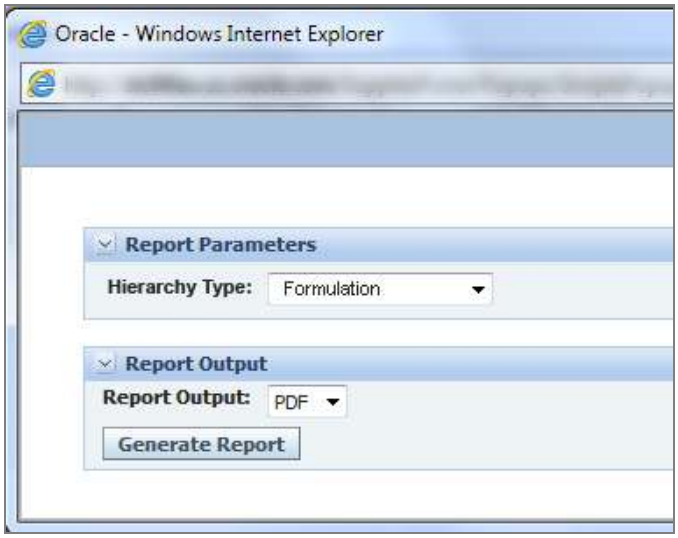
Custom Report Configuration: Example Click Stream

1

Contextually link to the report using an icon or a Text link in the action menu.

The following example shows the "Where Used – Parents" report.

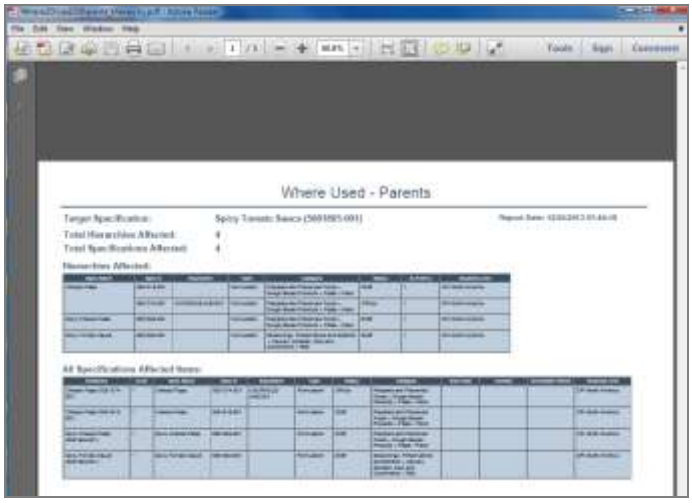
2



PLM for Process Custom Report popup opens. User is able to select parameters using PLM for Process UI tools. Any report parameter can be configured here.

You can even add parameters that allow you to search for objects like specifications. When the user searches for a specification all application security will be respected.

3



Report is returned in PDF format.

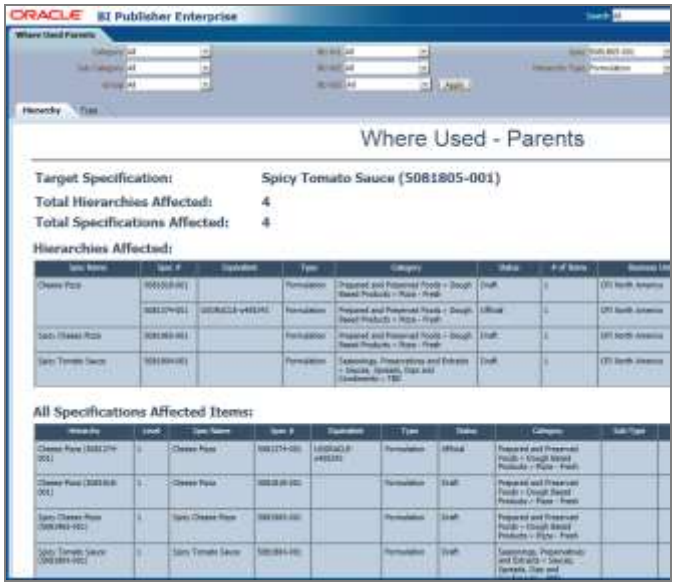
BI Publisher Integration: Example Click Stream

1

Contextually link to the report using an icon or a text link in the action menu.

The following example shows the "Where Used – Parents" report.

2



The BI Publisher application is open in a new window. The user is presented with an interactive report. They are able to select any parameters and see the results instantly. There are also interactive filtering features the user can participate in. If they have admin access they can adjust the reports they have access to by changing layout, column display etc.

This approach can also leverage the Report Dialog application. Learn more in the Report Dialog section below.

Chapter 2—Report Packs

There are “out of the box” BI Publisher reports available as reference implementations, known as Report Packs. This section describes the intent of each released report.

All of the reports are deployed using Oracle BI Publisher and can be accessed using Custom Report Configuration or the BI Publisher Integration method.

The following report packs are available:

ADMN Report Pack

- Access Report
- Access Details
- Basic Statistics

EQ Report Pack

- eQ Dashboard
- eQ Dashboard – Supplier Portal

GSM Report Pack

- Approval / Audit Trail
- Hierarchy Object Status
- Material Cost Matrix
- Spec Lineage History
- Supporting Document Review
- Where Used Parents
- Where Used Parents and Children
- Nutrition Facts Panel Report Pack

SCRM Report Pack

- Supplier Review
- Sourcing Approval Risk
- Supplier Rationalization
- Supplier Compliance Report – Companies & Facilities
- Supplier Compliance Report – Sourcing Approvals
- Supplier Compliance Report – Supplier Portal

PQM Report Pack

- Individual Supplier Quality
- Individual Supplier Quality – Supplier Portal
- Alternate Supplier
- Quality Analyst Dashboard
- Quality Manager Dashboard
- Audits by Spec
- Audits by Suppliers
- Open Issues by SAC
- Open Issues by Spec
- Open Issues by Spec Hierarchy
- Open Issues by Supplier

IMPORTANT: All Report Packs are released as reference implementations. These reports should be copied to another folder before using. It is expected that they will then be customized to meet your specific requirements.

ADMN Report Pack

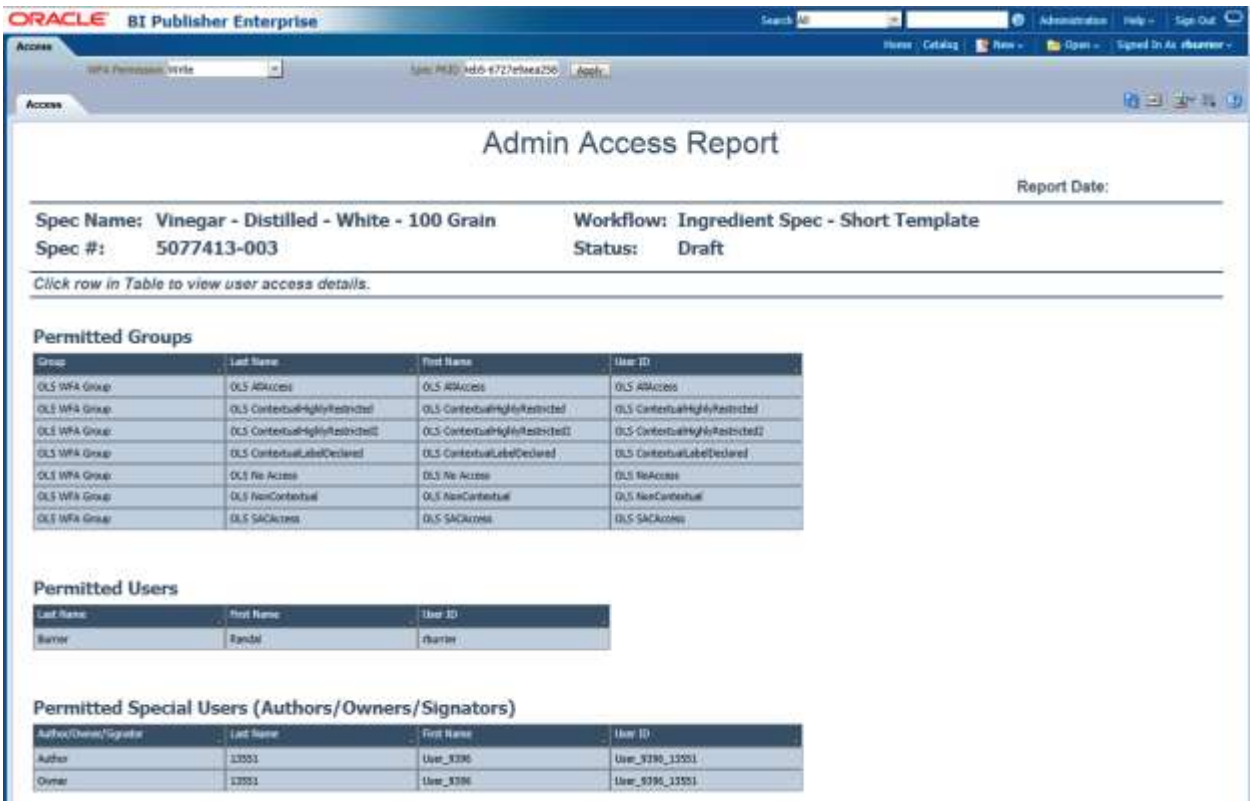
The admin report pack includes data models and reports that would be beneficial to PLM for Process administrators.

Access Report (ADMN)

Who has access to this specification?

The Access Report is intended for specific system admin users responsible for UGM and/or WFA security. This report should be launched from a GSM specification using a navigation extension that is only exposed to the system admin. This report pulls all UGM groups, permitted users, and special permitted users (i.e. Authors, Owners or Signators) that have access to the GSM specification on a specific workflow at a specific status.

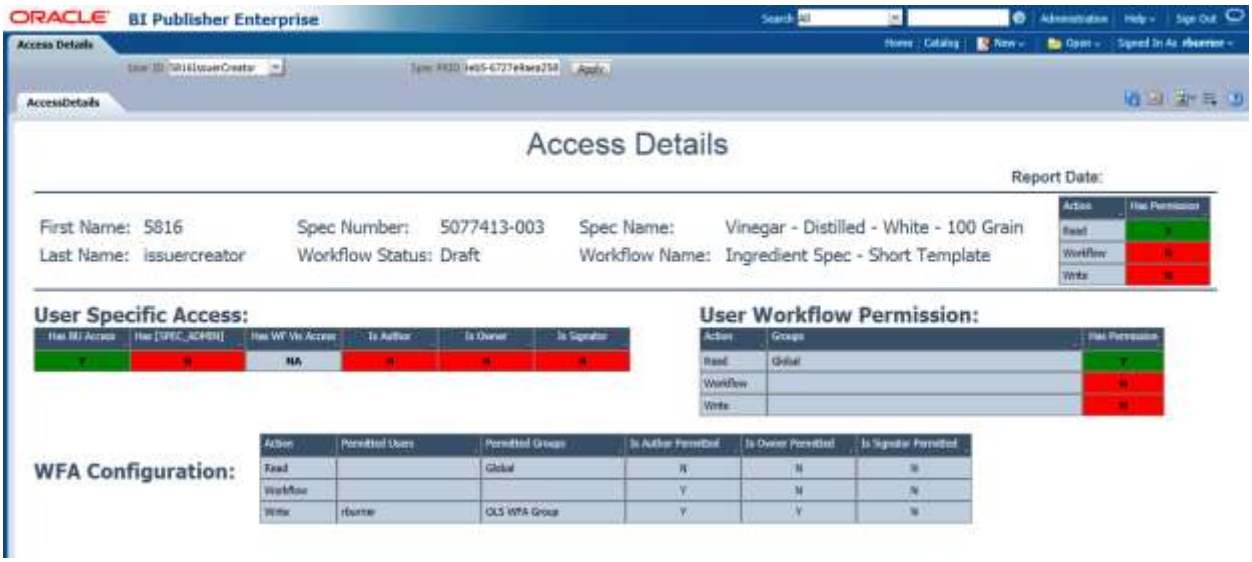
The report has two parameters: WFA Permission and Spec PKID. The specification PKID is passed to the report via navigation extension. WFA Permission has three values: Read, Write, and Workflow. The report contains only one tab (or view) called Access. This report should work with the UGM Access Detail report, as the Access Detail report can be launched from within this report to expose more security detail for a specific user.



Access Details (ADMN)

What kind of access does a specific user have to this specification?

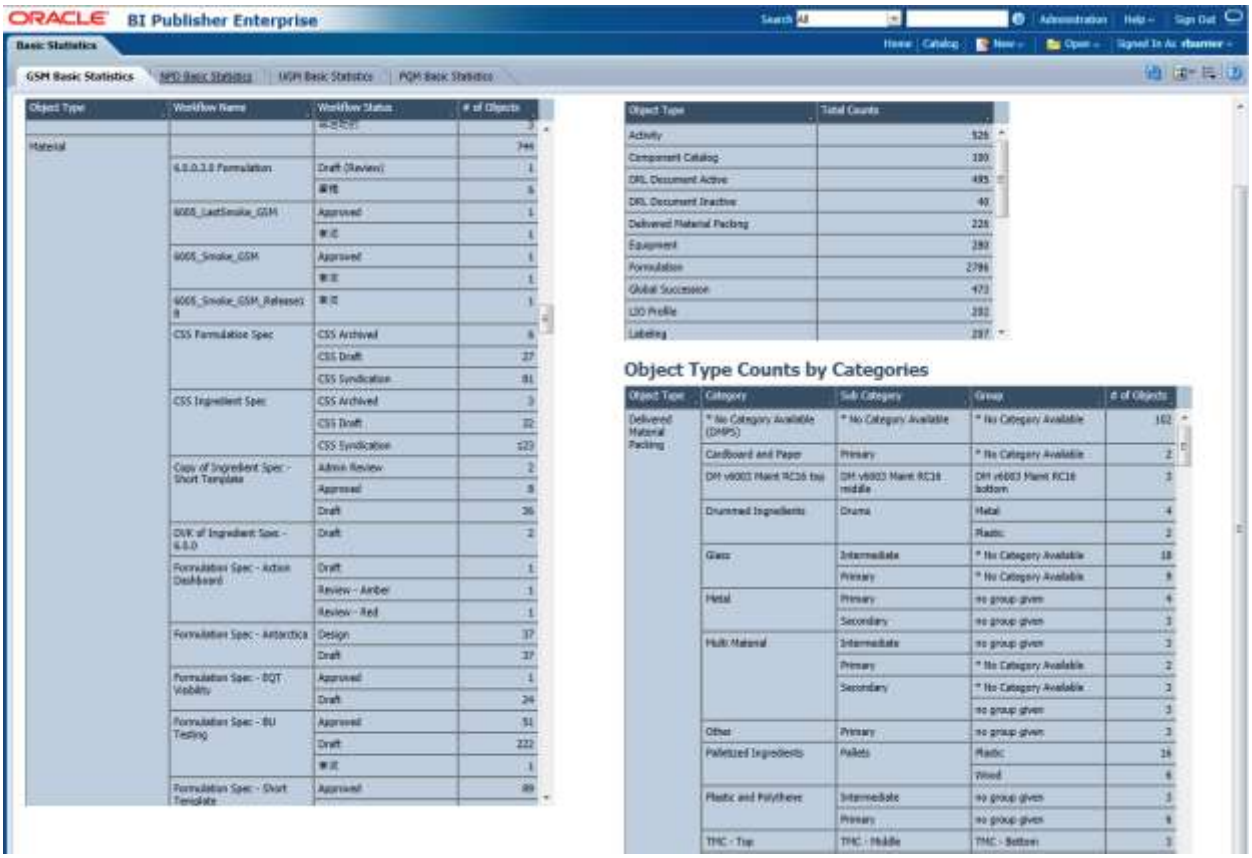
The Access Details report should be launched from a hyperlink within the UGM Access report, providing the access detail for the specified specification and user from which the report was launched. This report pulls user specific security access such as BU access, [SPEC_ADMIN] role assignment, workflow visibility access, workflow permission, special access (Author, Owner and Signator) and summarizes the Read, Workflow, and Write permissions. At a glance, the system admin can determine if a user has Read, Write, or Workflow permission and view details that allows or denies the user access to the specification. The report contains only one tab (or view) called Access Details.



Basic Statistics (ADMN)

How many active users, approved suppliers, GSM, NPD and PQM objects are in the system?

The Basic Statistics report should be launched from ADMN or PLM for Process Portal. The report could be used by system executives that would like to have a high level understanding of the utilization of the PLM for Process application, which will lead to a better understanding of the impact that the PLM for Process application (and investment) has on their business. This report pulls all basic counts of active users, approved suppliers, GSM specifications (by workflow, specification type and Category), NPD objects (Projects, Strategic Briefs and Innovation Sales Pipelines) and PQM objects (Issues, Actions and Audits). The report contains four tabs (or views): GSM Basic Statistics, NPD Basic Statistics, UGM Basic Statistics, and PQM Basic Statistics.

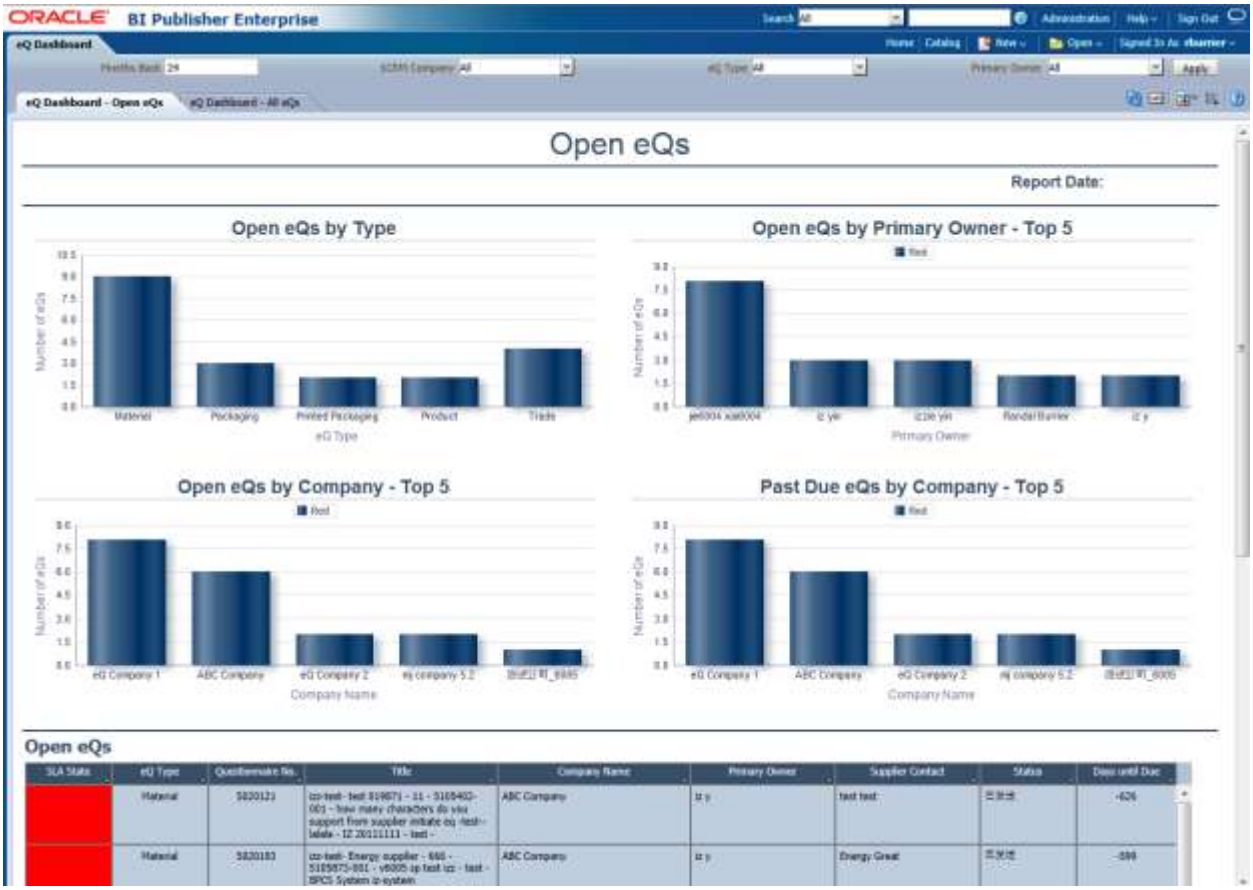


eQ Report Pack

The eQ report pack includes data models and dashboards that will be beneficial to eQ administrators as well as suppliers who have been sent eQuestionnaires.

