

IBM Predictive Maintenance and Quality
Version 2 Release 6

Solution Guide



Note

Before you use this information and the product it supports, read the information in "Notices."

This edition applies to version 2.6.1 of IBM Predictive Maintenance and Quality and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation , 2017.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Introduction	vii
Chapter 1. Introduction	1
What's new in version 2.6.1	2
Accessibility features	2
Notices	2
Trademarks	4
Terms and conditions for product documentation	5
IBM Online Privacy Statement	5
Chapter 2. Configuring the product	7
Identification of the assets, resource types, event types, and measurement types	7
Create a custom application	8
Integration with asset management and manufacturing execution systems	9
IBM Predictive Maintenance and Quality dashboard application	9
Creating a dashboard	10
Chapter 3. Orchestration	13
Message flows	13
Example of an orchestration XML file	15
Generic batch orchestration	16
Chapter 4. Master data	21
Master data process	21
File format and location	22
Master data using InfoSphere MDM Collaboration Server	24
IBM Master Data Management Collaboration Server dynamic references	25
Creating a company in IBM InfoSphere MDM Collaboration Server	26
Configuring the IBM InfoSphere MDM Collaboration Server user interface	27
Guidelines for managing data in IBM InfoSphere MDM Collaboration Server	27
Configuring and running data exports	28
Importing metadata into InfoSphere MDM Collaboration Server	29
Solution XML file	30
IBM Maximo Asset Management	32
How master data is mapped in IBM Maximo Asset Management	32
Mapping master data in IBM Maximo Asset Management	34
Enabling master data loading in realtime mode	36
Importing event data from IBM Maximo Asset Manager	37
Creating a work order service in IBM Maximo Asset Management	37
Configuring work orders in Maximo	38
Mapping work orders for maintenance	46

Chapter 5. Event data	49
How events are processed	49
Event definition	50
Flat file event input	51
Schema definition of the event format	53
Profile and KPI tables	53
Profile variables	53
KPI tables	54
Profiles	56
Profile calculations	57
Custom calculations	59
Predictive scoring	59
Events and actual, planned, and forecast values	60
Event processing queue	60
Event processing	60
Remove events	61
Configuring solution.xml for the event flow	62
Chapter 6. Quality early warning system use cases	65
Quality inspection	65
Business and technical challenges	66
Defining the quality inspection solution	66
Quality inspection solution details	67
Results and benefits	72
Warranty	72
Business and technical challenges	74
Defining the warranty solution	74
Warranty solution details	74
Results and benefits	84
Parametric	84
Business and technical challenges	86
Defining the parametric solution	86
Parametric solution details	87
Results and benefits	94
Chapter 7. Situational Awareness and UI and Service Framework	95
Managing the Standard Operating Procedures	95
Configuring Standard Operating Procedures for different activities	97
Defining a Standard Operating Procedure	102
Creating a reference for Standard Operating Procedures	103
Editing a Standard Operating Procedure	104
Submitting a draft Standard Operating Procedure for approval	104
Testing a Standard Operating Procedure	105
Exporting a Standard Operating Procedure	105
Importing a Standard Operating Procedure	105
Reverting to a particular version of a Standard Operating Procedure	106
Viewing a Standard Operating Procedure	106
Extend situation awareness	107
Configuration of the front end development	108

Model driven back end implementation	109
Extending the application	110
Customizing the solution using the UI and Service Framework	119
Customizing the user interface	119
Configuring notifications	122
Configuring the application	122
UI Framework	123
The REST service framework	167
The Sample Application	168

Chapter 8. Predictive models. 171

The Maintenance predictive model	172
Data understanding	172
Pre-modeling the data	173
Modeling the data	173
Post modeling data manipulation	174
Evaluation of the model	175
Deployment of the model	176
Recommendations from ADM	176
The Sensor Health predictive model	177
Data understanding	177
Data preparation	178
Data modeling	180
Evaluation of the model	181
Deployment	181
Recommendations	182
The Top Failure Reason predictive model	183
Understanding the data	183
Preparing the data	183
Modeling the data	184
Evaluation	184
Deployment	185
The Feature-based predictive model	185
Input data for training	188
Minimum data requirements	188
Resource sub type level modeling	189
Training job	189
Data preparation	190
Data modeling	190
Deployment	192
Recommendations	193
The Integrated predictive model	193
Input data for training	194
Minimum data requirements	194
Resource sub type level modeling	194
Training job	195
Data preparation	195
Orchestration rules	196
Predictive modeling	196
Deployment	198
Recommendations	199