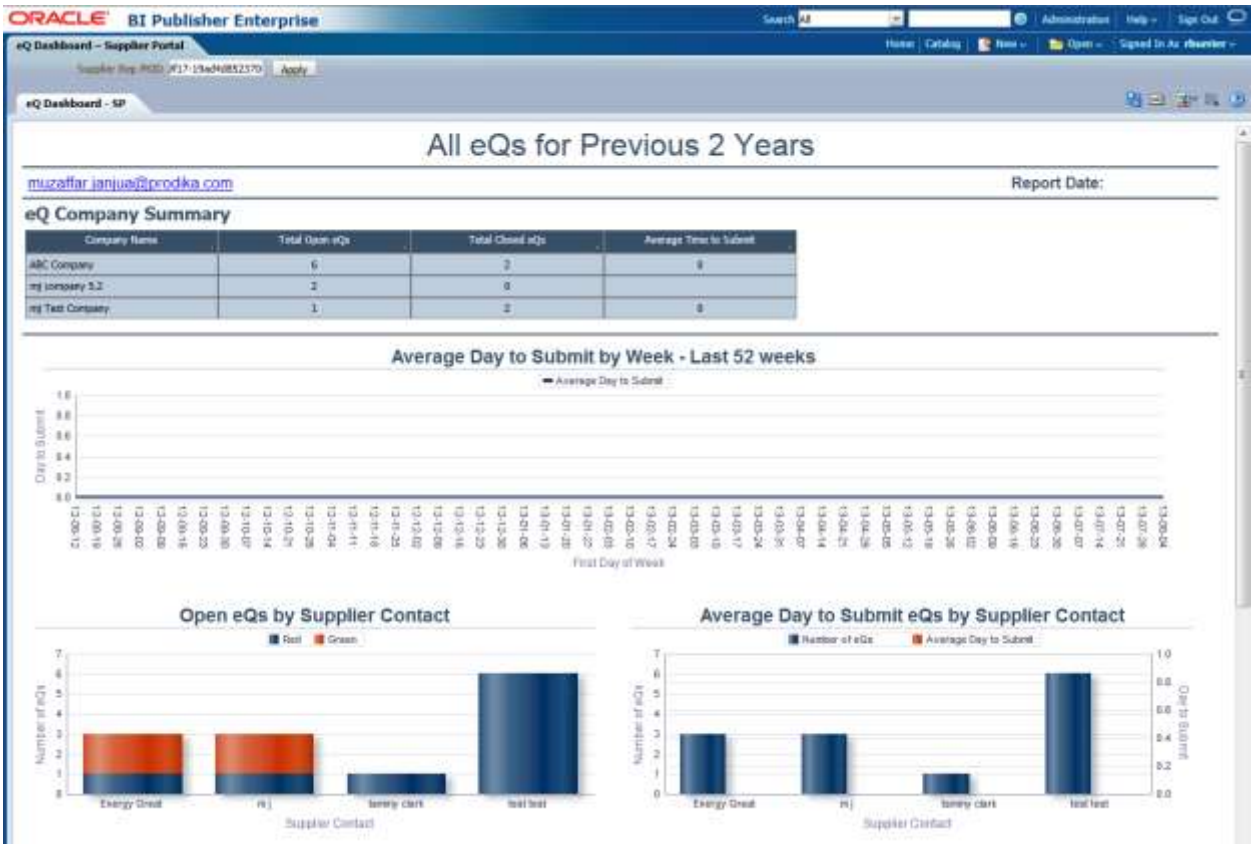
eQ Dashboard (eQ)

The eQ Dashboard should be launched from eQ or the PLM for Process Portal. Any eQ user or administrator can utilize it. This report enables a view into open and completed eQs across eQ owners, suppliers and eQ types. The eQ Dashboard also exposes trending information on supplier response times and total open eQ counts by supplier. This report helps better manage the eQ business process by exposing vital eQ information across eQ users and suppliers. This report has four parameters: Months Back, SCRM Company, eQ Type and Primary Owner. The report contains 2 tabs (or views): Open eQs and All eQs.



eQ Dashboard – Supplier Portal (eQ)

The Supplier Portal eQ Dashboard should be launched from Supplier Portal and utilized by a supplier eQ user. This report enables a supplier to view open and completed eQs across supplier contacts and also exposes trending information on supplier response times and total open eQ counts. This report helps the supplier better manage the eQ business process by exposing vital eQ information. This report has one parameter called Supplier Rep PKID, which should be hidden prior to releasing this report. As with all Supplier Portal reports, this report should be released in pdf format and use the custom report configuration method.



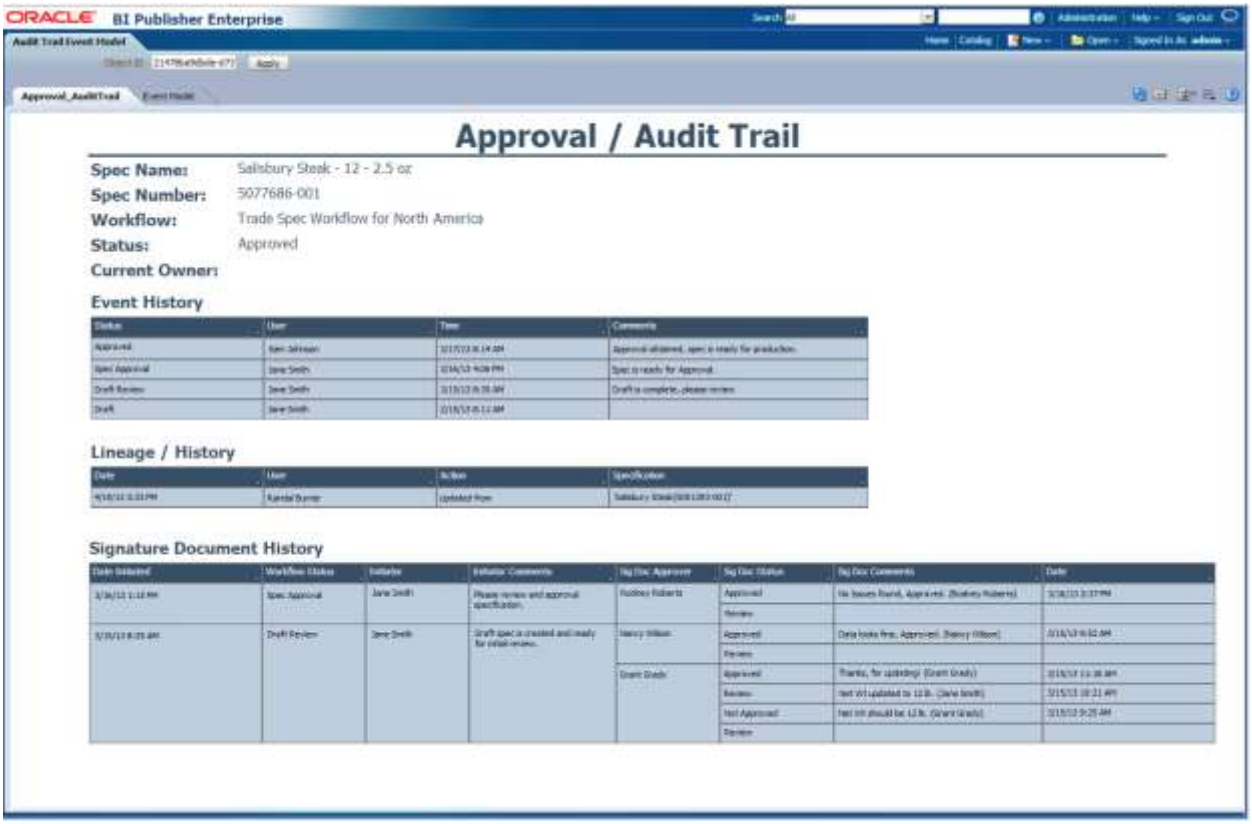
GSM Report Pack

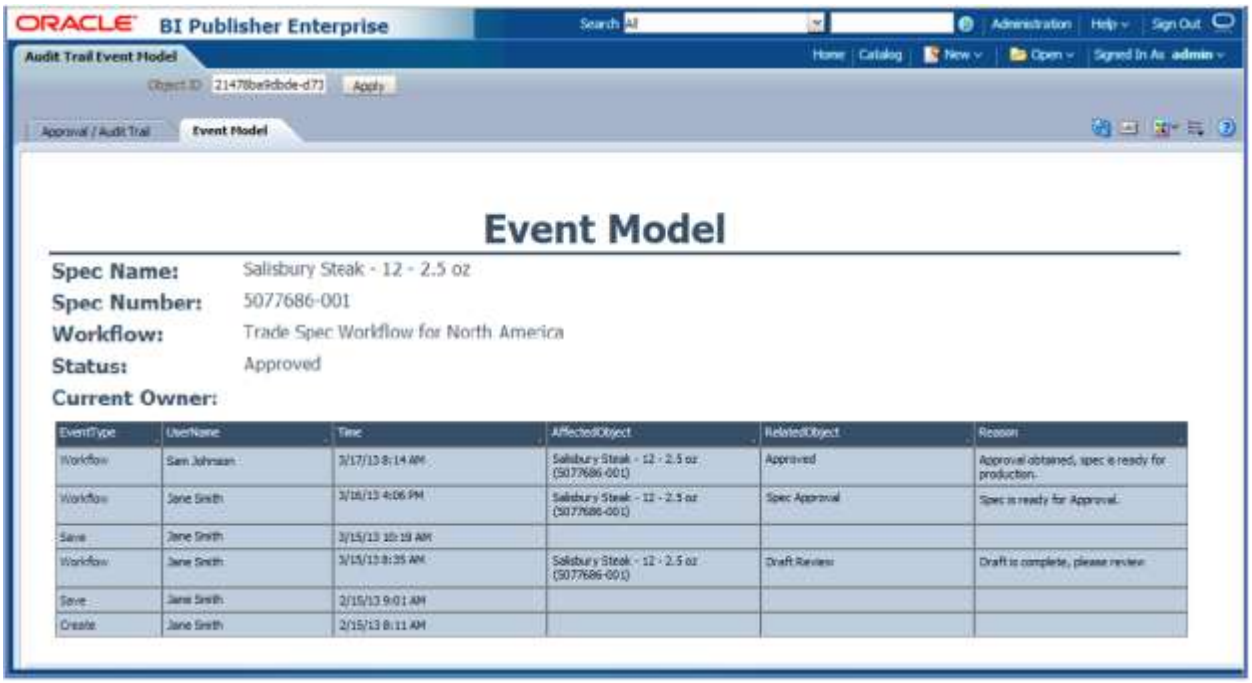
The GSM report pack includes data models and reports that would be beneficial to GSM users in all sorts of functional areas.

Approval / Audit Trail (GSM)

What is the historical overview of a specification, including workflow, signature documents, cross-reference associations and event table entries?

This report was designed to be launched from the Hierarchy Object Status Report, but it can be configured to launch from a GSM specification or the Hierarchy Navigator. The report exposes the Event History, Lineage/History, Signature Document History, and the Event Model tables in one consolidated report, which allows the user to quickly review the entire history of the specification. This report contains one parameter, Object (Spec) PKID. The specification PKID is passed to the report via navigation extension. This report contains two tabs (or views) called Approval / Audit Trail and Event Model.



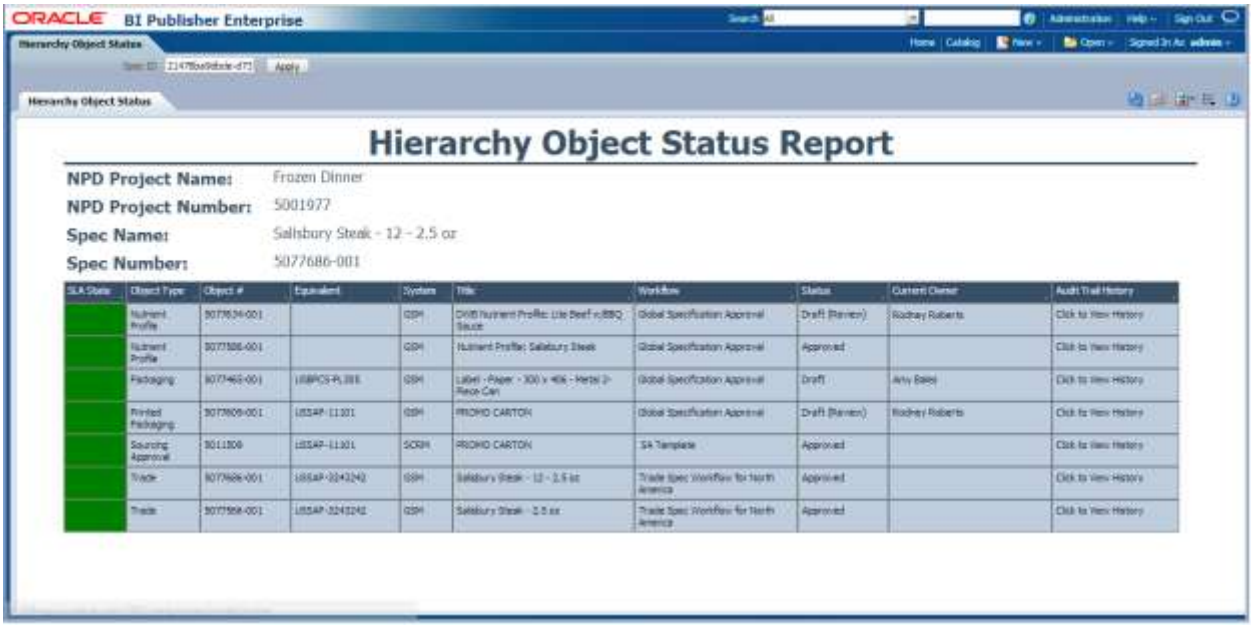


Note: In order for the Event Model to populate, the Common.Auditing.LifecycleEvents.Enabled configuration must be set to “true”.

Hierarchy Object Status (GSM)

What are the statuses of all the objects in the current hierarchy?

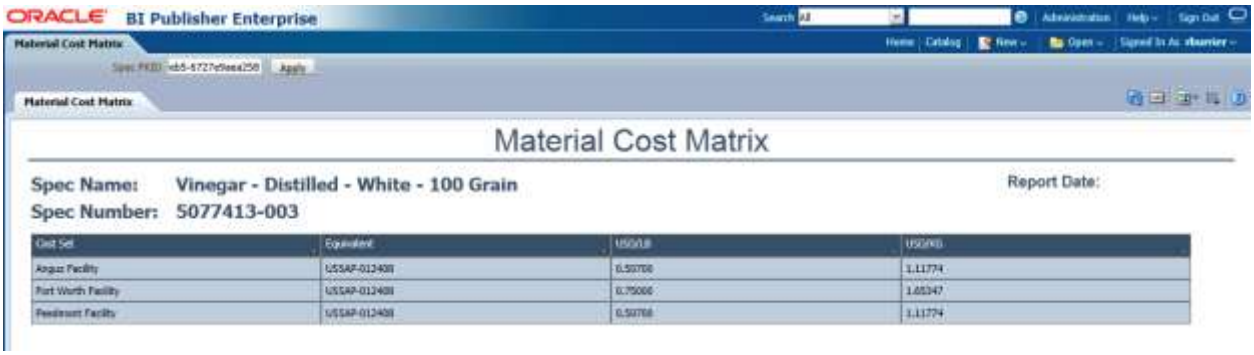
The Hierarchy Object Status Report is designed to launch from the Hierarchy Navigator to reveal a workflow status review of all specification types, GSM activities, and specification-related sourcing approvals associated to the specification's hierarchy. The report also exposes the related NPD projects and project numbers. The report contains an embedded link to the Approval / Audit Trail report, which gives the user visibility to an historical overview of the entire hierarchy in just a few clicks. This report contains one parameter, Spec PKID. The specification PKID is passed to the report via navigation extension. This report contains one tab (or view) called Hierarchy Object Status.



Material Cost Matrix (GSM)

How much does this material or packaging specification cost per facility?

This report was designed to be launched from a material or packaging specification. The report returns cost book information associated to the GSM object. If the cost book is populated, this report can be utilized by formulators to quickly view the cost information for a specific GSM objects across all Cost Sets (or Facilities). This report contains one parameter, Spec PKID. The specification PKID is passed to the report via navigation extension. This report contains one tab (or view) called Material Cost Matrix.



Spec Lineage History (GSM)

What specifications have been associated to the same cross-reference?

The Spec Lineage History report was designed to be launched from a specification. This report can be utilized by GSM users who would like to be able to view a specification's complete lineage history based on the specification's cross reference information. For example, in change management scenarios new system specification numbers are created when a material change occurs. This report allows you to see the specification's lineage based on the cross reference instead of just the specification number. The report uses the Spec PKID as a parameter. This report contains one tab (or view) called Spec Lineage History.

Supporting Document Review (GSM)

Show me all supporting documents related to this hierarchy or specification.

This report should be launched from a GSM specification or a specification listed in the hierarchy navigator panel. It provides the user easy access to the supporting documents for the entire GSM hierarchy, including supplier documents. The report has two parameters: Spec ID and Level. The Level parameter decides whether it pulls documents from the entire hierarchy or the single specification level, while the Spec parameter is used as the top level specification in the hierarchy. The report contains one tab.

This report respects all application security including object level security.

IMPORTANT: Hierarchy Denormalization Service is required for this report. You can learn more about this service in the *Agile Product Lifecycle Management for Process Hierarchy Denormalization Guide.*

Report Setup

The Attachment Web Services are required to open attachments directly from this report. To use this report, first set up the attachment web service as described in the *Agile Product Lifecycle Management for Process Web Services Guide,* and then configure BI Publisher to call the service.

To configure BI Publisher to use the attachment service:

1. Download the GetAttachment.html file from

   <EP_Pack_Root>\ReferenceImplementations\ReferenceReportPacks\GSM\

2. Copy the file to the following folder in BI Publisher environment.

3. Create the custom_pages folder if it does not exist.

   <BI Publisher Main Directory>\bip\server\wls\user_projects\applications\bipdomain\xmlpserver\custom_pages\

   Here, <BI Publisher Main Directory> is the root directory for the BI Publisher installation.

Where Used - Parents (GSM)

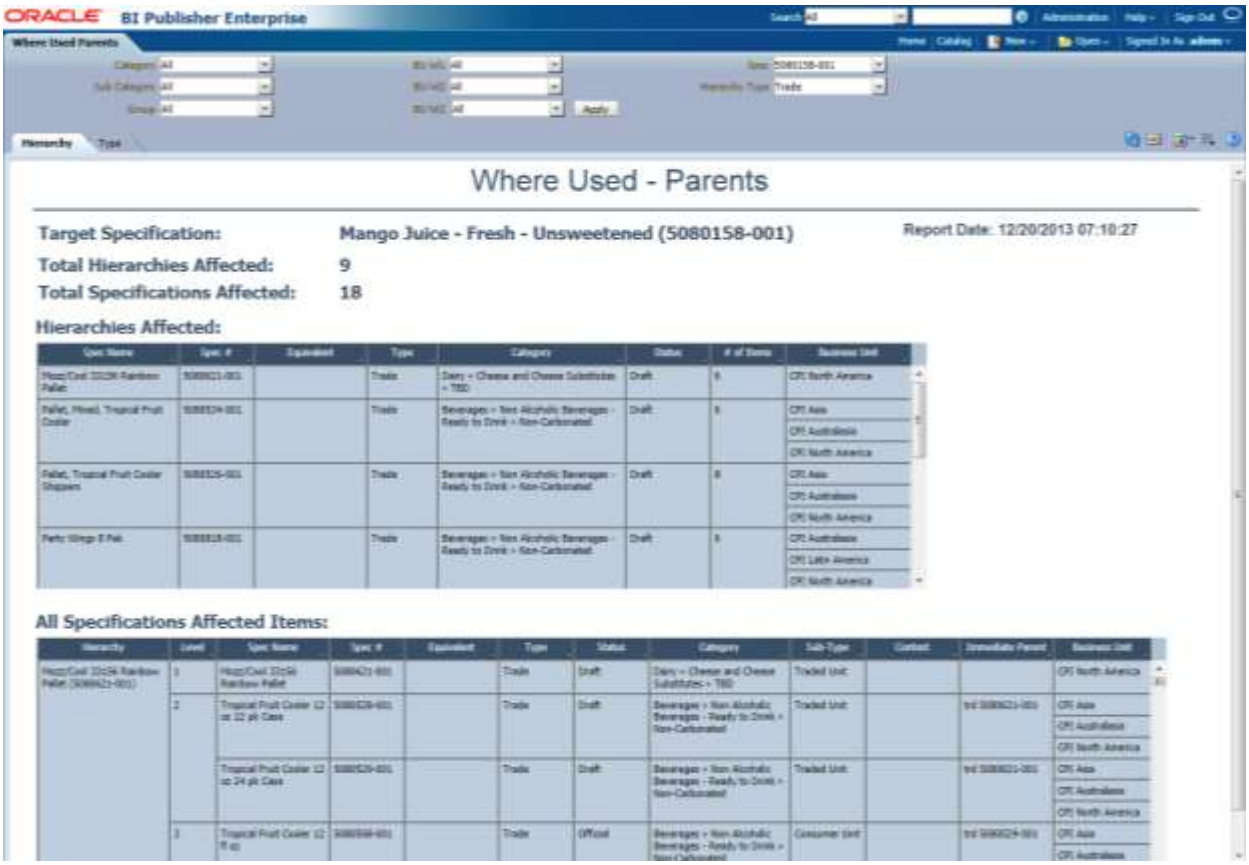
Show me every parent where a specific specification is used.