Chapter 9. Recommendations 201

Preventing scoring for incoming events	202
Disabling work order creation	202

Chapter 10. Reports and dashboards 203

Site Overview Dashboard	204
Top 10 Contributors dashboard	206

KPI trending report	207
Actual vs plan report	207
Equipment listing report	207
Outliers report	208
Recommended actions report	209
Equipment dashboard	209
Equipment profile report	209
Equipment control chart	210
Equipment run chart	211
Equipment outliers	211
Event type history report	212
Product quality dashboard	212
Defect analysis dashboard	212
Inspection rate analysis	214
Material usage by process crosstab	215
Audit Report	215
Material usage by production batch	216
Maintenance Overview Dashboard	216
Statistical process control reports	219
SPC - Histogram	219
SPC - X Bar R/S Chart	220
Advanced KPI Trend Chart	220
QEWS quality dashboards	221
Quality dashboard - Inspection	221
Quality dashboard - Inspection Detail History	222
QEWS - Inspection Chart	222
Quality dashboard - Warranty	223
Quality dashboard - Warranty Detail History	224
QEWSL - Warranty Chart	224
Quality dashboard - Parametric	225
Quality dashboard - Parametric Detail History	226
QEWSV - Parametric chart	226
TopN Failure Analysis Report	227

Chapter 11. Implement a data lake by using the Hadoop Distributed File System 229

Use IBM SPSS Modeler and IBM Analytics Server	229
Setting up a connection between IBM SPSS Modeler and IBM Analytics Server	229
Creating input and output folders in HDFS	230
Creating an SPSS Modeler stream to load data into the data lake	230
Checking the result of the data load	231
Configuring data sources in SPSS Analytics Server	231
Creating an SPSS Modeler stream to profile data	232
Checking the result of the data profiling	232
Use the HDFS command line and Apache Spark	233
Uploading raw data by using the HDFS command line	233
Use Apache Spark for data profiling	233
Submitting an Apache Spark application	234

Chapter 12. Implement a data lake by using IBM DB2. 237

Creating an SPSS Modeler stream to load data into the data lake	237
---	-----

Appendix A. Accessibility features 239

Appendix B. The flat file API 241

Master data in the API 241

- batch_batch 242
- event_code 243
- group_dim 244
- language 244
- location 245
- material 246
- material_type 247
- process 247
- product 248
- production_batch 249
- profile_calculation 249
- resource** 250
- resource_type 252
- source_system 252
- supplier 253
- tenant 254
- Changing the tenant code and name. 254
- value_type 255

Metadata in the API 255

- event_type** 255
- measurement_type** 256
- profile_variable** 257
- Mandatory profile variables and measurement types 258
- Remove master data 260

Appendix C. Data access API 263

Basic query 263

Generic query 264

Generic ingestion for KPI table 265

Appendix D. IBM Cognos Framework Manager model description 267

IBM Cognos Framework Manager model database layer 267

IBM Cognos Framework Manager model logical layer 282

IBM Cognos Framework Manager model dimensional layer 283

IBM Cognos Framework Manager model security 283

Query mode. 283

- Using Compatible Query Mode to see real-time data 283

Appendix E. IBM Predictive Maintenance and Quality Artifacts . . . 285

Data model 285

IBM InfoSphere Master Data Management Collaboration Server file. 285

IBM Integration Bus and ESB artifacts 285

Sample master data, event data, and QEWS data files 287

IBM SPSS artifacts 288

IBM Cognos Business Intelligence Artifacts . . . 294

Appendix F. Troubleshooting. 301

Troubleshooting resources 301

- Support Portal 301
- Service requests 302
- Fix Central 302
- Knowledge bases 302

Log files 303

Unresolved library error on import of PMQDancingCharts or PMQMasterDDLGenerator . 304

SPSS job invocation from the orchestration message flow fails 305

Performance tuning guidelines 305

- Deadlock errors happen when parallel processing is enabled. 305
- Event processing performance. 306

Troubleshooting reports 307

- Audit Report fails with error DMB-ECB-0088 A
- DMB cube build limit has been exceeded . . . 307

Notices 309

Trademarks 311

Terms and conditions for product documentation 311

IBM Online Privacy Statement. 312

Index 313

Introduction

The IBM® Predictive Maintenance and Quality solution uses data from multiple sources to give you the information to make informed operational, maintenance, or repair decisions.

IBM Predictive Maintenance and Quality provides you with operational intelligence data, which enables you to perform the following tasks:

- Understand, monitor, predict, and control product and process variability.
- Perform in-depth root cause failure analysis.
- Identify incorrect operating practices.
- Enhance equipment and process diagnostics capabilities.

It also provides you with asset performance management capabilities that help you to achieve these goals:

- Have forward visibility into equipment and process performance.
- Increase asset uptime.
- Identify safety issues.
- Identify improper maintenance procedures.
- Optimize maintenance intervals and procedures.

Audience

This information is intended to provide users with an understanding of how the IBM Predictive Maintenance and Quality solution works. It is designed to help people who are planning to implement IBM Predictive Maintenance and Quality know what tasks are involved.

Finding information

To find documentation on the web, including all translated documentation, access IBM Knowledge Center (<http://www.ibm.com/support/knowledgecenter>).

Accessibility features

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products. Some of the components included in the IBM Predictive Maintenance and Quality solution have accessibility features. For more information, see “Accessibility features” on page 2.