This report was designed to be launched from any GSM specification. The report returns all parent specifications where a specific specification is used. The report has the following parameters: Category, Sub Category, Group, GSM BU lvl 1, GSM BU lvl 2, GSM BU lvl 3, Spec ID and Hierarchy Type. There are two tabs, Hierarchy and Type. The Hierarchy tab is used to display the hierarchy of specifications that consumes the driving specification for this report. The Type tab is used to show similar data but grouped by type, without displaying the hierarchy.

The Hierarchy tab has two tables, an upper and a lower one. The upper one displays the top level specifications where the main specification is being used and only of the type selected in the Hierarchy Type parameter. For example, if you want to see all top level trade specifications that consume the main specification, select Trade. If you are only concerned about formulations, select Formulation. The lower table shows all the hierarchies between the main specification and each row in the top table.

The Type tab also has two tables. The upper table shows a count of all the specifications where the main specification is used, grouped by type. For example, a user can see in the top table how many trade specifications consume the main specification, by setting the Hierarchy Type to Trade.
The lower table shows all the specification contained within each row of the upper table, grouped all together by type.

This report respects all application security. If you want to improve performance and read security is not important, the Where Used – Parents [No Restrictions] report is available. Where Used – Parents – No Restrictions is the same report but does not check for specification read access.



IMPORTANT: Hierarchy Denormalization Service is required for this report. You can learn more about this service in the *Agile Product Lifecycle Management for Process Hierarchy Denormalization Guide*.

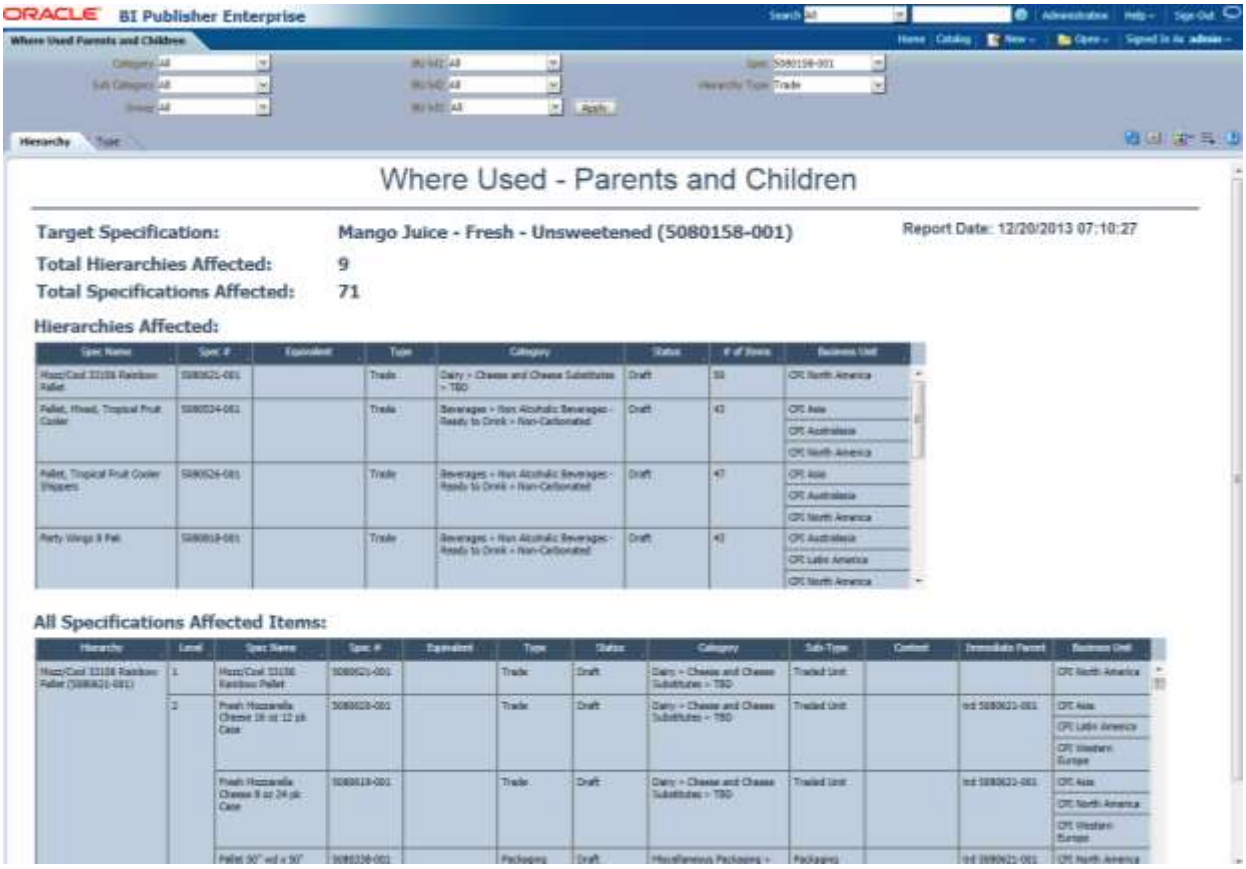
Where Used - Parents and Children (GSM)

Show me every parent and child of hierarchies where a specific specification is used.

This report was designed to be launched from any GSM specification. The report returns all parent and child specifications where a specific specification is used. The report has the following parameters: Category, Sub Category, Group, GSM BU lvl 1, GSM BU lvl 2, GSM BU lvl 3, Spec and Hierarchy Type. Hierarchy Type decides what type of hierarchies is returned. For example, if you want to see all trade specification hierarchies affected select trade. If you are only concerned about formulations select formulation.

The Where Used and the Where Used - Parents and Children reports will return the same data in the upper table, 'Hierarchies Affected'. The difference is that in this report the lower table 'All Specifications Affected Items' will include children as well as the parents.

Due to performance concerns this report does not respect application security. Users will see all specifications listed even if they don't have read access to the specification. However, if the users were to try to open the specification, PLM for Process security will still prevent them from reading the specification. This report is named "Where Used - Parents and Children No Restrictions".



IMPORTANT: Hierarchy Denormalization Service is required for this report. You can learn more about this service in the *Agile Product Lifecycle Management for Process Hierarchy Denormalization Guide.*

Nutrition Facts Panel Report Pack

The Nutrition Facts Panel Report Pack spans across formulation specifications, nutrient profiles and trade specifications. This reporting solution utilizes calculated extended attributes embedded in custom sections to create nutrition facts panel data as defined by Canadian Food Inspection Agency and the United States Food and Drug Administration.

Each report can be executed in English, Spanish or French, and each bilingual report will display the Nutritional Facts Panel in any two of the aforementioned languages. The reports include the following Nutrition Facts Panel formats:

- Standard Format
- Bilingual Standard Format
- Dual Format
- Bilingual Dual Format
- Aggregate Format (up to five products)
- Bilingual Aggregate Format (up to five products)

Where applicable, reports also include Allergens, Label Claims, Complies With, and Ingredient Statement information.

US Standard Report

US - Nutrition Facts Panel Report

Spec Name: Poppers Oatmeal Bites

Spec #: 5081781-001

Equivalent: USORACLE-558899

Nutrition Facts

Allergens

Egg, Milk, Wheat

Complies With

Organic

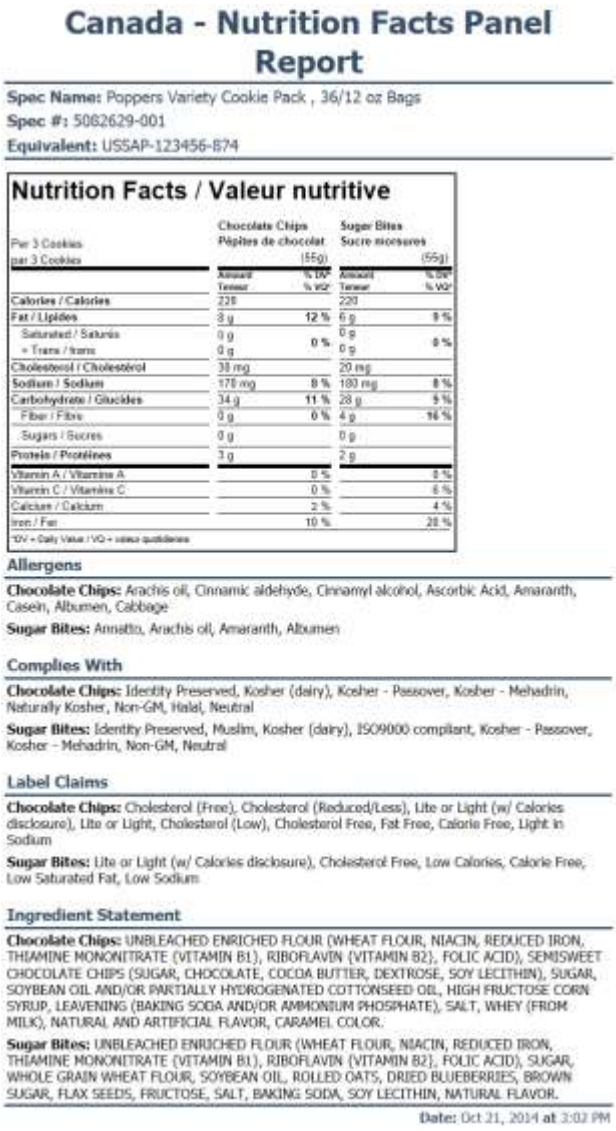
Label Claims

Reduced Sugar, Reduced Fat

Ingredient Statement

Organic Oats, Organic Brown Sugar, Organic Butter (Organic Cream [Milk], Salt), Organic Whole Wheat Flour, Organic Cane Sugar, Organic Whole Eggs, Sea Salt, Baking Powder (Sodium Acid Pyrophosphate, Sodium Bicarbonate, Corn Starch, Monocalcium Phosphate), Organic Vanilla Extract, Baking Soda. Contains Milk, Soy, Wheat, Eggs. Made On A Line Shared With Peanuts, Tree Nuts.

Canada Bilingual Aggregate Report



Canada - Nutrition Facts Panel Report

Spec Name: Poppers Variety Cookie Pack , 36/12 oz Bags

Spec #: 5082629-001

Equivalent: USSAP-123456-874

Nutrition Facts / Valeur nutritive

Per 3 Cookies / par 3 Cookies	Chocolate Chips / Pépites de chocolat (55g) Amount / Teneur	% DV / % VQ*	Sugar Bites / Sucre mesures (55g) Amount / Teneur	% DV / % VQ*
Calories / Calories	220		220	
Fat / Lipides	8 g	12 %	6 g	9 %
Saturated / Saturés + Trans / trans	0 g 0 g	0 %	0 g 0 g	0 %
Cholesterol / Cholestérol	30 mg		20 mg	
Sodium / Sodium	170 mg	8 %	180 mg	8 %
Carbohydrate / Glucides	34 g	11 %	28 g	9 %
Fiber / Fibre	0 g	0 %	4 g	16 %
Sugars / Sucres	0 g		0 g	
Protein / Protéines	3 g		2 g	
Vitamin A / Vitamine A		0 %		0 %
Vitamin C / Vitamine C		0 %		6 %
Calcium / Calcium		2 %		4 %
Iron / Fer		10 %		28 %

*DV = Daily Value / VQ = valeur quotidienne

Allergens

Chocolate Chips: Arachis oil, Cinnamic aldehyde, Cinnamyl alcohol, Ascorbic Acid, Amaranth, Casein, Albumen, Cabbage

Sugar Bites: Annatto, Arachis oil, Amaranth, Albumen

Complies With

Chocolate Chips: Identity Preserved, Kosher (dairy), Kosher - Passover, Kosher - Mehadrin, Naturally Kosher, Non-GM, Halal, Neutral

Sugar Bites: Identity Preserved, Muslim, Kosher (dairy), ISO9000 compliant, Kosher - Passover, Kosher - Mehadrin, Non-GM, Neutral

Label Claims

Chocolate Chips: Cholesterol (Free), Cholesterol (Reduced/Less), Lite or Light (w/ Calories disclosure), Lite or Light, Cholesterol (Low), Cholesterol Free, Fat Free, Calorie Free, Light in Sodium

Sugar Bites: Lite or Light (w/ Calories disclosure), Cholesterol Free, Low Calories, Calorie Free, Low Saturated Fat, Low Sodium

Ingredient Statement

Chocolate Chips: UNBLEACHED ENRICHED FLOUR (WHEAT FLOUR, NIACIN, REDUCED IRON, THIAMINE MONONITRATE (VITAMIN B1), RIBOFLAVIN (VITAMIN B2), FOLIC ACID), SEMISWEET CHOCOLATE CHIPS (SUGAR, CHOCOLATE, COCOA BUTTER, DEXTROSE, SOY LECITHIN), SUGAR, SOYBEAN OIL AND/OR PARTIALLY HYDROGENATED COTTONSEED OIL, HIGH FRUCTOSE CORN SYRUP, LEAVENING (BAKING SODA AND/OR AMMONIUM PHOSPHATE), SALT, WHEY (FROM MILK), NATURAL AND ARTIFICIAL FLAVOR, CARAMEL COLOR.

Sugar Bites: UNBLEACHED ENRICHED FLOUR (WHEAT FLOUR, NIACIN, REDUCED IRON, THIAMINE MONONITRATE (VITAMIN B1), RIBOFLAVIN (VITAMIN B2), FOLIC ACID), SUGAR, WHOLE GRAIN WHEAT FLOUR, SOYBEAN OIL, ROLLED OATS, DRIED BLUEBERRIES, BROWN SUGAR, FLAX SEEDS, FRUCTOSE, SALT, BAKING SODA, SOY LECITHIN, NATURAL FLAVOR.

Date: Oct 21, 2014 **at** 3:02 PM

For additional information regarding the Nutrition Facts Panel reports, please refer to *Oracles Agile Product Lifecycle Management for Process Fact Panel Reporting Guide*.

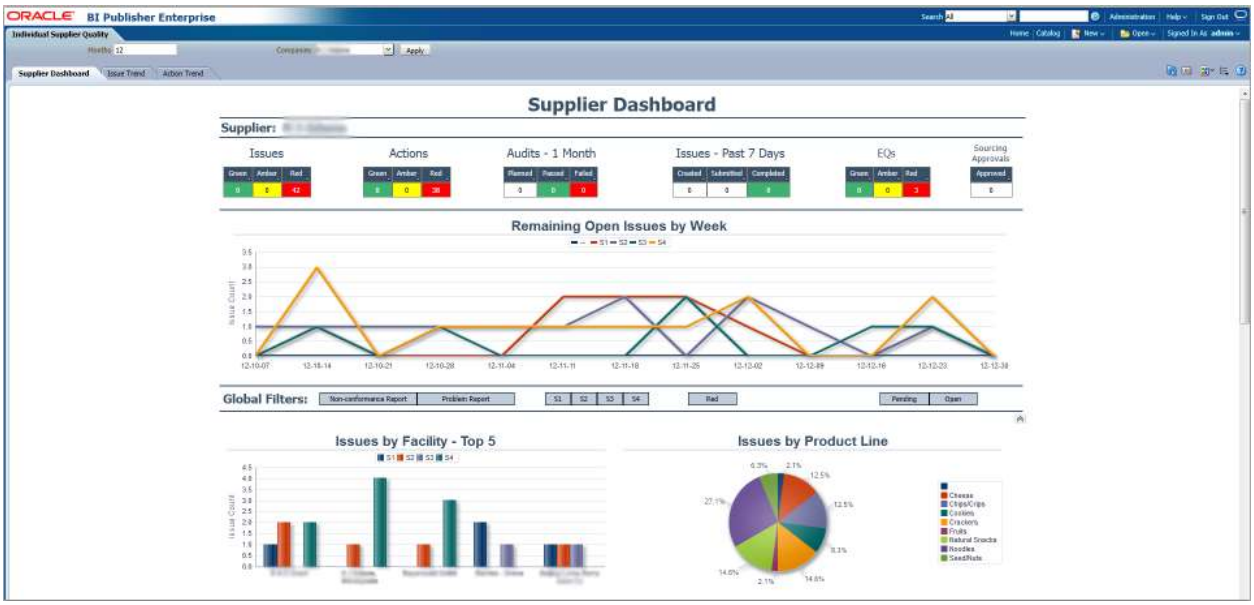
PQM Report Pack

The PQM report pack includes data models, dashboards and reports that would be beneficial to not only quality managers but supplier and specification managers as well.

Individual Supplier Quality (PQM)

The Individual Supplier Quality dashboard focuses on PQM Issues, Actions and Audits for a specific Supplier (Company). This report should be used by personnel responsible for ensuring suppliers are addressing quality related issues in timely manner and/or for helping better understand risks associated with a supplier. The report contains two parameters: Months and Companies. The report contains three tabs (or views): Supplier Dashboard, Issues Trend, and Action Trend.

Note: The Alternate Supplier report can be launched from the Individual Supplier Quality report from a defined column within tables on both the Issues Trend and Actions Trend tab.



Alternate Suppliers (PQM)

What other suppliers can provide this same material?

The Alternate Suppliers report focuses on aiding users to locate an alternate supplier for a GSM specification. The Alternate Supplier report was designed to be launched from within the *Individual Supplier Quality* report from a defined column within tables on both the Issues Trend and Actions Trend tab. The report has one parameter that is used to pass the specification PKID from the Individual Supplier report to pull the appropriate report information. This report contains one tab (or view) called Alternate Suppliers.

Note: The Individual Supplier Quality report can be launched from defined columns within tables on Alternate Suppliers. This feature allows user to perform a side-by-side comparison of two different suppliers via the Individual Supplier Quality report.

Individual Supplier Quality – Supplier Portal (PQM)

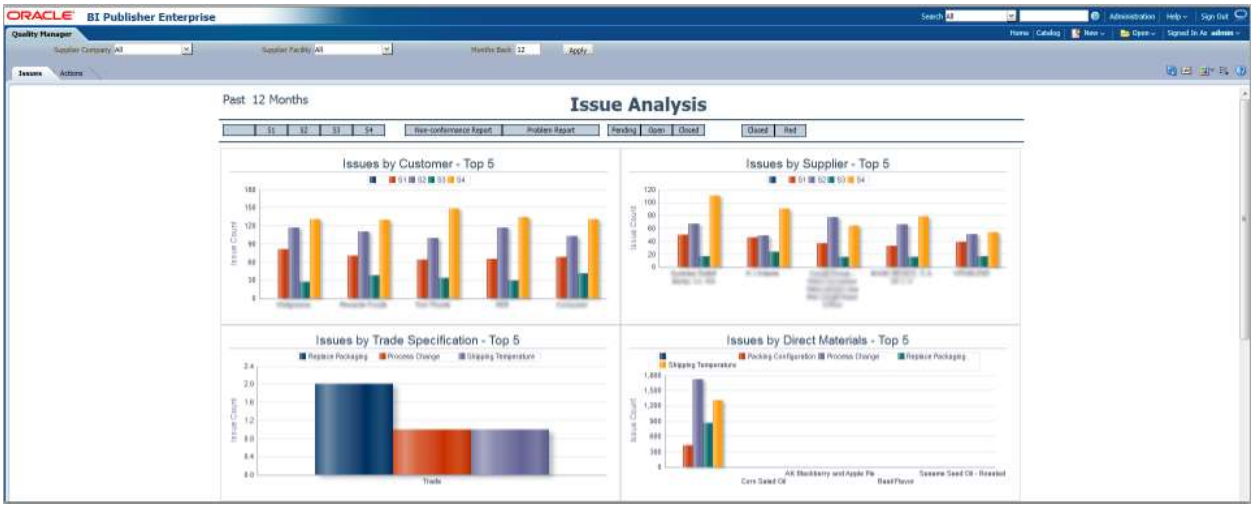
The Individual Supplier Quality – Supplier Portal dashboard focuses on PQM Issues, Actions and Audits for a specific Supplier (SCRM Company) that is accessible to the supplier through Supplier PQM. This report should be used by the supplier to self evaluate their quality performance. The report only returns issues that are related to the SCRM company to which the user is associated. The report has two parameters, Supplier and Months. The report also contains one tab (or view) called Supplier Report.

Quality Analyst Dashboard (PQM)

The Quality Analyst dashboard focuses on open PQM Issues, Actions and Audits and should be used by personnel responsible for ensuring these PQM objects are being addressed by the appropriate personnel in a timely manner. The report has two parameters, Company and Facility. The report also contains three tabs (or views): Issues Dashboard, Action Dashboard, and Audit Dashboard.

Quality Manager Dashboard (PQM)

The Quality Manager dashboard focuses on all PQM Issues, Actions and Audits and should be used by personnel responsible for reviewing and analyzing PQM data to identify possible areas in the quality landscape that need improvement or attention. The report has three parameters, Company, Facility and Months Back. The report also contains two tabs (or views): Issues and Actions.



Audits by Spec (PQM)

Show me all PQM Audits for this specification (Planned or Executed).

The Audits by Spec report focuses on all PQM Audits related to a specific GSM specification. This report should be launched from a GSM specification and grants the user easy access to the quality audit results for the GSM specification. The report has one parameter specification PKID. The report contains one tab (or view), All Audits.

Audits by Suppliers (PQM)

Show me all PQM Audits for this supplier (Planned or Executed).

The Audits by Supplier report focuses on all audits related to a specific SCRM company or facility. This report should be launched from a SCRM company or facility and grants the user easy access to the quality audit results for the SCRM company or facility. The report has two parameters Company PKID and Facility PKID. The report contains one tab (or view), All Audits.

.

Open Issues by SAC (PQM)

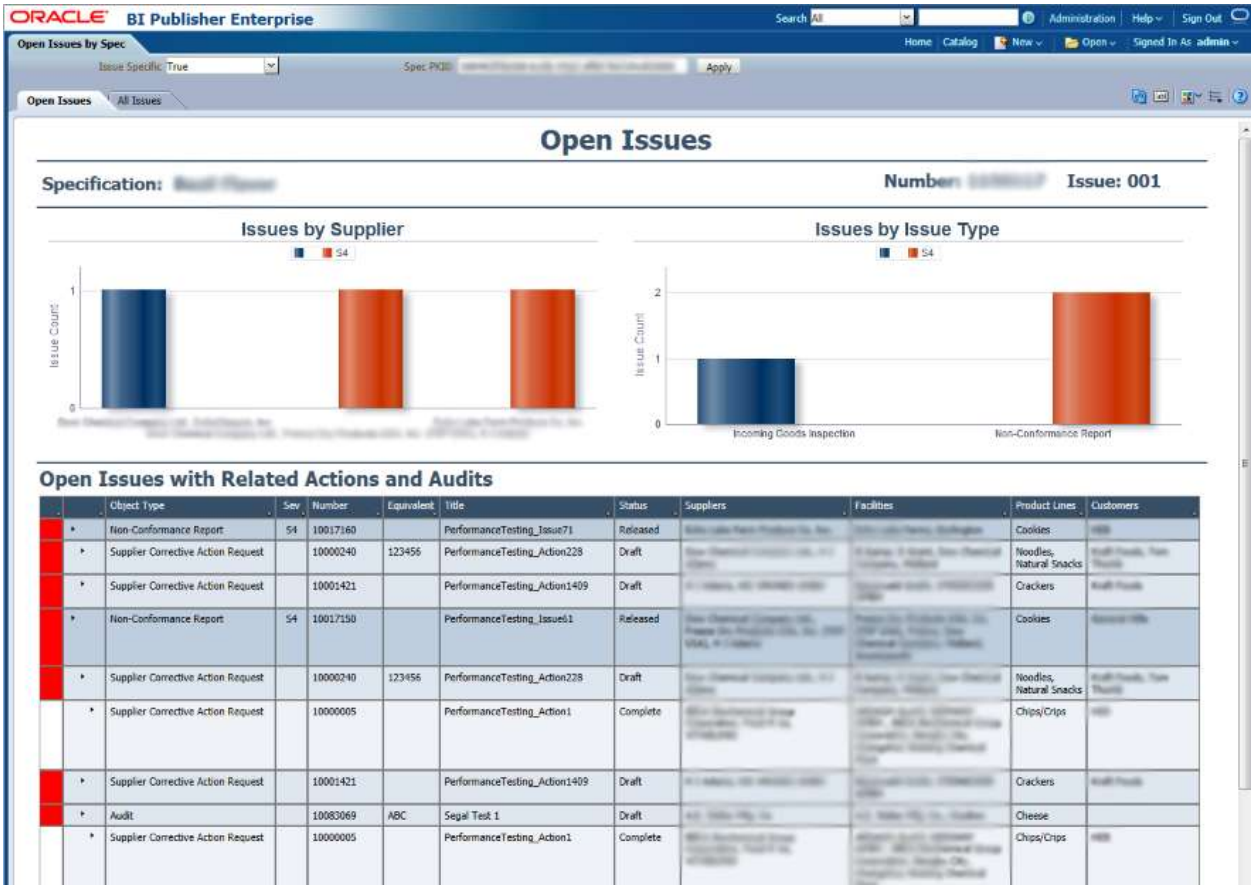
Show me all PQM Issues for this sourcing approval relationship.

The Open Issues by SAC report focuses on open PQM issues related to a specific SCRM sourcing approval relationship. This report should be launched from an SCRM sourcing approval and grants the user easy access to the open quality issues for the GSM specification associated to the sourcing approval. The report has three parameters: Issue Specific, Facility ID and Spec PKID. The Issue Specific parameter can be use to make the report specification "issue" specific or not. The report contains two tabs (or views); Open Issues and All Issues.

Open Issues by Spec (PQM)

Show me all PQM Issues for this specification.

The Open Issues by Spec report focuses on all PQM issues related to a specific GSM specification. This report should be launched from a GSM specification to grant the user easy access to the quality issues for that GSM specification. The report has two parameters: Issue Specific and Spec PKID. The Issue Specific parameter can be use to make the report specification "issue" specific or not. The report contains two tabs (or views): Open Issues and All Issues.



Object Type	Sev	Number	Equivalent	Title	Status	Suppliers	Facilities	Product Lines	Customers
Non-Conformance Report	S4	10017160		PerformanceTesting_Issue71	Released			Cookies	
Supplier Corrective Action Request		10000240	123456	PerformanceTesting_Action228	Draft			Noodles, Natural Snacks	
Supplier Corrective Action Request		10001421		PerformanceTesting_Action1409	Draft			Crackers	
Non-Conformance Report	S4	10017150		PerformanceTesting_Issue61	Released			Cookies	
Supplier Corrective Action Request		10000240	123456	PerformanceTesting_Action228	Draft			Noodles, Natural Snacks	
Supplier Corrective Action Request		10000005		PerformanceTesting_Action1	Complete			Chips/Crips	
Supplier Corrective Action Request		10001421		PerformanceTesting_Action1409	Draft			Crackers	
Audit		10083069	ABC	Segal Test 1	Draft			Cheese	
Supplier Corrective Action Request		10000005		PerformanceTesting_Action1	Complete			Chips/Crips	

Open Issues by Spec Hierarchy (PQM)

Show me all PQM Issues for this entire specification hierarchy.

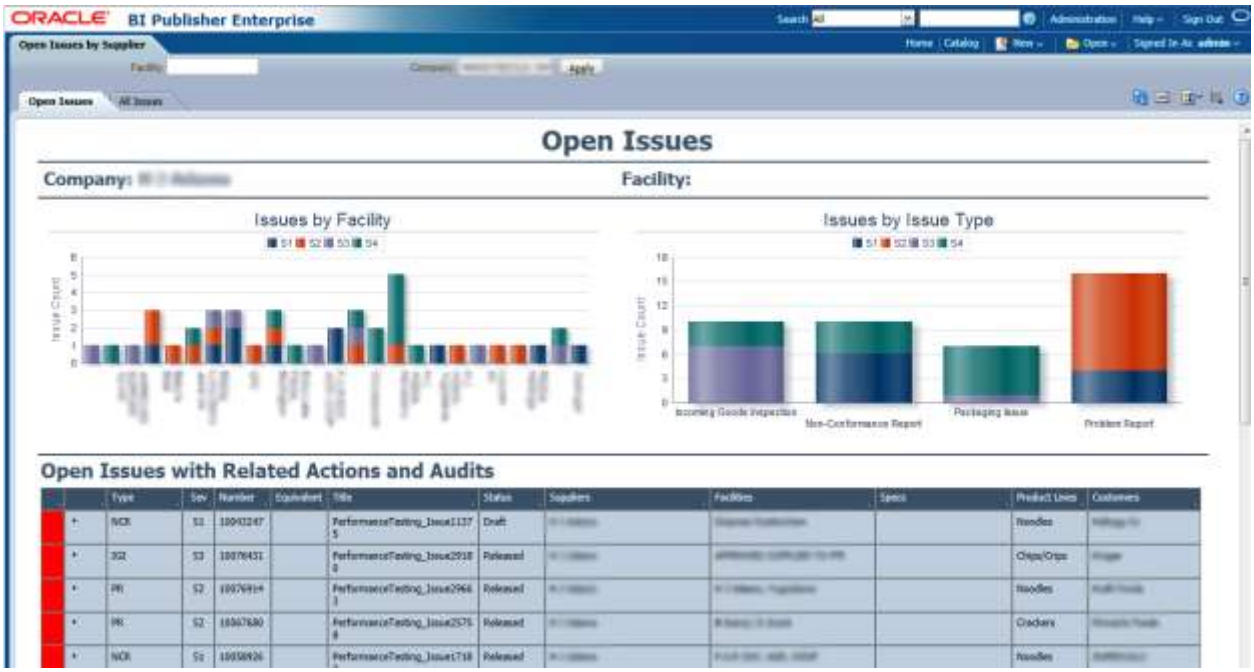
The Open Issues by Spec Hierarchy report focuses on all PQM issues related to a specific GSM hierarchy. This report should be launched from a GSM specification or a specification listed in the hierarchy navigator panel. It grants the user easy access to the quality issues for the entire GSM hierarchy. The report has the Spec PKID parameter, which is used to pass the PKID of the top level specification in the hierarchy. The report contains two tabs (or views): Open Issues and All Issues. In the example below you can see that the "Tropical Fruit Cooler" finished goods have quality issues with raw materials as well as packaging.

IMPORTANT: Hierarchy Denormalization Service is required for this report. You can learn more about this service in the *Agile Product Lifecycle Management for Process Hierarchy Denormalization Guide*.

Open Issues by Supplier (PQM)

Show me all PQM Issues for this Company or Facility.

The Open Issues by Supplier report focuses on all PQM issues related to a specific SCRM company or facility. This report should be launched from a SCRM company or facility and grants the user easy access to the quality issues for that SCRM company or facility. The report has two parameters: Facility and Company. The report contains two tabs (or views): Open Issues and All Issues.



SCRM Report Pack

The SCRM report pack includes data models and reports that are beneficial to sourcing managers.

Supplier Review (SCRM)

What suppliers are involved in this hierarchy?

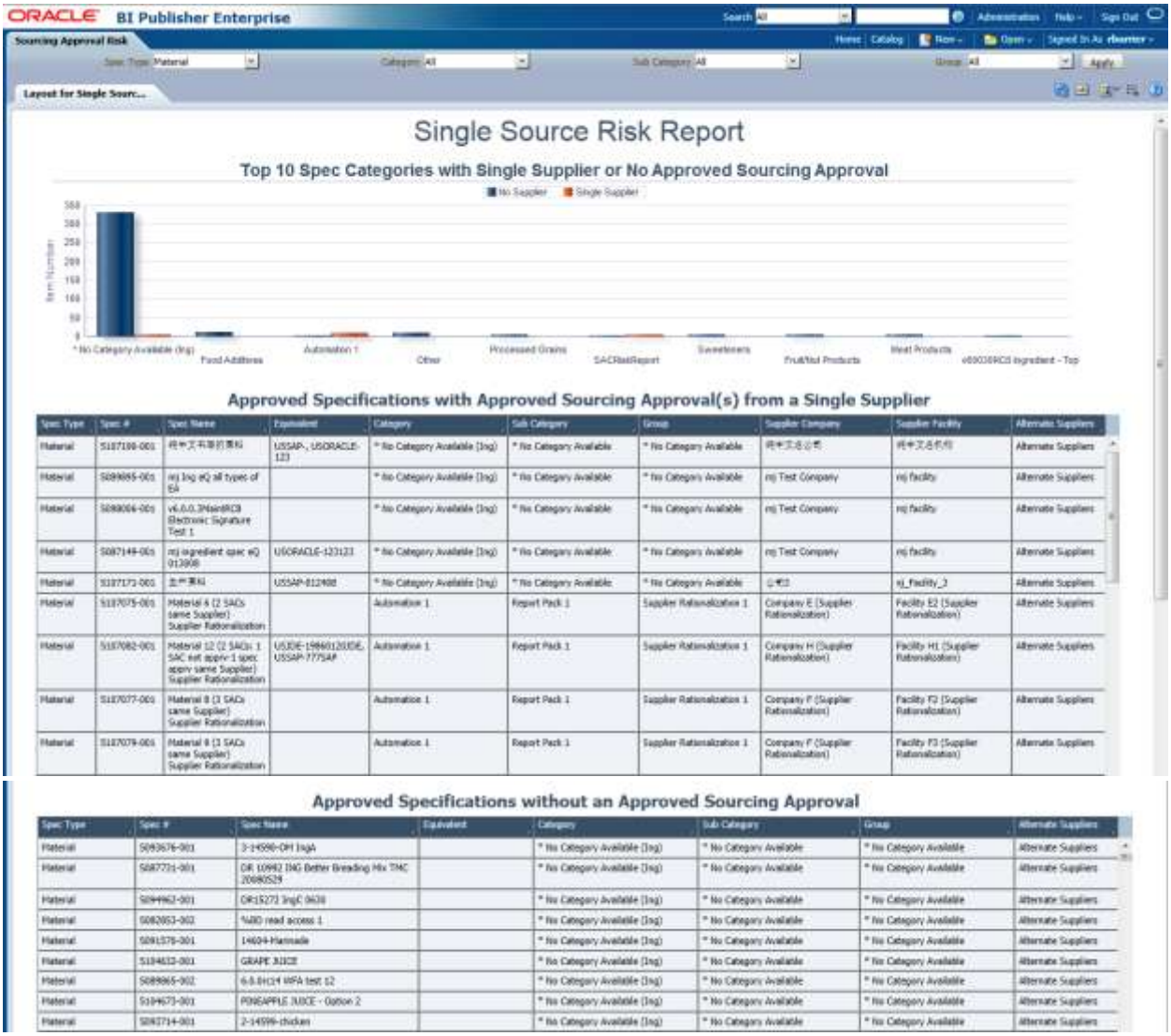
This report should be launched from a GSM specification or a specification listed in the hierarchy navigator panel. It grants the user easy access to the suppliers for the entire GSM hierarchy. You can quickly see all of the suppliers involved, their status, when their next scheduled audit date is and even if they have quality issues reported against them. The report has seven parameters: SCRM Business Unit Level 1, 2, and 3, Sourcing Approval Status, “Is Approved” Tag, Spec and Level. Is Approved is looking for the “Is Approved” tag on the sourcing approval status. This allows you to see only approved suppliers. The Level parameter decides whether it pulls documents from the entire hierarchy or the single specification level. The report contains one tab.

IMPORTANT: Hierarchy Denormalization Service is required for this report. You can learn more about this service in the *Agile Product Lifecycle Management for Process Hierarchy Denormalization Guide.*

Sourcing Approval Risk (SCRM)

What materials are only sourced from a single supplier?

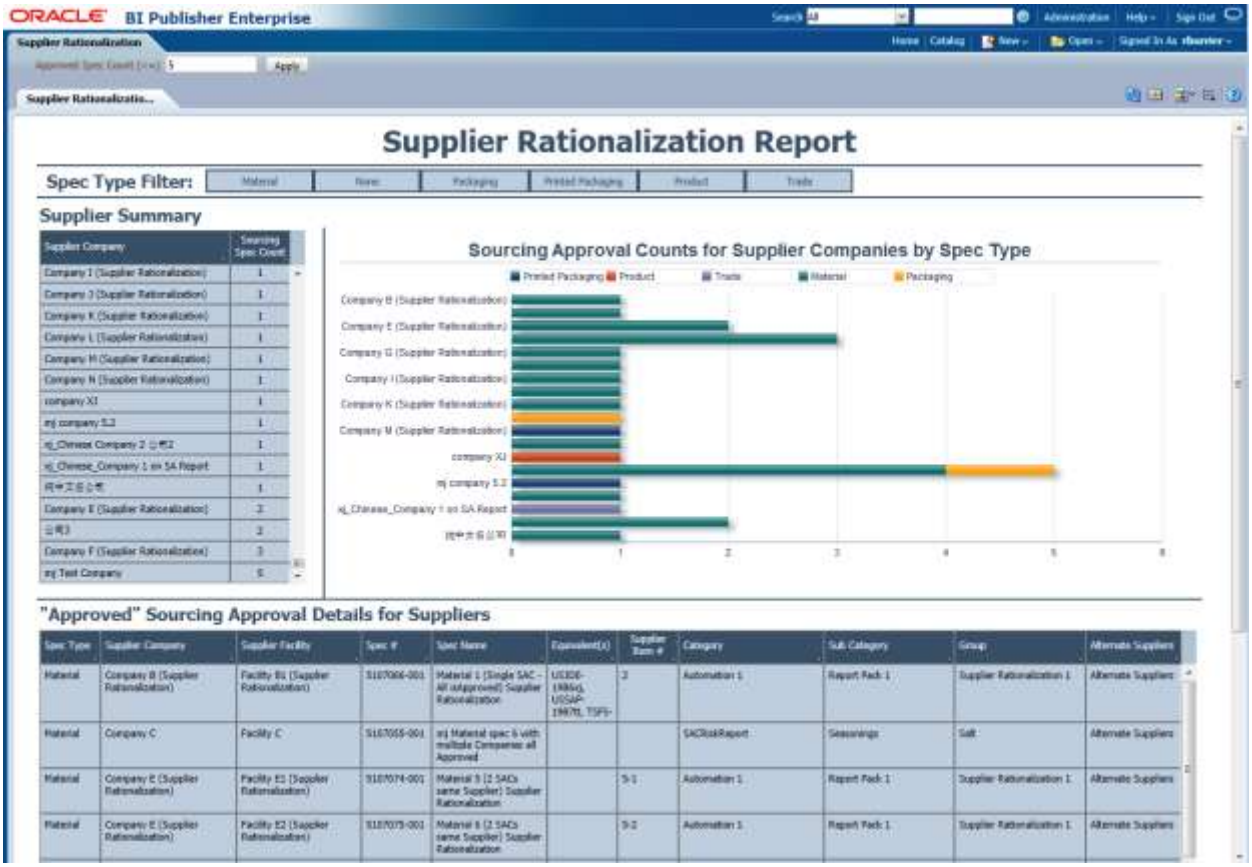
The Sourcing Approval Risk report was intended to be utilized by purchasing personnel to identity packaging material specifications, printed packaging specifications, and material specifications that have only “approved” sourcing approvals from one supplier (company), or no sourcing approvals at all. The report has four parameters: Spec Type, Category, Subcategory, and Group. This report contains one tab (or view) called Sourcing Approval Risk.