IBM Predictive Maintenance and Quality HTML documentation has accessibility features. PDF documents are supplemental and, as such, include no added accessibility features.

Forward-looking statements

This documentation describes the current functionality of the product. References to items that are not currently available may be included. No implication of any future availability should be inferred. Any such references are not a commitment, promise, or legal obligation to deliver any material, code, or functionality. The

development, release, and timing of features or functionality remain at the sole discretion of IBM.

Chapter 1. Introduction

The IBM Predictive Maintenance and Quality solution uses data from multiple sources to give you the information to make informed operational, maintenance, or repair decisions.

IBM Predictive Maintenance and Quality provides you with operational intelligence data, which enables you to perform the following tasks:

- Understand, monitor, predict, and control product and process variability.
- Perform in-depth root cause failure analysis.
- Identify incorrect operating practices.
- Enhance equipment and process diagnostics capabilities.

It also provides you with asset performance management capabilities that help you to achieve these goals:

- Have forward visibility into equipment and process performance.
- Increase asset uptime.
- Identify safety issues.
- Identify improper maintenance procedures.
- Optimize maintenance intervals and procedures.

Audience

This information is intended to provide users with an understanding of how the IBM Predictive Maintenance and Quality solution works. It is designed to help people who are planning to implement IBM Predictive Maintenance and Quality know what tasks are involved.

Finding information

To find documentation on the web, including all translated documentation, access IBM Knowledge Center (<http://www.ibm.com/support/knowledgecenter>).

Accessibility features

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products. Some of the components included in the IBM Predictive Maintenance and Quality solution have accessibility features. For more information, see “Accessibility features” on page 2.

IBM Predictive Maintenance and Quality HTML documentation has accessibility features. PDF documents are supplemental and, as such, include no added accessibility features.

Forward-looking statements

This documentation describes the current functionality of the product. References to items that are not currently available may be included. No implication of any future availability should be inferred. Any such references are not a commitment, promise, or legal obligation to deliver any material, code, or functionality. The

development, release, and timing of features or functionality remain at the sole discretion of IBM.

What's new in version 2.6.1

The following new features are available in IBM Predictive Maintenance and Quality 2.6.1.

SPSS

SPSS was updated to version 18.1.

Oracle Predictive Maintenance and Quality 2.6.1 now supports Oracle.

Other updates

Predictive Maintenance and Quality 2.6.1 includes recent supporting program updates and fixes available at the time of release.

Accessibility features

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products.

For information about the commitment that IBM has to accessibility, see the IBM Accessibility Center (www.ibm.com/able).

IBM Cognos HTML documentation has accessibility features. PDF documents are supplemental and, as such, include no added accessibility features.

Report output

In IBM Cognos Administration, you can enable system-wide settings to create accessible report output. For more information, see the *IBM Cognos Business Intelligence Administration and Security Guide*. In IBM Cognos Report Studio, you can enable settings to create accessible output for individual reports. For more information, see the *IBM Cognos Report Studio User Guide*. You can access the previously mentioned documents at IBM Knowledge Center (<http://www.ibm.com/support/knowledgecenter>).

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year).

Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. _enter the year or years_.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software

Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's name, user name, password, or other personally identifiable information for purposes of session management, authentication, single sign-on configuration or other usage tracking or functional purposes. These cookies can be disabled, but disabling them will also likely eliminate the functionality they enable.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at <http://www.ibm.com/privacy> and IBM's Online Privacy Statement at <http://www.ibm.com/privacy/details> in the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at <http://www.ibm.com/software/info/product-privacy>.

Chapter 2. Configuring the product

You must configure IBM Predictive Maintenance and Quality before the application can be deployed to users.

The following tasks are necessary to configure the product:

- Identify the assets, resource types, their event types, and measurements.
- Load the master data. Master data supplies IBM Predictive Maintenance and Quality with information about the context in which events occur, for example, location of a resource or event, definition of a material or production process.
- Load the event data. The event data is data that you want to measure about an event. Data comes from many sources, and it must be transformed into a format that can be used by the product.
- Configure the event types, measurement types, and profile variables. Set up the types of measurement that must be done, and the key performance indicators (KPIs) that must be calculated from these measurements. Profiles are a condensed history of resources that help to speed up scoring.
- Configure the predictive models. Run the historical data through the modeler to determine which values are needed. You can then refine the model so that it gives you accurate predictions and generates scores.
- Define rules that determine what actions happen when a score threshold is breached.
- Configure the reports and dashboards that the user sees. Reports and dashboards can be customized, and new ones can be created.

Identification of the assets, resource types, event types, and measurement types

Before you deploy an IBM Predictive Maintenance and Quality application, identify the assets and the information that you want to monitor.

To establish what data is required, and what preparation must be done, ask the following questions.

- Which assets must be monitored, and why?
- What events do you want to monitor for those assets?
- What measurements do you want to capture for the events?

Resource types

The two supported resource types are asset and agent. An asset is a piece of equipment that is used in the production process. An agent is the operator of the equipment. When you define resources, you can use the resource subtype field to identify specific groups of assets or agents.

The following table shows some sample event types in the data model.

Table 1. Sample event types in the data model

Event type code	Event type name
ALARM	Alarm

Table 1. Sample event types in the data model (continued)

Event type code	Event type name
WARNING	Warning
SYSTEM CHECK	System Check
MEASUREMENT	Measurement
RECOMMENDED	Recommended Actions
FAILURE	Failure
REPAIR	Repair

The following table shows some sample measurement types in the data model.

Table 2. Sample measurement types in the data model

Measurement type code	Measurement type name
RECOMMENDED	Recommend Action
RPM	RPM
FAIL	Incident Count
INSP	Inspection Count
LUBE	Lube Count
OPHR	Operating Hours
PRS1	Pressure 1
PRS2	Pressure 2
PRS3	Pressure 3
R_B1	Replace Ball Bearing Count
R_F1	Replace Filter Count
RELH	Relative Humidity
REPT	Repair Time
REPX	Repair Text
TEMP	Ambient Temperature
Z_AC	High Temperature/ Humidity Warning Count
Z_FF	Latent Defect
Z_PF	Probability of Failure
Z_TH	High Temperature/ Humidity Count
OPRI	Operating Hours at Inception
REPC	Repair Count
MTBF	MTBF
MTTR	MTTR
OPRD	Operating Hours Delta

Create a custom application

You can create a custom IBM Predictive Maintenance and Quality application by creating custom IBM Integration Bus flows, IBM Cognos® Business Intelligence reports and dashboards, or predictive models.

The following list describe the high-level tasks you can take to create a custom application.

- Customize or create new predictive models by using IBM SPSS® Modeler.
- Create new business rules by using IBM Analytical Decision Management.
- Create new flows that interface with external systems by using IBM Integration Bus.
- Customize scoring during event processing by using IBM Integration Bus.
- Customize or create the message flows to orchestrate the activities by using IBM Integration Bus.
- Customize or create new reports by using IBM Cognos Report Studio.
- Modify the metadata for the reports by using IBM Cognos Framework Manager.

Sample files, model files, and other content is supplied to help you to configure IBM Predictive Maintenance and Quality for the needs of your business. For more information, see Appendix E, “IBM Predictive Maintenance and Quality Artifacts,” on page 285.

Integration with asset management and manufacturing execution systems

Asset management and manufacturing execution systems are an important source of master data and event data. You can feed recommendations and forecasts produced by IBM Predictive Maintenance and Quality into these systems to close the loop and perform action.

Predictive Maintenance and Quality can create work orders in IBM Maximo® Asset Management based on recommendations from predictive scoring and decision management. Predictive Maintenance and Quality contains the APIs for integration with these systems and technology for building connectors to the systems. Predictive Maintenance and Quality includes a prebuilt adapter for integration with Maximo.

IBM Maximo is not installed as part of IBM Predictive Maintenance and Quality. If required, it must be purchased separately. However, IBM Predictive Maintenance and Quality contains adapters for IBM Maximo, which allow data integration.

IBM Predictive Maintenance and Quality dashboard application

The Predictive Maintenance and Quality dashboard application provides a single interface where users can access IBM Cognos Business Intelligence reports, access the IBM Analytics Solutions Foundation, and view live profile values for a particular resource.

Users access the Predictive Maintenance and Quality dashboard application by typing the following URL.

```
http://bi_node_name:port_number/PMQDashboard/pmqui
```

bi_node_name is the name or IP address for the BI node computer. Consult your system administrator to obtain the BI node computer name and port number.

IBM Cognos Business Intelligence reports

The **Reports** link in the dashboard application banner gives users access to IBM Cognos BI content. The link opens IBM Cognos Connection, and users can view available reports.

IBM Analytics Solutions Foundation

The **Foundation** link in the dashboard application banner gives users that belong to the Administrator group access to the IBM Analytics Solutions Foundation interface. In this interface, users can author orchestration and solution definition XML files.

Note: The IBM Analytics Solutions Foundation Developer Guide is available from the customer support portal. To access it, see the IBM Analytics Solutions Foundation Developer Guide (<http://www.ibm.com/support/docview.wss?uid=swg27045826>).

Dashboards

The **Dashboard** link in the dashboard application banner gives users the ability to create, save, and delete dashboards that contain one or more charts. Users configure each chart to display live data for a profile of a selected resource.

Related concepts:

Chapter 10, “Reports and dashboards,” on page 203

You can customize and extend the reports and dashboards that are supplied with IBM Predictive Maintenance and Quality. You can also design your own reports and dashboards and add them to the menu.








Creating a dashboard

Create a dashboard to show live data in a chart for a selected profile of a resource.

About this task

The following table describes the tasks that you can perform when you create a dashboard.

Table 3. Dashboard tasks

Icon	Task
	Save dashboard
	Edit dashboard
	Add chart
	Configure chart
	Delete chart
	Move chart up or down
	Delete dashboard

There is no limit to the number of dashboards that you can create, and there is no limit to the number of charts that a dashboard can contain.