Supplier Rationalization (SCRM)

What suppliers are being underutilized?

The Supplier Rationalization report was intended to be utilized by purchasing personnel to identify suppliers that have limited approved sourcing approvals for GSM specifications. The report has one parameter called “Approved Spec Count (<=)”, and when the user enters a number into parameter, the report pulls all suppliers that have a total number of approved specifications with an approved sourcing approval “equal to” or “less than” the parameter value entered. This report contains one tab (or view) called Supplier Rationalization.



Supplier Compliance Report – Companies & Facilities (SCRM)

Show me all supplier documents for companies and facilities.

The Supplier Compliance Report: Companies and Facilities was designed to be utilized by users that manage supplier documents that are attached to companies or facilities. When this report is executed, it pulls all "Non-Archived" SDM documents for companies and facilities and exposes details about supplier documents including Days until Due and Days until Expired. The report has eight parameters: SCRM BU lvl 1, SCRM BU lvl 2, SCRM BU lvl 3, Company Name, Facility Name, Document Type, Days until Due Date and Days until Expiration Date. This report contains one tab (or view) called Supplier Compliance Report – Companies and Facilities.

Note: The SCRM BU parameters were by designed limited to three levels. If there are more levels defined in a specific implementation, the report still pulls documents from lower levels, but you will not be able to limit the report by the lower BU levels without modifying the report.

Supplier Compliance Report – Sourcing Approval (SCRM)

Show me all supplier documents for sourcing approvals.

The Supplier Compliance Report: Sourcing Approval was designed to be utilized by users that manage supplier documents that are attached to sourcing approvals. When this report is executed, it pulls all "Non-Archived" SDM documents for sourcing approvals and exposes details about supplier documents including Days until Due and Days until Expired. The report has ten parameters: GSM BU lvl 1, GSM BU lvl 2, GSM BU lvl 3, Company Name, Facility Name, Receiving Facility, Spec Type, Document Type, Days until Due Date and Days until Expiration Date. This report contains one tab (or view) called Supplier Compliance Report – Sourcing Approvals.

Note: The GSM BU parameters were by designed limited to three levels. If there are more levels defined in a specific implementation, the report will still pull documents from lower levels, but you will not be able to limit the report by the lower BU levels without modifying the report.

Supplier Compliance Report – Supplier Portal (SCRM)

Show the supplier all documents for their associated company, facility or sourcing approvals.

The Supplier Compliance Report – Supplier Portal report was designed to be launched from a navigation extension from within Supplier Portal and to be utilized by suppliers that manage supplier documents that are attached to companies, facilities or sourcing approvals. When this report is executed, it pulls all "Non-Archived" SDM documents for companies, facilities, and sourcing approvals and exposes details about supplier documents including Days until Due and Days until Expired. This report contains one tab (or view) called Supplier Compliance Report – SP. As with all supplier portal reports, this report should be released in PDF format using the custom report configuration method.

Chapter 3—Custom Report Configuration

Overview

The Reporting application allows client to organize, configure, secure, and launch custom reports. The configuration is managed in the `CustomerReportExtensions.xml` located in the `\Config\Extensions\` directory. Clients can configure custom reports, specify the categorization of the reports, configure visibility rules via custom classes, and define the various report parameters to display.

Two grouping levels categorize reports: Report Contexts and Report Groups, each of which can be secured by configuring security classes.

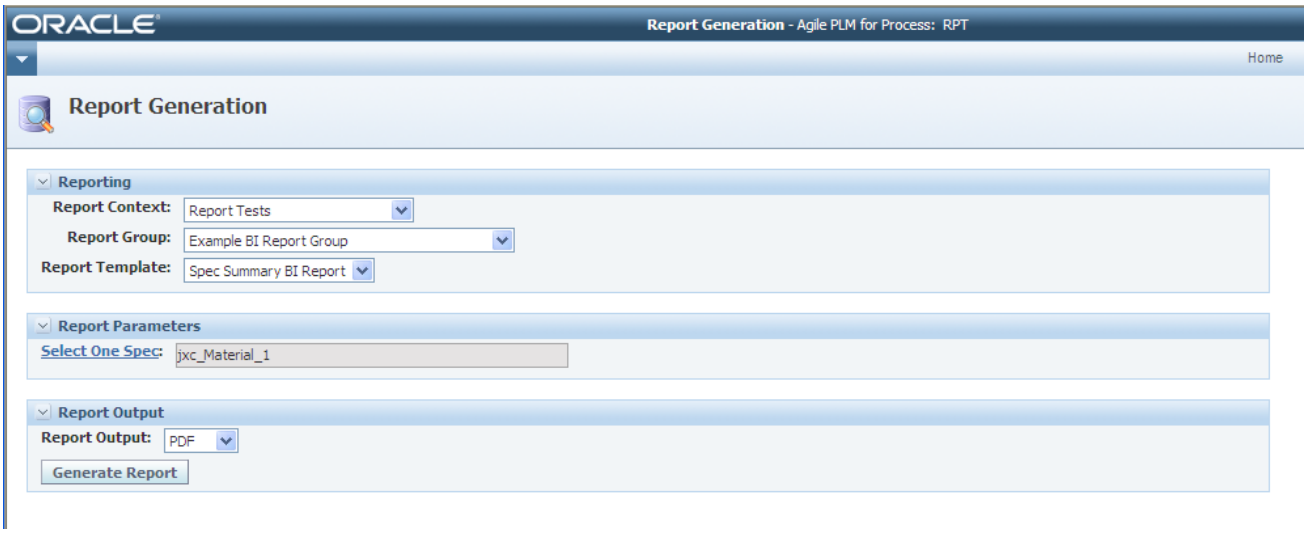
Report parameters can use existing pop-ups found throughout the application, or use custom-defined parameters. The parameter values are then sent to a reporting service, such as Oracle BI Publisher or SQL Server Reporting Services, which process the parameters to make custom SQL queries to produce the report output. The reporting service then returns the results to the user.

For details on configuring the reporting service, see the *Agile Product Lifecycle Management for Process Configuration Guide*.

Report Generation Screen

The report generation screen allows users to select an available report, enter report parameters, select an output format, and launch the report.

Figure 2. Sample Report Generation Screen



The Report Generation screen is comprised of three main sections:

1. Reporting—Provides a way to organize, categorize, and secure reports.
2. Report Parameters—Presents parameter input and options for each report.

3. Report Output—Presents the possible reporting output formats, such as XML and PDF.

The data that is displayed to users is driven from the `CustomerReportExtensions.xml` file. This guide explains how to configure this file for custom reporting needs.

Configuring the CustomReportExtensions.xml File

Customizing the report configuration primarily consists of defining any new report parameters types that will be available to the reports, and configuring the custom reports. Any changes to the CustomerReportExtensions.xml file will require the ProdikaReporting web application to be restarted.

File Structure Overview

Common objects, such as report parameter input types and common reports are defined in the ServiceConfig section. The reports, their categorization, security, and parameter details are defined in the ReportContexts section. The Contextual Reporting section defines ways to launch specific reports from the various web applications (such as GSM, SCRM) based on several criteria.

A high-level outline of the Report Contexts and ServiceConfig hierarchies found in the `CustomerReportExtensions.xml` file follows:

- ServiceConfig
  - ParameterTypes
    - ParameterType
  - CommonReports
- ReportContexts
  - ReportContext
    - ContextCriteria
    - ReportGroup
      - Parameters
      - Reports
        - Report
          - AllowedOutputTypes

Reports are categorized by two grouping levels: Report Contexts and Report Groups.

- Report Contexts represent the highest level of report organization, and contains multiple report groups. These are displayed in the Report Context drop-down list.
- Report Groups provide a way to group multiple reports that all use the same report parameters. The parameters types specified for each ReportGroup are defined in a separate section called ServiceConfig.

The following sections contain detailed information on how to configure each of these nodes.

Configuring Report Parameter Types

All report parameter types that are used by the reports must be defined in the `\ServiceConfig\ParameterTypes` section. These parameters may then be referenced by the reports defined in the `\ReportContexts` section. Several predefined report parameters are available for use, but you may also create their own custom parameter types.

There are three different types of report parameters:

1. Simple type parameters – Parameter input types, such as a string or date field input
2. DataSource parameter types – Classes that are responsible for retrieving data lists for display
3. Models – Predefined Oracle Agile PLM data retrievers that are already used in the application

Each parameter type must specify a user interface element that will be used to display the report parameter to the user. DataSource and Model report parameter types must also specify the method used to retrieve the data to display.

Global Report Parameters

The following report parameters are automatically passed in and available to each report server request:

1. `paramCurrentUser` – The current user's PKID (40 character unique identifier string)
2. `paramCurrentLanguage` – The current user's language ID setting (see SupportedLanguages table)

Note: Currently these are only passed to Microsoft SQL Server Reporting Services.

Note: Using DataSources and Models requires detailed technical knowledge of the Oracle Agile PLM for Process application and may require writing custom classes. See *Appendix A—Creating Custom Report Parameter Types* for details.

<ParameterTypes> Node

The ParameterTypes node contains a list of ParameterType nodes. Each ParameterType defines an input parameter that may then be used to display selection criteria for reports.

Attributes

Attribute Name	Description	Required?
N/A		

Child Nodes

- <ParameterType>

Parent Node

- <ServiceConfig>

<ParameterType> Node

The ParameterType node defines an input parameter that may then be used to display selection criteria for reports. Each Parameter Type must also declare the user interface control to use for displaying the report parameter to the user. The child nodes required are based on the specified web control.

Attributes

Attribute Name	Description	Required?
type	Unique identifier for this parameter type. This value will be used to refer to the parameter type when configuring which parameters are used for each report in a Report Group.	Yes
webControl	The name of one of the following available web controls, or a custom control: ▪ ReportingControls/**StringInput**.ascx – An input box that allows alphanumeric entry ▪ ReportingControls/**DateInput**.ascx – A date picker control preset to the current date ▪ ReportingControls/**MonthYearInput**.ascx – A month selection control and a year input control ▪ ReportingControls/**LookupInputSingleSelect**.ascx – An drop-down input allowing a single selection from data specified by a DataSource ▪ ReportingControls/**LookupInputMultiSelect**.ascx – A pop up control allowing multiple selections from data specified by a DataSource ▪ ReportingControls/**TreeViewInput**.ascx – A pop up control for nested hierarchies, allowing a single or multiple selections from data specified by a DataSource ▪ WebCommon/Controls/**EQTInput**.ascx – A pop up control with differing behavior based on which existing EQT View is used The following controls are available in the ReportingExtensions package ▪ ~/WebCommon/Extensions/Reporting/**ConstantValueParameter**.ascx – A hidden control that is set to some constant value, to be passed in automatically to the report. ▪ ~/WebCommon/Extensions/Reporting/**SessionParamCurrentSupplierUser**.ascx – A hidden control that is set to a specific property of the current supplier user (such as the supplier rep PKID, email, etc), if used via SupplierPortal ▪ ~/WebCommon/Extensions/Reporting/**SessionParamCurrentUgmUser**.ascx – A hidden control that is set to a specific property of the current UGM user (such as the user PKID, email, etc). ▪ ~/WebCommon/Extensions/Reporting/**SecuredObjectControl**.ascx – A hidden control that retrieves a given PKID from the Request, using the supplied RequestParameterName (such as SpecID, CompanyID, etc), and checks the user's security of the object specified by the PKID. The security check is performed by the specified SecurityHandler class. Several SecurityHandler classes are provided, but custom classes can be created using the Oracle.PLM4P.ReportingExtensions.Security.IHostObjectSecurityHandler interface See *Appendix A—Creating Custom Report Parameter Types* for more details on how each web control functions.	Yes

Child Nodes

The possible child nodes will depend on which web control is used:

- StringInput, DateInput: No child controls
- MonthYearInput : Allows for customizing the parameter names for the month and year controls
  - <MonthParamNameExtension> – InnerText should be set to the name extension
  - <YearParamNameExtension> – InnerText should be set to the name extension
- LookupInputSingleSelect, LookupInputMultiSelect:
  - <DataSource>
- TreeViewInput:
  - <DataSource>
  - <MultiSelect> – InnerText should be 'Yes' to enable selection of multiple entries, or 'No' for single selection.
- EQTInput:
  - <Model>

Parent Node

- <ParameterTypes>

<DataSource> Node

The DataSource element is used to specify a class that will retrieve data to be used in the given web control. Existing DataSources may be used to retrieve data, or custom DataSource classes may be created to fulfill more specific requirements.

Attributes

Attribute Name	Description	Required?
className	The full path of the DataSource class to use, along with any parameters. Format should be in the form of an Object Loader URL. Please see *Appendix C—Object Loader URLs* for details. Example: `Class:Xeno.Web.UI.Controls.DataSources.LookupServiceListViewDataSource,XenoWebControls$Allergens|Allergens`	Yes

Child Nodes

- N/A

Parent Node

- <ParameterType>

<Model> Node

The Model element is used to specify an existing search model (EQTModel) that will retrieve data to be used in the given EQTInput web control. The searches available through the EQT models may be much more detailed and allow the user to specify exact search criteria.

Attributes

Attribute Name	Description	Required?
displayVariableIndex	Indicates which return value from the EQT popup selection should be displayed. When an EQT view returns a value for display, the /DisplayColumns/ColumnInfo element (from the Model node) includes an attribute `provideInSelectJS="true"`. If multiple display columns in the EQT Model have the provideInSelectJS="true" attribute, the `displayVariableIndex` attribute determines which one should be displayed to the user. For Models that only return one display value, the displayVariableIndex attribute should be set to "1".	Yes

InnerText Value

- The path to the EQT Model. See *Appendix A—Creating Custom Report Parameter Types* for more details.

Child Nodes

- N/A

Parent Node

- <ParameterType>

Configuring Reports

<ReportContext> Node

The ReportContext node is used to group and secure related Report Groups together. All available Report Contexts are displayed to the user in the Report Context drop-down list of the Report Generation Screen.

Attributes

Attribute Name	Description	Required?
Name	The name of the report context that will be displayed to the user in the Report Generation screen.	Yes
SecurityFactoryRef	The security plug-in to use to limit access to this Report Context. See *Appendix B—Applying Security to Reports* for more details.	No

Child Nodes

- <ReportGroup>
- <ContextCriteria>

Parent Node

- <ReportContexts>

<ContextCriteria> Node

This node is currently not used.

Child Nodes

- <Parameters>

Parent Node

- <ReportContext>

<ReportGroup> Node

The ReportGroup node is used to group and secure related Report Groups together. All available Report Contexts are displayed to the user in the Report Context drop-down list of the Report Generation Screen.

Attributes

Attribute Name	Description	Required?
name	Used to uniquely identify this report group in the configuration files.	Yes
label	The display name that will appear to users or the translation lookup key. See isLabelTranslatable below.	Yes

Attribute Name	Description	Required?
isLabelTranslatable	**true** – Uses the value of the label attribute to look up the report group name in the translation caches, using the translationCache attribute, and returns the translation. **false** – Uses the value of the label attribute as the display name of the report group.	No
translationCache	The name of the translation cache used to look up the translation for the label value, if the isLabelTranslatable attribute is set to true.	Yes if isLabelTranslatable is true
securityFactoryRef	The security plugin to use to limit access to this Report Context. The value must be a valid Object Loader URL. See *Appendix B—Applying Security to Reports* for more details.	No
serviceLocation	If present, will use the given value as the web service location to use to process all reports in this report group; otherwise, will use the serviceLocation attribute value in the\ServiceConfig\ReportingService element.	No
reportEngineName	If present this will specify which report engine all reports within that group will use. The options are: ▪ SQLReportingService ▪ OracleBIPublisher10_1_3_3 ▪ OracleBIPublisher10_1_3_4 ▪ OracleBIPublisher11_1_1_5 Refer to the Install/Upgrade Guide for setting the report engine configuration entries. If no report engine is specified it will use the default report engine: SQLReportingService Note: the default report engine can be changed in the EnvironmentSettings.config file, by setting the default=”true” attribute to the desired reportEngine entry in the /ProdikaReporting/ServiceConfig child nodes.	No

Child Nodes

- Parameters
- Reports

Parent Node

- <ReportContext>

<Parameters> Node

The Parameters node contains a list of Parameter nodes. Each Parameter will be used as input criteria for the reports in the current ReportGroup.

Attributes

Attribute Name	Description	Required?
N/A		

Child Nodes

- Parameter

Parent Node

- <ReportGroup>

<Parameter> Node

The Parameter node will be used as an input criterion for the reports in the current ReportGroup. The parameter declared here uses a Parameter Type that must be defined in the ServiceConfig section of the configuration.

Attributes

Attribute Name	Description	Required?
name	The name of the parameter that will be passed to the Report Server. Must be unique within the <Parameters> node.	Yes
label	The display name that will appear to users or the translation lookup key. See isLabelTranslatable below.	Yes
isLabelTranslatable	**true** – Uses the value of the label attribute to look up the parameter name in the translation cache of the parent ReportGroup and returns the translation. **false** – Uses the value of the label attribute as the display name of the parameter.	No – Will use false if not specified
type	The name of the ParameterType from the \ServiceConfig\ParameterTypes list.	Yes
Required	**true** – This parameter will be a required criterion for launching the report. **false** – This parameter will be an optional criterion for launching the report.	No – Will use false if not specified

Child Nodes

- N/A

Parent Node

- <Parameters>

<Reports> Node

The Reports node contains a list of Report elements. Each Report will use the same Report Parameters as input criteria in the current ReportGroup.

Attributes

Attribute Name	Description	Required?
N/A		

Child Nodes

- <Report>

Parent Node

- <ReportGroup>

<Report> Node

The Report node represents an individual report that can be launched using the report parameters defined by the current report group. Reports can reference previously declared common reports and use the attribute values defined for that original report.

Attributes

Attribute Name	Description	Required?
id	Used to uniquely identify reports that can be reused throughout the reporting configuration. If any other report in the configuration matches the ***id*** value, an error is thrown. See the *Configuring Common Reports* section on page 56.	No
idref	May be used to reference a previously declared Common Report entry. The ***idref*** value must match the unique ***id*** value of the desired report. If a matching reference report is found, no other attributes from this element are used. If no matching report is found, an error is thrown. See the *Configuring Common Reports* section on page 56.	No
name	Used to uniquely identify this report in the configuration files within the report group.	Yes, unless ***id*** attribute is used
label	The display name that will appear to users or the translation lookup key. See isLabelTranslatable below.	Yes

Attribute Name	Description	Required?
isLabelTranslatable	**true** – Uses the value of the label attribute to look up the report name in the translation cache of the parent ReportGroup and returns the translation. **false** – Uses the value of the label attribute as the display name of the report.	No – Will use false if not specified
reportPath	Path to the individual report on the reporting server.	Yes
serviceLocation	If present, uses the given value as the web service location to use to process this report; otherwise, uses the ReportGroup's value.	No
translationCache	If present, uses the given value as the name of the translation cache to use to process this report's label; otherwise, uses the ReportGroup's value.	No
SecurityFactoryRef	The security plug-in to use to limit access to this report. The value must be a valid Object Loader URL. See *Appendix B—Applying Security to Reports* and *Appendix C—Object Loader URLs* for more details.	No
template	The name of the BI Publisher layout template to use for the report output.	Yes if using BI Publisher for report generation; No otherwise
reportEngineName	If present this will specify which report engine this report will use. The options are: ▪ SQLReportingService ▪ OracleBIPublisher10_1_3_3 ▪ OracleBIPublisher10_1_3_4 ▪ OracleBIPublisher11_1_1_5 Refer to the Install/Upgrade Guide for setting the report engine configuration entries. If no report engine is specified it will use the default report engine: SQLReportingService Note: the default report engine can be changed in the EnvironmentSettings.config file, by setting the default="true" attribute to the desired reportEngine entry in the /ProdikaReporting/ServiceConfig child nodes.	No

Child Nodes

- <AllowedOutputTypes>

Parent Node

- <Reports>

<AllowedOutputTypes> Node

The AllowedOutputTypes node contains a list of OutputType nodes indicating the possible output formats available for the report. Individual reports therefore restrict the full list of allowed output types defined in the \ServiceConfig\ActiveReportOutputTypes node.

Attributes

Attribute Name	Description	Required?
N/A		

Child Nodes

- AllowedOutputType

Parent Node

- <Report>

<AllowedOutputType> Node

The AllowedOutputType node indicates a possible output format available for the given report. Entries are limited to values defined in the reporting framework's list, such as EXCEL, PDF.

Attributes

Attribute Name	Description	Required?
key	Unique identifier for the output type	Yes
value	The display name	Yes
extension	File extension (ex: ".pdf")	Yes

Child Nodes

- N/A

Parent Node

- <AllowedOutputTypes>

Configuring Common Reports

A <Report> node may also be defined in the /ServiceConfig/CommonReports section of the configuration and then referenced in other report groups. This allows a report to be defined once, but used multiple times. If, for instance, a particular report exists in multiple report groups, it would be beneficial to define the report in this section, and then reference it in those report groups.

For example, the following section declares a common report that can then be used anywhere else in the configuration:

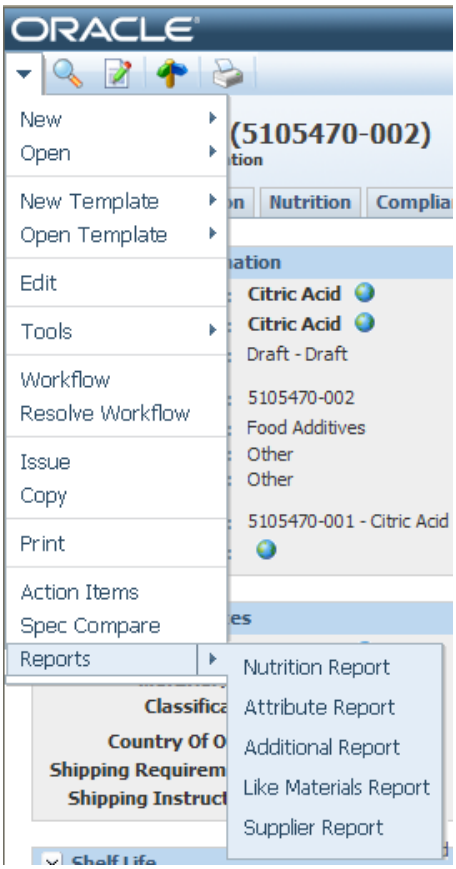
```
<CommonReports configChildKey="name" >
    <Report id="ExampleCommonReport" label="Example Common Report"
isLabelTranslatable="false" reportPath="/ExampleReport/ExampleCommonReport">
        <AllowedOutputTypes configChildKey="key">
            <OutputType key="PDF" value="PDF" extension=".pdf"/>
        </AllowedOutputTypes>
    </Report>
</CommonReports>
```

The id attribute represents the unique key that will need to be referenced if this report is to be included elsewhere. To include this report in a ReportGroup, simply declare the Report node with the idref attribute assigned to the above id. For example, the following ReportGroup declares a new report and also includes the ExampleCommonReport from the CommonReports section:

```
<Reports configChildKey="name">
    <Report name="ExampleReportA" label="Example Report A "
isLabelTranslatable="false" reportPath="/ExampleReport/ExampleReportA">
    </Report>
    <Report idref="ExampleCommonReport" />
</Reports>
```

Configuring Contextual Reporting

Individual Reports can be launched from most locations throughout the application suite (GSM, SCRM, NPD, PQS, CACS, NSM, CC, eQ Admin) by using Contextual Reporting.



Report links are added by extending the Action Navigation and adding new menu item nodes. Each new menu item entry specifies which report context, report group, and individual report should be loaded, and can send specific data to the reporting application to pre-populate specific report parameters. The visibility of each report link can be configured easily, such as specifying that the report link is only available when the user is on a Packaging specification and if the user is a member of group “Packaging Reports.” See the Navigation Extensibility Guide for more information around how to add new navigational items and configure their visibility.

Clicking the report link will launch the Reporting application in a popup window, allowing the user to maintain their current context.

You can display the active object's name inside your report parameter fields. For example, if the user is on the Cajun Spice Blend specification and the report takes in a specification parameter, the string "Cajun Spice Blend" can appear in the specification parameter field as a pre-selected value. For SpecInstanceReport items, the target Report must contain a Parameter with `name="paramObjectPKID"` using a parameter type that is one of the pop-up parameter types. Setting the parameter type this way will allow for the name of the object to be displayed. Setting the parameter type to a parameter type string will simply display the PKID itself.

For Contextual Reporting, the new navigation item must be assigned an XML attribute, called **ClientSideCommand**, which is configured with the following possible behaviors:

1. Launching reports using the report parameter screen, which allows users to select other report parameters
    a. **LaunchSpecificObjectPKIDReport** – will launch the specified report*, sending the current business object's PKID and name to the Reporting Application*. If the specified report has a report parameter with a name="paramObjectPKID", it will be pre-filled with the object's name for display, and the object's PKID for the value.
    Required parameters:
        i. report context name
        ii. report group name
        iii. report name

Example:

If you have a NLEA Fact Panel (report name: NLEAPanel) report inside the context Material Reports (context name: Material) and group Nutritional Reports (group name: Nutritional) and you want to call that report using contextual reporting you would use the following function:

LaunchSpecificObjectPKIDReport('Material', 'Nutritional', 'NLEAPanel');

If you want to show the navigation item on a material specification when it's in read mode, the menu item node would look something like this:

```
<MenuItem ID="GSMNavSpec" configChildKey="ID">
       <MenuItem ID="NLEAPAnel" DisplayText="NLEA Panel"
ClientSideCommand="LaunchSpecificObjectPKIDReport('Material', 'Nutritional',
'NLEAPanel');" Visible="eval:${ObjectType}==1004 &amp;&amp; ${IsInReadMode}" />
</MenuItem>
```

    b. **LaunchObjectTypeReport** – will launch the specified report, *sending the current business object's 4-digit object Type value* to the Reporting Application. If the specified report has a report parameter with a name="paramObjectType", it will be pre-filled with the object's 4-digit type for the parameter value.
    Required parameters:
        i. report context name
        ii. report group name

iii. report name

2. Automatically launching reports with no further user entry. This feature requires the ReportingExtensions package, available in the Utils folder, and contains a report launch page and various report parameter controls.
   To launch a report automatically, the ClientSideCommand specifies the following:
   a. Open a new popup window using a URL pointing to the /Reporting/WebCommon/Extensions/Reporting/AutoReportLauncher.aspx, available in the ReportingExtensions package.
   b. the report context, group, and report name using a pipe delimiter and assigning it to the ReportConfig parameter
      i. For example: ReportConfig=Material|Nutritional|NLEAPanel
   c. the current object's parameter name and assigning it to the Javascript function GetObjectHeader_ObjectPKID()
      i. For example: . . . &SpecID=' + GetObjectHeader_ObjectPKID()
   d. The popup window properties

   For example:

```
<MenuItem ID="GSMNavSpec" DisplayText="**GSM" IsInEditMode="${IsInEditMode}" >
       <MenuItem ID="lblOpenIssuesBySpec" Text="Open Issues By Spec Report"
              ClientSideCommand="var win =
window.open('/Reporting/WebCommon/Extensions/Reporting/AutoReportLauncher.aspx?ReportCo
nfig=PQM Reports|OpenIssues|OpenIssuesbySpec&amp;amp;SpecID=' +
GetObjectHeader_ObjectPKID(),'Reporting',
       'height=600,width=800,status=no,toolbar=no,menubar=no,location=no,dependent=yes,
scrollbars=yes,resizable=1');win.focus();" />
       </MenuItem>
```

Note that for auto launching reports, all report parameters for the given report must be configured using the IsHidden xml element. For instance, the sample report, Open Issues By Spec, is configured using two report parameters, IssueSpecific and SpecID:

```
<ReportGroup name="OpenIssues" label="Open Issues by Spec" isLabelTranslatable="false"
translationCache="Reporting">
  <Parameters configChildKey="name">
       <Parameter name="IssueSpecific" label="IssueSpecific" isLabelTranslatable="false"
type="Constant_IssueSpecific" Required="true"/>
       <Parameter name="SpecID" label="SpecID" isLabelTranslatable="false"
type="SecuredSpec" Required="true"/>
  </Parameters>
  <Reports configChildKey="name">
    <Report name="OpenIssuesbySpec"  ReportEngine="OracleBIPublisher11_1_1_5" label="Open
Issues by Spec" isLabelTranslatable="false"
   reportPath="/PLMforProcess_Reference/PQM/Reports/Open Issues by Spec.xdo"
template="Open Issues">
      <AllowedOutputTypes configChildKey="key"><OutputType key="PDF" value="PDF"
extension=".pdf"/></AllowedOutputTypes>
    </Report>
```

```
  </Reports>
</ReportGroup>
```

There report parameters use the Constant_IssueSpecific and the SecuredSpec report Parameter Types, defined as follows:

```
<ParameterType type="Constant_IssueSpecific"
webControl="~/WebCommon/Extensions/Reporting/ConstantValueParameter.ascx" >

  <Value>True</Value>
  <IsHidden>true</IsHidden>

</ParameterType>

<ParameterType type="SecuredSpec"
webControl="~/WebCommon/Extensions/Reporting/SecuredObjectControl.ascx" >

<SecurityHandler>Class:Oracle.PLM4P.ReportingExtensions.Security.UGMUserSpecSecurityHandler,Re
portingExtensions</SecurityHandler>

  <RequestParameterName>SpecID</RequestParameterName>
  <IsHidden>true</IsHidden>
</ParameterType>
```

Note that the RequestParameterName (SpecID) in the SecuredSpec ParameterType must match the parameter name assigned to the GetObjectHeader_ObjectPKID() function in the URL above.

Adding Custom Reports to Supplier Portal

You can now provide links to custom reports in Supplier Portal.

Important Note: Remember security should be the primary concern when providing reports to suppliers. If the supplier can select parameters, only allow them to select objects they have access to.

Setting Up Contextual Custom Reports in Supplier Portal

You set up a Supplier Portal report in the same way as any other custom report. See the example below for a specific example. Notice the name “SupplierReports”.

```
<ReportContext name="SupplierReports" configChildKey="name">

       <ReportGroup name="QualityReports" label="Quality Reports Group"
isLabelTranslatable="false" translationCache="Reporting">

               <Parameters configChildKey="name">

               <Parameter name="paramSpecID" label="Select One Spec"
isLabelTranslatable="false" type="SupplierQuality" Required="true"/>

               </Parameters>

               <Reports configChildKey="name">

               <Report name="QualitytBIReport" label="Quality BI Report"
isLabelTranslatable="false" reportPath="/PLM for Process Sample Reports/Sample Spec Summary
Report/Sample Spec Summary Report.xdo" template="ExampleLayout">

               <AllowedOutputTypes configChildKey="key">

                      <OutputType key="PDF" value="PDF" extension=".pdf"/>

                      <OutputType key="EXCEL" value="EXCEL" extension=".xls"/>

               </AllowedOutputTypes>

               </Report>
```

```
</Reports>
</ReportGroup>
</ReportContext>
```

However, Supplier Portal does not use the Reporting web application to launch reports.

Add Links to Supplier Portal

You can add a link to the report using the following format:

```
<MenuItem ID="lblSampleReport" Text="Sample Report" ClientSideCommand="var
win =
window.open('/SupplierPortal/Popups/SimplePopup.aspx?ContentKey=SupplierRepor
ting&amp;ReportConfig=<REPORT_CONTEXT_NAME>|<REPORT_GROUP_NAME>|<REPORT_NAME>
','Prodika_SupplierReporting','titlebar=no,width=800,height=450,left='+(((scr
een.width-10-800)/2)+0)+',top='+(((screen.height-35-450)/2)+0));win.focus();"
/>
```

NOTE: The "ReportConfig" should be a combination of the report context, group name and report name using the pipe delimiter (|).

Example 1: Add a Report to the Primary Navigation

```
<MenuItem ID="SpecificationAndDocumentsListing" configChildKey="ID">

      <MenuItem ID="lblQualityReports" Text="Quality Reports"
configChildKey="ID">

            <MenuItem ID="lblSampleReport" Text="Sample Report"
ClientSideCommand="var win =
window.open('/SupplierPortal/Popups/SimplePopup.aspx?ContentKey=SupplierRepor
ting&amp;ReportConfig=SupplierReports|QualityReports|QualitytBIReport','Prodi
ka_SupplierReporting','titlebar=no,width=800,height=450,left='+(((screen.widt
h-10-800)/2)+0)+',top='+(((screen.height-35-450)/2)+0));win.focus();" />

      </MenuItem>

</MenuItem>
```

You can add custom report command to object navigation also. To achieve this, please add the menu item config as the following format:

```
<MenuItem ID="lblSampleReport" ClientSideCommand="var win =
window.open('/SupplierPortal/Popups/SimplePopup.aspx?ContentKey=SupplierRepor
ting&amp;ReportConfig=<REPORT_CONTEXT_NAME>|<REPORT_GROUP_NAME>|<REPORT_NAME>
&amp;
SpecId=${SpecID}','Prodika_SupplierReporting','titlebar=no,width=800,height=4
50,left='+(((screen.width-10-800)/2)+0)+',top='+(((screen.height-35-
450)/2)+0));win.focus();" />
```

This difference with primary navigation is that you should supply one more parameter: SpecId=${SpecID}

Example 2: Add a Custom Report Link to Material Specification Object Navigation

```
<MenuItem ID="1004">

      <MenuItem ID="lblSampleReport" ClientSideCommand="var win =
window.open('/SupplierPortal/Popups/SimplePopup.aspx?ContentKey=SupplierRepor
ting&amp;ReportConfig=SupplierReports|QualityReports|QualitytBIReport&amp;Spe
cId=${SpecID}','Prodika_SupplierReporting','titlebar=no,width=800,height=450,
left='+(((screen.width-10-800)/2)+0)+',top='+(((screen.height-35-
450)/2)+0));win.focus();" />

</MenuItem>
```

Chapter 4—BI Publisher Integration

Customers that want to execute custom reports using Oracle BI Publisher are able to do so. There are BI Publisher reports available as reference implementations, known as Report Packs. This section discusses how to use the reference implementations and develop new reports with BI Publisher.

PLM for Process Reporting Database Layer

To facilitate reporting, database objects such as views, stored procedures and tables were created and are part of what is called the PLM for Process Reporting Database Layer. The Reporting Database Layer is a prerequisite for the Report Packs. Installation instructions and a description of the Reporting Database Layer can be found in the Feature Pack starting with release 3.1, in the ReferenceImplementations\ReportingDBLayer directory. Follow the installation instructions found there.

BI Publisher Installation

It is recommended to always install the latest BI Publisher release with all the patches. BI Publisher documentation can be found here: http://www.oracle.com/technetwork/middleware/bi-foundation/documentation/bi-pub-096318.html

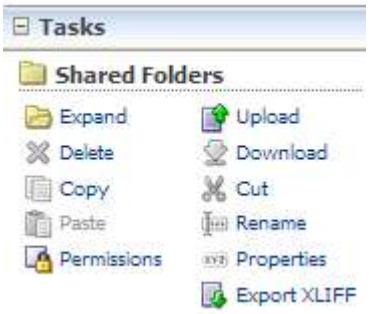
Installing a PLM for Process Reference Implementation Report Pack

Because it is believed that reports delivered by Oracle will be customized for each implementation, the reports are delivered as a reference implementation. The reports are intended to be copied to another BI Publisher directory where they can then be customized. If they are modified in the original directory then they might be overwritten with the installation of the next version of the Report Pack.

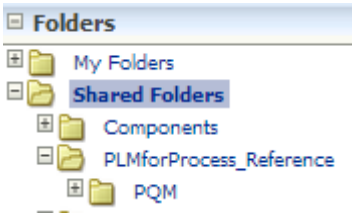
The Report Packs are found in the Feature Pack starting with release 3.1, in the ReferenceImplementations\ReferenceReportPacks. They are created by the PLM for Process Module and can be installed one at a time.

Uploading the Report Folder

There is one .xdrz file per Reporting Pack. This file is a BI Publisher binary file which is the folder containing all the reports and the BI Publisher Data Models for both Oracle and SQL Server. This file should be uploaded to the 'Shared Folders' directory in BI Publisher. To upload this file you will need to go to: 'http://{host}:{port}/xmlpserver/servlet/catalog' where {host} and {port} are those of your BI Publisher installation. When you use this URL, the 'Upload' functionality becomes available in the lower left panel as shown here:

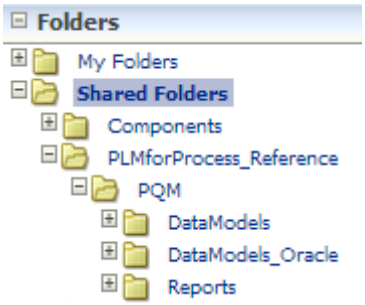


After uploading this file, there should be a new folder under 'Shared Folders' called 'PLMforProcess_Reference'. Under this folder there will be one folder per Report Pack. The following image shows the example of the PQM Report Pack installed. If not uploaded to this location the reports will not function.



Configuring the Proper Data Model – Oracle or SQL Server

Out of the box, under the Reporting Pack folder such as 'PQM', there are 2 Data Model folders, one for Oracle and one for SQL Server. They are named 'DataModels_' with either 'Oracle' or 'SQL Server' appended to it. You will need to rename the one you want to use to just 'DataModels'. If you like you can delete the other one. Below is an example of the Data Models configured to use SQL Server.



Copy Reference Reports to another Directory for Customization

All the reports in the Report Packs are reference implementations and should not be modified directly. Create another directory and copy the Report Pack directory into it.

In order for the reports to use the intended BI Publisher Data Models (the one in the new folder), each report will have to be update to point to the Data Model in the newly created folder. If this is not done then the reports will be using the Data Models in the reference implementation.

Report Security

There are different types of security that can be applied depending on how the report is called.

Report is Called using PLM for Process Reporting Application

1. Report level: The report itself can be secured by using the UserGroupBasedSecurityPluginFactory or the UserRoleBasedSecurityPluginFactory explained in Appendix B – Applying Security to Reports.
2. Parameter Level: PLM for Process can perform security verification on the parameters passed to the report using the SecuredObjectControl.ascx, described above.

BI Publisher Report is Called Directly either through a Link or the BI Publisher UI

1. Report level: BI Publisher roles can be used to secure access to the report. See BI Publisher documentation for more detail.
2. Parameter Level:
    - **For SQL Server** – A query can be executed in the report which then calls a function to validate if the user has access to the parameter. If the user does not have access then the query should generate an exception which will not return any data for the report.
    - **For Oracle** – A BI Publisher 'Before Data' Event Trigger could be used to call an Oracle function that validates if the user has access to the parameter. There are some examples of this in the PQM Reporting Pack. One report is 'OpenIssuesWithActions_BySpec'. You can see how the specification is secured by looking at the event trigger and the function that is called.
      When a user does not have access to the parameters there will be no data returned for the report. They will not receive a message that the user does not have access.
      To help the report writer test and debug the security functionality, they can turn on security debugging which is explained in the Reporting Database Layer documentation which can be found in the ReferenceImplementations\ReportingDBLayer of the Feature Pack.

If accessing the report through BI Publisher directly, the users must login in order to take advantage of the security options. If securing the parameters the user's name will have to match that in PLM for Process, since that is what is being used to check authorization. It is recommended that a Single Sign On solution be used between PLM for Process and BI Publisher.

Multilingual

There are two aspects to multilingual support.

1. Labels: The report labels are controlled by the user's setting in BI Publisher. By setting the Report Locale and UI Language in 'My Account' within BI Publisher, the report labels can be translated. In the reference implementation there are no translation files, so customers will have to translate the labels if needed.
2. Data: The data that is returned from PLM for Process can be multilingual. The Reporting Packs are enabled for this for Oracle only. BI Publisher provides the ability to call an Oracle function

when the database connection is established. The Reporting Packs use this feature to enable multilingual data. The function that is called, sets a database session variable to the language ID of the PLM for Process user. The queries executed will then take advantage of this variable. The views in the Oracle Reporting DB Layer are designed to use this. The PLM for Process user will have to have the same login name as the BI Publisher user. The function that is called is configured in the 'Pre Process Function' field within the BI Publisher Data Source.