When you work in a dashboard, two modes are available. A dashboard enters the view mode when you save the dashboard. From the view mode, a dashboard enters the edit mode when you click the **Edit dashboard** icon.

When you configure a chart, the chart can display data for only one profile of a selected resource. When you select a resource, you are shown a list of corresponding profiles to choose from. Resources are categorized under Location and Resource sub types to help you search for the resource that you want. You choose a profile by subscribing to it.

After you subscribe to a profile, if you receive a subscription failed message, ask your administrator to check the following possibilities for the failure.

- The Predictive Maintenance and Quality dashboard WebSphere Application Server cannot connect to the IBM Integration Bus (IIB).
- The IIB server is offline.
- The IIB server is incorrectly configured in the application server.

After you configure a chart, live data for the selected profile of a resource is displayed on the chart. At the most, 15 points are displayed on maps. Chart values are displayed by using the first in, first out concept. New values are pushed at the back, and old values are removed from the front.

Tip: If you receive a message that states that no data is available for the selected resource and profile, it might be because there is no data reported for that resource and profile. Otherwise, click the Refresh icon  to try retrieving the data again.

Procedure

1. Log in to the Predictive Maintenance and Quality dashboard application.
2. In the navigation pane, click the **Create a new dashboard** link, and in the work area, in the **Dashboard Name** box, type the name of the dashboard that you want to create.
3. Add a chart to the dashboard.
4. Configure the chart.
 - a. In the Chart Configuration window, select a resource.
 - b. For the selected resource, click the **Profiles** menu and choose a profile.
 - c. Click **Subscribe**.
5. To add more charts to the dashboard, repeat steps 3 and 4.
6. Save the dashboard.

Chapter 3. Orchestration

Orchestration is the process that ties activities in IBM Predictive Maintenance and Quality together.

Message flows

Orchestration is achieved with message flows in IBM Integration Bus.

The following activities can be tied together:

- Acquiring and storing data
- Aggregating data
- Running predictive models
- Feeding data back to external systems or starting external processes

Message flows are supplied with IBM Predictive Maintenance and Quality and must be customized with IBM Integration Bus. The message flows are organized into the following applications:

- PMQEventLoad
- PMQMasterDataLoad
- PMQMaximoOutboundIntegration
- PMQMaintenance
- PMQModelTraining
- PMQQEWSInspection
- PMQQEWSIntegration
- PMQQEWSWarranty
- PMQTopNFailure

For more information about developing message flows, see IBM Integration Bus Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSMKHH_9.0.0/com.ibm.etools.mft.doc/bi12005_.htm).

By default, IBM Integration Bus is installed in advanced mode. Advanced mode is the correct mode to use for full functionality.

The following examples describe how orchestration is used in IBM Predictive Maintenance and Quality.

Orchestration example: Load real-time event data

This orchestration example is similar to the message flow used to load batch event data.

1. Incoming equipment measurement data is provided through real-time connectivity.
2. A map must be defined in IBM Integration Bus to describe the transformation of incoming data into the IBM Predictive Maintenance and Quality event structure.
3. Incoming business keys are converted to internal integer surrogate keys.

4. Event data is written to the datastore.
5. Event data is aggregated. Profile and key performance indicator (KPI) data is written to the datastore.

Orchestration example: Load batch event data

The following steps take place when batch event data is loaded into IBM Predictive Maintenance and Quality.

1. Incoming measurement data is loaded from a file.
2. The file system is polled for new incoming data.
3. A map that is defined in IBM Integration Bus describes the transformation of incoming data into the IBM Predictive Maintenance and Quality structure.
4. Incoming business keys are converted to internal integer surrogate keys.
5. Event data is written to the datastore.
6. Event data is aggregated. Profile and key performance indicator (KPI) data is written to the datastore.

Orchestration example: Score event data

The following steps take place when event data is scored.

1. New input triggers scoring. For example, to recalculate the health score if a new measurement is reported, then that measurement is processed and the health score is recalculated.
2. A map that is defined in IBM Integration Bus describes the transformation of the data into the model structure.
3. The predictive model is invoked through a web services interface.
4. A map that is defined in IBM Integration Bus describes the transformation of model outputs to the event structure.
5. Model outputs are written as new events.
6. As with external events, model output events can be aggregated and stored on the profile and as KPIs.

For more information about scoring predictive models, and the triggers to score models, see “Predictive scoring” on page 59.

Orchestration example: Apply business rules to data

The following steps take place when business rules are applied.

1. New input triggers evaluation of business rules.
2. A map that is defined in IBM Integration Bus describes the transformation of the data into the model structure.
3. IBM Analytical Decision Management Model is invoked through a web services interface.
4. A map that is defined in IBM Integration Bus describes transformation of model outputs to the event structure.
5. Model outputs are written as new events.
6. As with external events, model output events can be aggregated and stored on the profile and as KPIs.

Orchestration example: Write-back of data

The following steps take place when write-back of data to an external process occurs.