Calling a BI Publisher Report Directly from a Link in PLM for Process

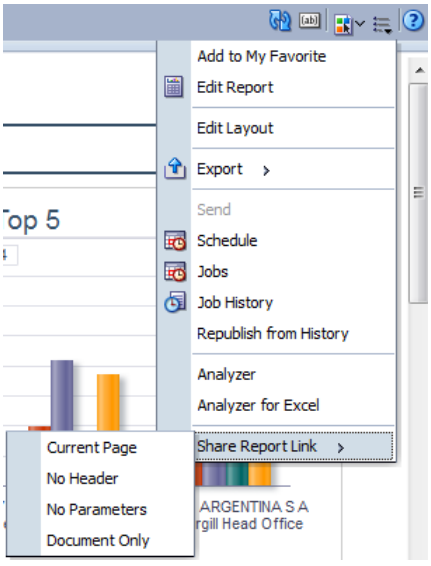
Calling a BI Publisher report directly from a link, allows users to take advantage of the BI Publisher Interactive output format. The interactive format allows for report filtering by selecting components of charts and lists. See the BI Publisher documentation for more information about the interactive output format.

Another advantage is that is that configuration is simpler. In other words a report does not need to be configured in the PLM for Process Report Application.

The disadvantage is that in order to secure the report and parameters a Single Sign On solution will need to be used in order to maintain a consistent user across the applications.

To configure a link within PLM for Process, please see the *Agile Product Lifecycle Management for Process Navigation Configuration Guide*. This guide shows how to access contextual data depending on the active object. For instance if the link is on a specification, it is possible to access the Spec Type and Spec ID and if on a facility the Facility ID can be accessed. This data can be passed to a report. To construct the report URL to be called follow the below steps:

1. Execute a report which takes parameters, directly in BI Publisher.
2. Pull down the menu from the Actions icon on the top right side of the report as shown below.



3. Select 'Share Report Link' and then choose one of the child menu items, depending on what you want.

4. A popup will then display the URL to access the report. Copy this URL and paste it into a text editor.
5. Replace the parameters values with the proper JavaScript functions or variables explained in the Navigation Extensibility Guide.
6. Convert the URL to a valid ASCII format i.e. URL Encoding.

For example a report taking the CompanyID as a parameter could be launched from a company profile within SCRM. The CompanyID could be retrieved and inserted as a parameter in a URL by first getting the data using the GetObjectHeader_ObjectPKID() JavaScript function and then constructing the URL with it. The URL would look like this after URL Encoding:

http://mysBIPServer:7001/xmlpserver/PLMforProcess/PQM/Individual%20Supplier%20Quality.xdo?_xpf=&_xpt=0&_xdo=%2FPLMforProcess%2FPQM%2FSupplierManagement%2FIndividual%20Supplier%20Quality.xdo&_xmode=2&xdo%3Axdo%3A&xdo%3Axdo%3A&xdo%3Axdo%3A&_paramsCompanyID='*+GetObjectHeader_ObjectPKID()+*'&_xt=Action%20List%20RTF&_xf=html&

This link holds display options such as the output format and whether or not the parameters should appear.

Once this link is configured and it appears on a company page, a user can click it and they will be taken to the report. If Single Sign On is set up, the report will be launched with the same user name that they were logged into PLM for Process with.

If Single Sign On is not configured, the user will either be presented with a BI Publisher login screen or if the report is configured for 'Guest' access they will be taken to the report.

Report Dialog

Sometimes the complexity of a report solution makes it very difficult for an end user to understand what report to execute. In these situations we recommend using the Report Dialog application. The Report Dialog will allow you to present selections to a user and then the application will determine which report to execute. You can think of this application as a report URL builder. You can see a working example of the Report Dialog by installing the Fact Panel Report Pack, see the Fact Panel Reporting Guide for more information.

Example Fact Panel Report Dialog Form & Fields

Installing Report Dialog

1. Run the Report Dialog Installer, found in Apps/ReportDialog/ReportDialogSetup.exe

2. Select the Application you are planning to launch the report dialog from.

Configuring Report Dialog

Below are instructions for creating your own Report Dialog Form.

1. **Create a Form** by adding an entry to the REPORT_DIALOG_FORM table

REPORT_DIALOG_FORM Table

PKID	New PKID
FORMID	Unique ID string representing the form.
RESULT_DATASOURCE_URL	The source of the form results table. We offer an out of the box datasource class that allows you to point to an existing database table. Class:Oracle.Agile.PlmProcess.ReportDialog.Lib.DataSource.DBResultDataSource,ReportDialog$<result-table-name>

```
insert into REPORT_DIALOG_FORM ( PKID, FORMID, RESULT_DATASOURCE_URL ) SELECT <NewPkid>
', '<FormID>',<ResultDatasourceURL>' FROM DUAL  WHERE NOT EXISTS(SELECT 1 FROM
REPORT_DIALOG_FORM WHERE PKID = '<NewPKID> ');
```

2. **Create Fields** by adding them to the REPORT_DIALOG_FIELD table

REPORT_DIALOG_FIELD Table

PKID	New Pkid
ID	Unique ID string representing the field. Do not use spaces
FIELD_TRANSLATION_ID	The UI label shown with the field. See *Adding a Translation* below.
FIELDTYPE	Currently only drop downs are supported. This should be populated with "DD"
FK_FORMID	Form PKID from the REPORT_DIALOG_FORM table
SEQUENCE	Sequence the field will appear in the UI
DATASOURCECLASS	Datasource class pointing to the values used to populate the field. You can add your own custom lookups using the Qualitative Lookup Configuration Toolkit Widget. Please name these new lookups using the following naming convention [CUSTOMERNAME]_[DATASOURCENAME]. Use the out of the box datasource class to point to custom lookups: 'Class:Oracle.Agile.PlmProcess.ReportDialog.Lib.DataSource.CustomLookupDataSource,ReportDialog$<Datasource Category>|<DEFAULT VALUE –EXTERNAL ID>'
PARAM_NAME	The BI Publisher parameter name that this field will map to.

```
insert into REPORT_DIALOG_FIELD ( PKID, ID, FIELD_TRANSLATION_ID, FIELDTYPE, FK_FORMID,
SEQUENCE, DATASOURCECLASS, PARAM_NAME ) SELECT '<newpkid>', '<ID>', '<TranslationID>',
```

```
'DD', '<formPKID>', <Sequence>, '<ClassPath>', '<ParameterName>' FROM DUAL  WHERE NOT
EXISTS(SELECT 1 FROM REPORT_DIALOG_FIELD WHERE PKID = '<newpkid>');
```

Adding a Translation

A. Add a Translation Cache

```
insert into commonXLAExtensionCache ( pkid, Name, InheritFrom, Category ) SELECT
'<1058-uniquecachepkid>', N'ReportDialog/<formid>', N'ReportDialog/<formid>', N'Report
Dialog' FROM DUAL  WHERE NOT EXISTS(SELECT 1 FROM commonXLAExtensionCache WHERE pkid =
'<1058-uniquecachepkid>');
```

B. Add a Translation Item

```
insert into commonXLAExtensionCacheItem ( pkid, fkParent, langID, Id, Value ) SELECT
'<1059-uniqueitempkid>', '<1058-xlacachepkid>', 0, N'<field-translation-id>',
N'<actual-field-translation-value>' FROM DUAL  WHERE NOT EXISTS(SELECT 1 FROM
commonXLAExtensionCacheItem WHERE pkid = '<1059-uniqueitempkid>' and langID = 0);
```

3. **Create Dependencies** by adding them to the REPORT_ FIELD_DEPENDENCY table

You can specify which fields are dependent on other fields. This allows you to control which selection options are available when the user has made a selection in a previous field. For example, Field 1 has the options A, B, C. Field 2 is dependent on Field 1 so Field 2 won't be populated until a Field 1 selection has been made.

FROM_FIELD_PKID	PKID of dependent field. Field 2 in the example above.
TO_FIELD_PKID	PKID of controlling field. Field 1 in the example above

```
insert into REPORT_FIELD_DEPENDENCY ( FROM_FIELD_PKID, TO_FIELD_PKID ) SELECT
'<dependent-pkid>', '<controlling-pkid>' FROM DUAL WHERE NOT EXISTS(SELECT 1 FROM
REPORT_FIELD_DEPENDENCY WHERE FROM_FIELD_PKID='<dependent-pkid>' AND TO_FIELD_PKID='<
controlling -pkid>');
```

4. **Declare Form Results** in custom table named [CUSTOMERNAME]_[FORMID]

You now need to set up your report dialog form results. Create a custom table named [CUSTOMERNAME]_[FORMID]. This name should match the RESULT_DATASOURCE_URL in the REPORT_DIALOG_FORM table. Each row of this table will describe an available report. The Report Dialog application will use these rows to determine the report to direct the user to.

FORMID	Required Column. FormID from the REPORT_DIALOG_FORM table
FIELD_ID(s)	A column should be created for each field added to the REPORT_DIALOG_FIELD table. The name of the column should match the field ID. This column is populated with the External ID of the data source item.
HOSTTYPE	Required Column. The object type (Use Type ID, such as 1004 etc) the report

	will be launched from. For example, when the report dialog is launched from a material spec (1004) it will show these options.
REPORT_FILENAME	Required Column. The file name of the BI Publisher report.
REPORT_ID	Required Column. A unique string representing the report. Will be used in the ReportDialog.xml configuration file.
PARAMETER_CLASS	Required Column. Adds additional parameters including the active host object pkid. You can write your own class here or use the default class that's available. Class:Oracle.Agile.PlmProcess.ReportDialog.Lib.Common.BasicSpecParameterGenerator,ReportDialog$<param-name for active object> This class will send the active object pkid and all field values using the field ID as the parameter name. See *Apps/ReportDialog/ReferenceImplementations* for more information.

Form Result Example

Below is an example Form Result Table. This form only has 3 fields: Region, Format and Language. There are a total of 4 reports.

FORMID	REGION	FORMAT	LANG	HOSTTYPE	REPORT_FILENAME	REPORT_ID	PARAMETER_CLASS
MYFORM	REGION_1	SIMPLE	0,1,2,3	1004	Region1Simple.xdo	Region1Simple	Class:Oracle.Agile.PlmProcess.ReportDialog.Lib.Common.BasicSpecParameterGenerator,ReportDialog$MatSpec
MYFORM	REGION_1	COMPLEX	0	1004	Region1Complex.xdo	Region1Complex	Class:Oracle.Agile.PlmProcess.ReportDialog.Lib.Common.BasicSpecParameterGenerator,ReportDialog$MatSpec
MYFORM	REGION_2	SIMPLE	0	1004	Region2Simple.xdo	Region2Simple	Class:Oracle.Agile.PlmProcess.ReportDialog.Lib.Common.BasicSpecParameterGenerator,ReportDialog$MatSpec
MYFORM	REGION_2	COMPLEX	0,1,2,3	1004	Region2Complex.xdo	Region2Complex	Class:Oracle.Agile.PlmProcess.ReportDialog.Lib.Common.BasicSpecParameterGenerator,ReportDialog$MatSpec

5. **Map Reports** to a BI Publisher report folder in Extensions/ReportDialog.xml

   Create a REPORTFORM node for each form you created. This node will tell the system where to find each report in BI Publisher.

   Create a REPORTDIRECTORY node for each unique BI Publisher directory where the reports are located. The ReportID attribute allows you to use RegEx. For example, you can declare a specific directory for all report IDs that start with US by setting the ReportID to "US*"

You can also add additional static parameters to the URL by adding the attribute "ADDITIONALPARAM". For example, if you added ADDITIONALPARAM="_xmode=3" to the node the report launched will not display parameter choices to the user.

```
<ReportDialog>
    <ReportForm FORMID="<FORMID>">
        <ReportFilter ClassURL="Class:Oracle.Agile.PLMProcess.ReportDialog.Lib.Common.ReportResultFilter,ReportDialog" />
        <ReportURLConfig>
            <!-- star special meaning applies to all -->
            <ReportDirectory REPORTID="<REPORTID>" URL="@@VAR:Prodika.ReportServer.URL@@/<BI FOLDER PATH>" />
            <ReportDirectory REPORTID="<REPORTID>" URL="@@VAR:Prodika.ReportServer.URL@@/<BI FOLDER PATH>" />
        </ReportURLConfig>
    </ReportForm>
</ReportDialog>
```

6. **Add a Link** in PLM4P to the new Report Dialog form

   You now need to provide a link in PLM4P that will open your new report dialog form. Do this by adding a new menu item using Navigation Extensions using the following ClientSideCommand:

```
window.open('/<APP>/WebCommon/ReportDialog/ReportDialog.aspx?MaintainSpec=true&amp;FORM
_ID=<FORMID>&amp;SPEC_ID='+GetObjectHeader_ObjectPKID(), '_blank',
'toolbar=no,height=500,width=500');
```

   See the Fact Panel Navigation item example below.

```
<MenuItem ID="lblFactPanel" MenuType="icon" DisplayText="**FactPanel"
SecurityProfileRef="FactPanelProfile" Visible="eval:(${ObjectType}==2147 ||
${ObjectType}==5750 || ${ObjectType}==5816) &amp;&amp; ${IsInReadMode}"
HotKey="alt+q" ClientSideCommand="window.open('/gsm/WebCommon/ReportDialog/ReportDialog
.aspx?MaintainSpec=true&amp;FORM_ID=FactPanel&amp;SPEC_ID='+GetObjectHeader_ObjectPKID(
), '_blank', 'toolbar=no,height=500,width=500');');" SortOrder="180" />
```

   You can learn more about how to add a navigation extension in the *Agile Product Lifecycle Management for Process Navigation Configuration Guide.*

Creating Links within a Report

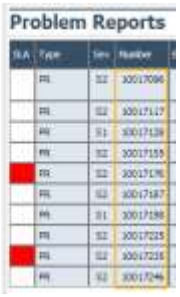
BI Publisher reports support http links typically in a cell within a table. The link is able to be constructed based on values being returned in the report. For instance if a table displays a list of material specifications, a link can be constructed for each row of the table to open the specification in PLM for Process. It is helpful to have the host name of the server generated dynamically based on a configuration. This way when moving between environments the reports do not have to change. Also if the server changes all that needs to change is the configuration.

The configuration for all the host names is stored in a single row of a table called 'ApplicationURLs', which is part of the Reporting DB Layer. In this table you can set the host name for the following applications:

- BI Publisher
- GSM
- SCRM

- NPD
- PQM

Examples of using this table to create links in a report can be found in the Reporting Pack Reference Implementations. The reports are built using the BI Publisher layouts. The URL is specified by clicking on a column in a table and then in the left hand pain opening the 'Properties' section. At the bottom there is a URL property that can be populated to construct the URL as shown below.



An example of a PQM URL would be the following.

http://{/DATA_DS/HOSTS/PQMURL}/pqm/MainPage/PQM.aspx?ContentKey=Issue&PQMItemID={/DATA_DS/ISS/IID}&InitialLoad=true

The information inside the curly braces is the path to the data elements as defined by the BI Publisher Data Model. It is case sensitive so remember to use upper case. This example shows how to dynamically create the host for the URL as well as the Item ID for the PQM Issue.

Oracle BIP Features

Parameters

Parameters are essentially global filters for a BI Publisher report. Once a parameter(s) are selected and applied, the report refreshes and only displays data specific to the applied parameters. For example, the Quality Analyst report uses two parameters: Company and Facility.

The user can choose a specific company or a facility to execute against the report, or the user has the option to execute the report against "All" companies or facilities. Parameters can be hidden, and under certain circumstances should be hidden to meet specific security requirements.

Interactive Filters

Interactive filters are configurable filters within a specific BI Publisher report. When the interactive filter is selected, specific analytics (as defined within the report) will update. For example, the Quality Analyst

report has an interactive filter for severity. When the user clicks S1, the chart will update to reveal only the information associated to “S1” severity.

The user can apply and remove multiple interactive filters with a report with ease, enabling the user to dive deep into the analytics without additional reports.

Conditional Formatting

Another powerful feature with Oracle BI Publisher is conditional formatting. Conditional formatting can be used to format against any data field in the data model with wide a range of operators to conditional format fonts, background colors and indentations. This feature is used throughout the PQM reports. In the following example, conditional formatting is used to color the first column to correspond to the SLA state of the PQM object; red, green, amber or no color if object is “closed”. Conditional formatting was also used to indent the arrow in column 2, and color the rows differently depending on the level defined the by PQM object relationships. Conditional formatting greatly enhances the usability of the reports by visually focusing the end user to PQM objects that need attention.

		Type	Sev	Number	Equivalent	Title	Status
		NCR	S4	10017150		PerformanceTesting_Issue61	Released
		SCAR		10000240	123456	PerformanceTesting_Action228	Draft
		SCAR		10000005		PerformanceTesting_Action1	Complete
		SCAR		10001421		PerformanceTesting_Action1409	Draft
		Audit		10083069	ABC	Segal Test 1	Draft
		SCAR		10000005		PerformanceTesting_Action1	Complete

Use of Workflow Tags in PQM Report Pack

Workflow Tags are used extensively in PQM Reporting to drive many of the analytics. The following workflow tags have been created for the PQM module:

- Cancelled
- Complete
- Pending
- Released
- Review
- Submitted

In many of the reports, a simplified status was used to identify PQM objects that are Cancelled, Closed, Pending or Open. The following table shows the mapping of the Workflow tags to corresponding simplified status.

Workflow Tag Name	Simplified Status
Cancelled	Cancelled
Complete	Closed
Pending	Pending
Released	Open
Review	Open
Submitted	Open

The BI Publisher data models that pull the information for PQM Reporting use this mapping in many of the reports to determine a particular PQM object's simple status. For example, many analytics only want to return "Open" PQM objects, if the workflow status is "tagged" with Released, Review, or submitted, the PQM Report analytic only displays open PQM objects. The use of these workflow tags allows end users to create custom workflows and still allows the out-of-the box reports to work.

Appendix A—Creating Custom Report Parameter Types

Methods of Retrieving Data

Clients can create custom report parameter types by leveraging existing data searches found throughout the application (via EQT Models) or by calling data retriever classes (DataSources) to retrieve the necessary search data.

Displaying Search Parameters

When specifying the report parameter type in the reporting configuration, a user interface control must also be declared. The web control that will display the parameter type will depend on which of the data retrieval approaches is used.

The Reporting module provides three web controls for DataSource parameters:

- ReportingControls/**LookupInputSingleSelect**.ascx – A drop-down input allowing a single selection from data specified by a DataSource
- ReportingControls/**LookupInputMultiSelect**.ascx – A pop up control allowing multiple selections from data specified by a DataSource
- ReportingControls/**TreeViewInput**.ascx – A pop up control for nested hierarchies, allowing a single or multiple selections from data specified by a DataSource

The EQT Model parameters all use the following web control:

- WebCommon/Controls/EQTInput.ascx – A pop up control with differing behavior based on which existing EQT View is used

EQT Models

The Entity Query Toolkit (EQT) is an internal framework used in Oracle Agile PLM for Process for mapping existing internal data models to search parameters and search results. This mapping is declared in configuration files that are available for reference. EQT Models define how a search occurs, including which search parameters are available, which search parameters are mandatory (and therefore hidden and always used), what data is returned, and how returned data is displayed.

Most searches throughout the application use the EQT framework. The Reporting framework can therefore leverage some existing EQT searches to create search parameters for the desired reports. However, some EQT searches used in the application may not function as needed when leveraged in EQT. For instance, one SCRM facility search allows the selection of a facility based on some search parameters. When selecting the facility, however, the facility name may not populate the web control in the reporting screen properly, even though the facility's unique ID (PKID) does get stored properly.

Caveat: The details of EQT are complex, difficult to interpret, and are comprised of many components. Additionally, EQT is an *internal* development tool only; although it may be leveraged in the Reporting module, it is not supported as a client facing tool. Therefore, a detailed review of EQT is beyond the scope of this document.