1. Creation of an event triggers the requirement to start an external process.
2. A map that is defined in IBM Integration Bus describes the transformation of the data to the structure of an external web service.
3. The external web service is called.

Example of an orchestration XML file

An example file, `inspection.xml`, demonstrates the purpose and structure of an orchestration file.

Each orchestration flow can be defined in a separate XML file. The file defines the behavior of the orchestration steps. A mapping determines the orchestrations to be performed for an event with an event orchestration key code.

In this example scenario, there are two kinds of events: production and inspection. Therefore, there are two event orchestration key codes, one for each type of event.

The example file “`inspection.xml`” on page 16 determines the orchestration for an inspection event.

Description

The first part of the file `inspection.xml` lists the event type, the adapter class, and configuration that is required for the particular class of adapter:

- `<event_orchestration_mapping>`
The type of event is defined as an inspection.
- `<adapter_class>`
The adapter class that will be executed, in this case `ProfileAdapter`, is called in the step.
- `<adapter_configuration>`
The profile adapter requires configuration to determine how observations with a specific measurement type will update specific profile tables.

The remainder of the file specifies how two specific profiles will be updated, depending on whether the measurement type has a value of `INSPECT` or `FAIL`:

- `<observation_profile_update>`
If the measurement type has a value of `INSPECT`
`<profile_update_action>` The `PRODUCT_KPI` table is updated with the shared calculation of `Product_KPI_Inspect_count`. This calculation produces the value for the number of days when an inspection has taken place.
- `<observation_profile_update>`
If the measurement type has a value of `FAIL`
`<profile_update_action>` The `PRODUCT_KPI` table is updated with the shared calculation of `PRODUCT_KPI_FAIL_COUNT`. This calculation produces the value for the number of times an asset has failed.

inspection.xml

The file inspection.xml contains the following code:

```
<event_orchestration_mapping>
  <event_orchestration_key_cd>inspection</event_orchestration_key_cd>
  <orchestration_cd>pmq.inspection</orchestration_cd>
</event_orchestration_mapping>

<orchestration>
  <orchestration_cd>pmq.inspection</orchestration_cd>
  <step>
    <adapter_class>com.ibm.analytics.foundation.adapter.profile.ProfileAdapter</adapter_class>
    <adapter_configuration xsi:type="ns3:profile_adapter_configuration">
      <observation_profile_update>
        <observation_selector table_cd="EVENT_OBSERVATION">
          <observation_field_value>
            <field_name>MEASUREMENT_TYPE_CD</field_name>
          </observation_field_value>
          <value>INSPECT</value>
        </observation_selector>

        <profile_update_action>
          <profile_row_selector>
            <shared_selector_cd>PRODUCT_KPI</shared_selector_cd>
          </profile_row_selector>
          <shared_calculation_invocation_group_cd>PRODUCT_KPI_INSPECT_COUNT
          </shared_calculation_invocation_group_cd>
        </profile_update_action>
      </observation_profile_update>

      <observation_profile_update>
        <observation_selector table_cd="EVENT_OBSERVATION">
          <observation_field_value>
            <field_name>MEASUREMENT_TYPE_CD</field_name>
          </observation_field_value>
          <value>FAIL</value>
        </observation_selector>
        <profile_update_action>
          <profile_row_selector>
            <shared_selector_cd>PRODUCT_KPI</shared_selector_cd>
          </profile_row_selector>
          <shared_calculation_invocation_group_cd>
PRODUCT_KPI_FAIL_COUNT</shared_calculation_invocation_group_cd>
          </profile_update_action>
        </observation_profile_update>
      </adapter_configuration>
    </step>
  </orchestration>
```

Generic batch orchestration

Generic batch orchestration provides capabilities to run a scheduler flow and invoke any IBM SPSS batch job by taking inputs from a configurable XML file instead of developing separate message flows for a specific use case.

Generic batch orchestration also provides the flexibility of changing the scheduled time of a flow and web service input parameters through an XML file, without changing message flows.

In generic batch orchestration, an orchestration XML file is used to store all configurations. Message flows read the XML file at run time from the properties folder on the Enterprise Service Bus (ESB) node.

The following capabilities are available in generic batch orchestration.

AutoTrigger

AutoTrigger is used to automatically trigger scheduler flows. The AutoTrigger flow creates a TimeoutRequest message according to the scheduler configurations in the XML file, and places the message on the queue that is specified in the scheduler configuration. AutoTrigger can accept any changes that are made to scheduler configurations.

The queue to which the AutoTrigger flow places the TimeoutRequest message can either be SPSSJobIntegration.msgflow or any other custom flow that is configured to run at a scheduled time. To run at the scheduled time, SPSSJobIntegration.msgflow or a custom flow must contain an MQInput node, TimeoutControl node, and a TimeoutNotification node that is configured to match the Identifier and queue name configurations in the XML file.

The following figure shows the parameters that are required for a scheduler configuration.