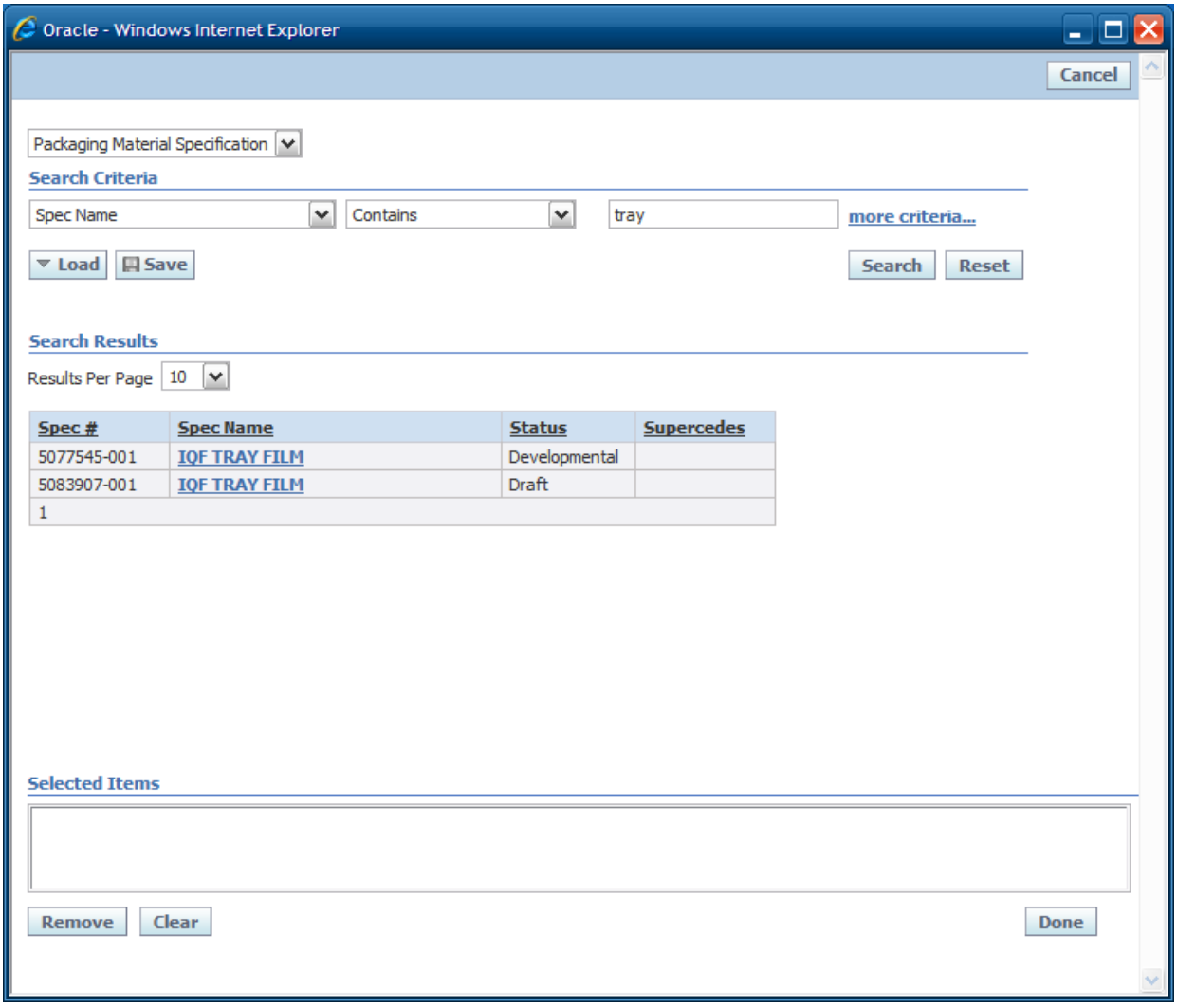
Oracle Consulting Services may be able to provide some assistance in leveraging EQT to meet reporting needs.

Identifying an EQT Search to Use

To leverage an existing EQT search, the name of the searchable EQT view must first be identified. For searches that occur in a pop-up window, the view name may be found by looking at the search pop-up window's properties listing (right click, select **Properties**) and examining the URL. The URL may include a parameter that references the name of the EQT view.

For instance, the URL of one popup includes the following parameter:

```
&DataSource=SearchableView:Config:ProdikaSettings/SearchableMultiSelectViews,
PackagingSpecViewForProcessAndTradeSpecs
```



The EQT search view name used in this popup is `PackagingSpecViewForProcessAndTradeSpecs`.

Return Value of the EQT Selection

Examining the EqtUIModelDefinitions.xml file and locating this view reveals some information that will be useful:

```
<PackagingSpecViewForProcessAndTradeSpecs . . . >
  <AllowedModels>
    <Model active="true" name="Packaging Specification" alias="lblType1009"
orderByColumns="-" captionColumn="1" primaryKeyColumnName="SpecID">
```

The attribute `primaryKeyColumnName` specifies which field is returned upon selection. In this instance, the SpecID property will be returned. This may be only partially informative, as the details of the model and its properties are hidden, but it generally can give an idea of what field value is returned.

Display Value of the EQT Selection

The display columns that contain an attribute `provideInSelectJS` are returned and accessible to the report parameter, so that they can fill the input box upon selection.

```
<DisplayColumns>
  . . .
  <ColumnInfo width="200" dataField="SpecName" dataFieldCaption="lblSpecName"
provideInSelectJS="true"/>
```

The columns where `provideInSelectJS` are set to true are then used by the <Model> node's `displayVariableIndex` attribute, representing the location/index of the desired property to display in the selected items input field. In this example, the SpecName field is returned as the only display item, and therefore its `displayVariableIndex` attribute would be set to 1.


```
<ParameterType type="ExampleEQTInput_PackagingSpecView"
webControl="ReportingControls/EQTInput.ascx">
   <Model
displayVariableIndex="1">SearchableView:Config:ProdikaSettings/EQTConfiguration/Search
ableMultiSelectViews,PackagingSpecViewForProcessAndTradeSpecs</Model></ParameterType>
```

Other Ways to Identify an EQT Model

While some EQT views can be identified by the URL of the pop-up, others views are harder to identify. Examining the `EqtUIModelDefinitions.xml` file and searching for the desired information may yield results.

Oracle Consulting Services may be able to provide more assistance in identifying a desired EQT model.

Available EQT Models

Currently, only the models in the following EqtUIModelDefinitions.xml sections are supported in the Reporting module:

- Views in the models/ProdikaReporting node (via SearchableMultiSelectViews)
- Views in the models/pqs node (via PQSSearchableMultiSelectViews)

For example, to create a report parameter to access the SpecSummaryView EQT view from ProdikaReporting, use the following entry:

```
<ParameterType type="ExampleEQTInput_SpecSummary"
webControl="WebCommon/Controls/EQTInput.ascx ">
<Model
displayVariableIndex="1">SearchableView:Config:ProdikaSettings/EQTConfiguration/Search
ableMultiSelectViews,SpecSummaryView</Model>
</ParameterType>
```

When configuring the Model node, the syntax above always begins with `SearchableView:Config:ProdikaSettings/EQTConfiguration/` and is followed by either `SearchableMultiSelectViews` (for the models in the ProdikaReporting node) or `PQSSearchableMultiSelectViews` (for the models in the PQS node), then a comma, followed by the name of the Model.

An additional and possibly more straightforward approach to creating custom search parameters is by using DataSources.

DataSources

DataSources are classes that implement a specific interface (discussed later) whose responsibility is to retrieve search result data for selection in a pop-up web control. Clients can leverage several existing DataSources, or implement their own classes and plug them in easily. DataSources are referenced by their full class name and package name. They may also include parameters in their declarations that are then used by the class to modify its behavior. Please see *Appendix C—Object Loader URLs* and some of the examples below for more details.

Here is a report parameter that uses an existing datasource (LookupServiceListViewDataSource):

```
<ParameterType type="ExampleGenericLookup_Allergens"
webControl="ReportingControls/LookupInputMultiSelect.ascx">
   <DataSource
className="Class:Xeno.Web.UI.Controls.DataSources.LookupServiceListViewDataSource,Xeno
WebControls$AllergensTitle|Allergens"/>
</ParameterType>
```

Existing DataSources

Several existing DataSources may be leveraged to retrieve data which only require parameters to customize their results. A few are outlined here.

LookupServiceListViewDataSource

There are many data lists throughout the application, such as Languages, NPDBrands, Allergens, etc. Most lists are managed through a service that uses the database table commonLookupServiceManagers to maintain each list and how the list is configured. The ManagerID column value of this table for the desired lookup data must be passed in as the second parameter and is used by the LookupServiceListViewDataSource class to retrieve that list.

See the Oracle Agile PLM Data Training for more details on how CommonLookups work.

In the example above, the DataSource used is as follows, and takes two parameters ("AllergensTitle|Allergens"), the first for the pop-up title, and the second for the lookupManagerID.

```
<DataSource
className="Class:Xeno.Web.UI.Controls.DataSources.LookupServiceListViewDataSo
urce,XenoWebControls$AllergensTitle|Allergens"/>
```

Some lookup items are status aware, so this Datasource allows for a third parameter to be passed in to indicate if only Active items should be displayed ('ForSearching'), or all historically valid items should be displayed ('StatusAwareConsumer').

GenericListSelectView

This DataSource is used for specifying a table and column to search on, and the return value for the selected item(s). Rather than pass in the table name, however, this DataSource expects a class name which it then uses to retrieve the table name (The tablename to classname mapping can be found in the database table `orclassmetainfo`). To specify the column to retrieve, this DataSource expects the class parameter name that will then be used to retrieve the column name (The property to column name mapping can be found in the `orpropertymetainfo` table). To specify the return value column name, this DataSource expects the class parameter name that will then be used to retrieve the column name.

Example SQL to find the classname and property name:

```
SELECT tablename FROM orclassmetainfo where classname='complieswith'

SELECT columnName, propertyName FROM orpropertymetainfo
where fkORClassMetaInfo = (select pkid from orclassmetainfo where
classname='complieswith')
```

An example of using the GenericListSelectView DataSource for CompliesWith items, displays the Name of the compliesWith item, and returns the PKID for the selected item(s).

```
<ParameterType type="ExampleGenericTableLookup_CompliesWith"
webControl="ReportingControls/LookupInputMultiSelect.ascx">
   <DataSource
className="Class:Xeno.Web.UI.DataSources.GenericListSelectView,XenoWebControl
s$|CompliesWith|Name|PKID"/>
</ParameterType>
```

TaxonomyXNodeTreeViewDataSource

This DataSource is used to load multi-level taxonomies that are managed in the commonTaxonomyNamespace table. It requires a namespaceid parameter and must use the TreeViewInput control. There is also the option of making the selectable data a single- or multi-select.

An example of using this DataSource to display SCRM business units in a multi-select pop-up:

```
<ParameterType type="ExampleGenericTaxonomy_SCRMBusinessUnit"
webControl="ReportingControls/TreeViewInput.ascx">
   <DataSource
className="Class:Xeno.Web.UI.DataSources.TaxonomyXNodeTreeViewDataSource,XenoWebContro
ls$SCRMBusinessUnit" />
  <MultiSelect>true</MultiSelect>
</ParameterType>
```

Custom DataSources

Clients may also create a custom DataSource class to implement their own data retrieval operations. The class must implement the Xeno.Web.UI.Controls.IListView interface, and must include implementations of the following interfaces from the Xeno.Web.UI.Controls namespace:

The IListView interface returns an IListDataSource, which must enumerate through the results and return an IListItem for each entry.

```
public interface IListView
{
      IListDataSource ListItemsByPrefix( String prefix );
      IListDataSource ListAllItems();
      String Title{ get; }
      bool AllowAddNew{ get; }
      void AddNewListItem( IListItem item );
}

public interface IListDataSource
{
      IListItem Current{ get; }
      bool Next();
}

public interface IListItem
{
      String Value{ get; }
      String DisplayValue{ get; }
}
```

The custom DataSource class must be compiled and placed into the Web\ProdikaReporting\bin directory.

Appendix B—Applying Security to Reports

Overview

The Reporting framework allows customers to configure security rules that govern if a Report or Report Group should be visible to the current user. The `SecurityFactoryRef` attribute of the Report and ReportGroup nodes can be declared with different security plug-ins using Object Loader syntax. Each security plug-in takes parameters in its declaration that determine the access restrictions for that Report or Report Group.

Because Report nodes are children of the ReportGroup node, security is enforced hierarchically. When considering access to a Report, the user must have access to the parent Report Group.

Existing Security Plug-ins

The following report security plug-ins are available:

UserRoleBasedSecurityPluginFactory

This plug-in secures a Report, Report Group, or Report Context based on the roles to which a user belongs. The specified roles are evaluated using an OR relationship, meaning that if the user belongs to any of the roles listed, they allowed access. Negation is allowed through the use of the "!" (exclamation mark) operator as a prefix to a role name.

Example

```
SecurityFactoryRef="Class:Xeno.Prodika.SecurityModel.Contextual.UserRoleBased
SecurityPluginFactory,ProdikaLib$[SCRM_COMPANY_READER]|[SCRM_FACILITY_READER]
"
```

Note that the brackets surrounding the role name are required.

UserPropertyBasedSecurityPluginFactory

This plug-in secures a Report, Report Group, or Report Context based on the user's application access. The specified applications listed are evaluated in an OR relationship. It is important to note that this type of security considers a user's "HasAdminAccess" property, and if so, allows access regardless of the supplied permissions.

Example

```
SecurityFactoryRef="Class:Xeno.Prodika.SecurityModel.Contextual.UserPropertyB
asedSecurityPluginFactory,ProdikaLib$SCRM"
```

UserGroupBasedSecurityPluginFactory

Note that this plug-in is available in the Feature Pack, under the Handlers package. Be sure to add the Handlers.dll file to the web\ProdikaReporting\bin directory.

This plug-in secures a Report, Report Group, or Report Context based on the user's group access. The specified groups listed are evaluated in an OR relationship. There are several ways to specify the group: using the group name or the group PKID. Negation is allowed through the use of the "!" (exclamation mark) operator as a prefix to a group name or PKID. Additional options are detailed in the Feature Pack documentation.

Example

```
SecurityFactoryRef="Class:Xeno.Prodika.SecurityModel.Contextual.UserGroupBase
dSecurityPluginFactory,Handlers$Group123|Group456"
```

Appendix C—Object Loader URLs

Object Loader URLs are classpaths that are used to dynamically load objects. They are used to declare the protocol to use when loading the class, the class path, and optionally any parameters to pass to the class.

Format

[Protocol] : [Path] $ [[param1] | param2]

- Protocol - Examples are "Class" and “Singleton".
- Path - The fully qualified class name, including the package name. For example `"Xeno.Prodika.SecurityModel.Contextual.UserRoleBasedSecurityPluginFactory,ProdikaLib"` where ProdikaLib is the name of the package (.dll file).

When loading an object, the loader first inspects the protocol and using lazy loading, determines an appropriate protocol handler based on this protocol’s name. The "Class" protocol may refer to a class that accepts parameters during instantiation which are defined after a "$" and delimited by "|"s (pipes).

Common Usage

The most common usage of this class is in configuration files. Often a factory class is supplied in a configuration and the Object Loader bootstraps the factory, which in turn facilitates the use of the rest of the implementation. These implementations are easily swapped by simply providing a different factory in the configuration.

Example

Class:Xeno.Prodika.Portal.WebUI.Util.Security.UserPropertyBasedSecurityPluginFactory,ProdikaLib$NPD

"Class" is the protocol, "NPD" is a parameter, and the rest of the string between the ":" and the "$" is the path as defined by the protocol. In this case, it is the class path of the object that is to be instantiated.

Appendix D—Third Party External Application Integration

The Oracle Agile Product Lifecycle Management for Process application allows customers to integrate with reports that exist in an external reporting engine. Specifically, customers can configure the application suite to gather information from within the application and provide well-formatted reports that are generated in an external system.

Currently, the PLM for Process application supports only two reporting engines:

- Oracle BI Publisher
- SQL Server Reporting Service

This document outlines the steps necessary to integrate the PLM for Process application with any other third party external application that can provide similar functionality.

Programmatic Interfaces

In order to achieve this integration, two classes must be created that conform to PLM for Process interfaces.

1. IServiceFactory
2. IReportGenerationService

Implementing IReportGenerationService

The PLM for Process application expects all implementations of its report integration adapters to implement the IReportGenerationService interface:

```
namespace Xeno.Prodika.Services
{
    public interface IServiceFactory
    {
        IService Create(IExecutionContext execContext, IBlockingResourcePool
servicePool);
    }
}
```

As the sample code illustrates, this class has two purposes:

i. Retrieve configuration information from the ReportingServiceConfig

ii. Initialize and return a Service that will be used to manage integration with a reporting engine (which in this case is called ParameterReportGenerationService)

```
public class ParameterReportGenerationServiceFactory : IServiceFactory
{
  private ReportingServiceConfig _config;

  public IService Create(IExecutionContext execContext, IBlockingResourcePool
servicePool)
```

```
  {
    ExtractConfiguration();
    return new ParameterReportGenerationService(_config);
  }

  private void ExtractConfiguration()
  {
    if (_config != null)
      return;
    IXMLConfigurationManager cm = (IXMLConfigurationManager)
AppPlatformHelper.ApplicationManager.EnvironmentManager.GetConfigManager("ReportingSer
viceConfig");

    _config = (ReportingServiceConfig) cm.GetConfig("ReportingServiceConfig");

    if (_config == null)
      throw new ProdikaConfigurationException("Could not read
ReportingServiceConfig");
  }
}
```

Implementing IReportGenerationService

The PLM for Process application expects all implementations of its report integration adapters to implement the IReportGenerationService interface:

```
namespace Xeno.Reporting.Service.ReportGenerationService
{
  public interface IReportGenerationService : IService
  {
    ReportingServiceConfig Config { get; }
    ReportResults GenerateReport(ReportConfig config, ITranslationCache
translationCache, string format, IDictionary parameters);
  }
}
```

This interface provides two capabilities:

i. The ability to access configuration information

ii. A method that can be converts a request for a report into the result of a report invocation.

```
internal class ParameterReportGenerationService : IReportGenerationService
{
  private readonly ReportingServiceConfig _config;
  public ParameterReportGenerationService(ReportingServiceConfig config)
  {
    _config = config;
  }

  public ReportResults GenerateReport(ReportConfig config, ITranslationCache
translationCache, string format, IDictionary parameters)
  {
    ReportResults result = new ReportResults();
    result.ReportContent = BuildParameterReport(parameters);
    result.MimeType = format;
    return result;
  }
```

```
  private static byte[] BuildParameterReport(IDictionary parameters)
  {
    StringBuilder builder = new StringBuilder();
    foreach (DictionaryEntry parameter in parameters)
    {
      builder.AppendFormat("Parameter '{0}' has value: '{1}'<br/>", parameter.Key,
parameter.Value);
    }
    return new ASCIIEncoding().GetBytes(builder.ToString());
  }

  public ReportingServiceConfig Config
  {
    get { return _config; }
  }
  public IServiceContext ServiceContext
  {
    get { return new ServiceContextAdapter(); }
    set { ; }
  }
}
```

In our example, the configuration information was passed into this class upon creation. All that needs to be done is to provide it back in order to fulfill the contract of the IReportGenerationService.

The generation is also reasonably straightforward. The following information is provided to the method that implements this behavior:

i. The ReportConfig, which contains information on which report is being requested as well as its service location

ii. The translation cache that has been configured for this particular report

iii. The format of the result

iv. All parameters that have been picked by a user and are being passed into the report

As can be seen, the sample code in question does not make a call to a reporting engine. Instead, it displays all parameters that have been provided to it in HTML.

A more realistic implementation would access the Name and ServiceLocation values off the ReportConfig object and then use that to drive which report needed to be invoked.

It should be noted that the approach taken by PLM for Process does not assume a specific protocol (SOAP/DB/etc). That is a decision best left up to the implementer.

Configuration Changes

In order for the PLM for Process application to use custom code for report integration, you must register it at the appropriate location. In this particular case, the correct location is the Config/Custom/CustomerSettings.config file.

For our sample implementation, the entry for SQLReportingService in the configuration file would be modified to this if the following assumptions about the class that implements IServiceFactory were true:

i. It is named ParameterReportGenerationServiceFactory and is in namespace ParameterReportingServiceAdapter

ii. It is contained in the ParameterReportingServiceAdapter.dll

iii. The DLL file is available to all web applications (i.e., it is present in each Web/*/bin directory)

```
<SQLReportingService refscope="Session"
factory="Class:ParameterReportingServiceAdapter.ParameterReportGenerationServiceFactor
y,ParameterReportingServiceAdapter"
  configChildKey="name"
  configAttributeOverrideModifier="Replace"  > </SQLReportingService>
```

PLM for Process must be configured with the user name and password in order integrate properly. To set this value, please refer to the “Setup Assistant” section of the *Agile Product Lifecycle Management for Process Configuration Guide*.