The image shows a configuration tree for a scheduler configuration. The tree is expanded to show the 'scheduler' node, which contains three sub-nodes: 'scheduled_time', 'queue_name', and 'duration_in_days'. The values for these nodes are 00:00:00, PMQ.QEWS.PTIMER.IN, and 1, respectively.

batch	Orchestration for Parametric
orchestration	Parametric
Identifier	Parametric
scheduler	
scheduled_time	00:00:00
queue_name	PMQ.QEWS.PTIMER.IN
duration_in_days	1

Figure 1. Required parameters for a scheduler configuration

The first trigger of the flow occurs on the date of the flow deployment or flow restart at the time specified (`<scheduled_time></scheduled_time>`) in the batch orchestration XML file. The trigger repeats at regular intervals, in days (`<duration_in_days></duration_in_days>`), as specified in the batch orchestration XML file.

If a change is made to **duration_in_days**, the change takes effect from the second run that was previously configured. For example, **duration_in_days** is set to 3, and with this value, the next time the flow is supposed to run is 2014-12-09. If **duration_in_days** is changed to 2, the change takes effect only after 2014-12-09. If you want the change to immediately take effect, you must restart the flow.

The following figure shows an example of the Timer flow.

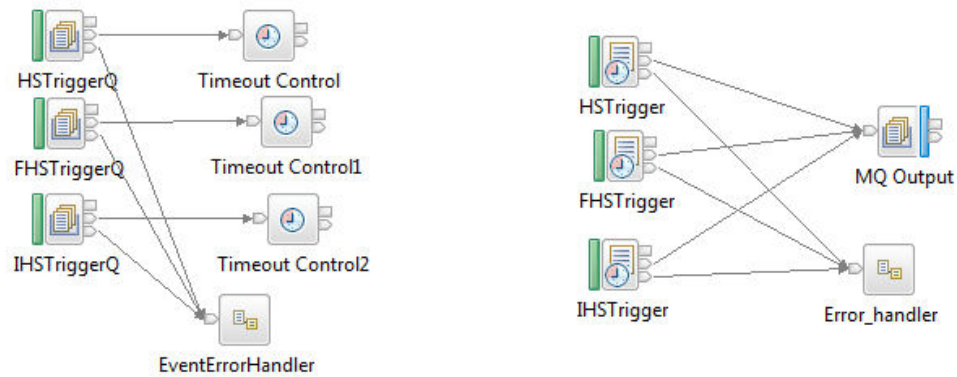


Figure 2. Timer flow example

SPSSJobIntegration

The SPSSJobIntegration flow is used to invoke the IBM SPSS SubmitJobWithOptions web service to trigger an SPSS job with parameters. The SPSSJobIntegration flow picks up parameters, the web service end-point URL, the SPSS JobLocation URL, and **notificationEnabled** field values from the XML file.

Parameters that are used in the web service can either be static parameters, which are values that are predefined in the XML file, or dynamic parameters, which are values that are expected to come from a data preparation flow. The data preparation flow is specific to a use case, or is a combination of both static and dynamic parameters. There can also be cases where there are no parameters that are required.

The **type** field in the XML file specifies whether the parameter is static or dynamic. The parameter name is configured in the **name** field. For static parameters, the parameter value is specified in the **value** field. For dynamic parameters, the value is taken from the input message in the data preparation flow, which is specific to a use case. In this case, the **value** field in the XML file contains the input message field name, from which the dynamic value is mapped.

For dynamic field mapping, the request from the data preparation flow must have **Request** as the parent XML tag name, and child elements must contain the parameters. This request must also contain **Identifier** as one of the child elements, which is used by SPSSJobIntegration to identify which set of parameters to use for the particular use case.

The following code is a sample request from a data preparation flow. StartDate is one of the dynamic parameter values.

```
<Request>
<Identifier>WTIMER</Identifier>
<StartDate>2014-02-02</StartDate>
</Request>
```

The following figure shows the message flow that SPSSJobIntegration uses.

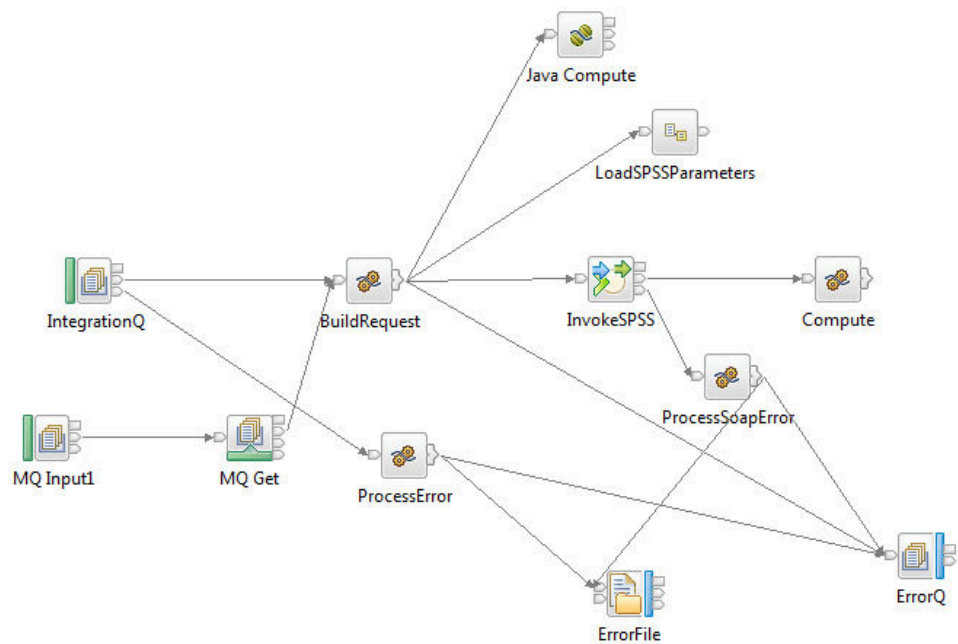


Figure 3. SPSSJobIntegration message flow

The following figure shows the parameters that the web service configuration requires.

webservice	
url	http://9.122.121.208:9080/process/services/ProcessManagement
jobLocationURI	spsscr:///id=569103e53065d83300000144f8d1202dbe9e
parameters	
parameter	
name	RunDateInFormatYYYYMMDDHyphenSeparated
value	StartDate
type	dynamic
parameter	
name	ServiceTablQtyMultiplie
value	1
type	static
parameter	
name	IsRunDateEqServerDate
value	0
type	static
notificationEnabled	true

Figure 4. Required parameters for the web service configuration

To trigger SPSSJobIntegration, a data preparation flow or Timer flow node that is specific to the use case must be created. The remaining parameters and fields that are required to invoke the SPSS batch job are taken from the orchestration XML file.

If there are only static parameters that are required for an SPSS job invocation, AutoTrigger and SPSSJobIntegration flows can be used together as a combination. In scenarios where dynamic parameters are involved, an extra data preparation flow that prepares the data, as required by the use case, is also needed.

If a new feature orchestration is introduced in the orchestration XML file, the PMQBatchIntegration flow must be restarted.

Tip: If you want to change configurable properties in the XML file, edit the changes in a local copy of the file, and replace the run time XML file with the modified file to avoid problems that are due to file locking.

The following table describes the features that use batch orchestration to run a scheduler flow, an SPSS job, an external flow, or a combination of flows and jobs, based on the scheduler and web service configuration in the batch orchestration XML file. Depending on your needs, configurations for flows and jobs can be updated at run time in the XML file.

Table 4. Features that use batch orchestration

Feature name	Purpose
Parametric	Triggers the Parametric adapter invocation at the scheduled time once a day, and modifies the default Parametric SubUseCase name.
Inspection	Triggers the Inspection adapter invocation at the scheduled time once a day.
Warranty	Triggers the Warranty SPSS job at the scheduled time once a day through the web service.
HS training	Triggers the HS Training SPSS job at the scheduled time once every 90 days through the web service.
FBA training	Triggers the Feature-based Analytics training SPSS job at the scheduled time once every 90 days through the web service.
IHS training	Triggers the Integrated Health Score training SPSS job at the scheduled time once every 90 days through the web service.
TopNFailure	Triggers the TopNFailure feature at the scheduled time once a day, and invokes the SPSS job for TopNFailure Analytics and Event generation through the web service.
Maintenance	Triggers the Maintenance feature at the scheduled time once a day, and invokes the SPSS job for Maintenance Analytics and Event generation through the web service.
Distribution transformer current aging	Triggers the Distribution Transformer Current Aging SPSS job at the scheduled time once a day through the web service.
Distribution transformer projected aging	Triggers the Distribution Transformer Projected Aging SPSS job at the scheduled time once every 180 days through the web service.
Pole FBA	Triggers the Pole FBA SPSS job at the scheduled time once every 30 days through the web service.

Chapter 4. Master data

Master data is the type of resource that you want to manage, for example, definition of a material or production process.

Master data can come from manufacturing engineering systems (MES) such as IBM Maximo, or other existing data sources. IBM InfoSphere® Master Data Management Collaboration Server can be used to complete gaps in the data from these sources or consolidate data from multiple sources. You can also add attributes, create relationships between items, or define data that you do not have another source for. For example, add hierarchy information to indicate which pieces of equipment belong to which site, in which location or classify resources into groups. In a report, the hierarchies and groups can be displayed as additional information or used as drill downs and filters.

Master data is normally loaded by one of the supplied connectors or the Flat File API. The connectors and the Flat File API use IBM Integration Bus flows to transform the data into the form that is required and update the data in the IBM Predictive Maintenance and Quality database.

Master data process

When a file is placed in the file input directory, IBM Integration Bus reads and processes it and then removes it from the directory. IBM Integration Bus stores and retrieves data from the database as required.

The response file indicates whether the operation was successful, and lists any results. If errors occur, a log file is written to the error directory.

The following diagram shows the flow of a file request and its response.

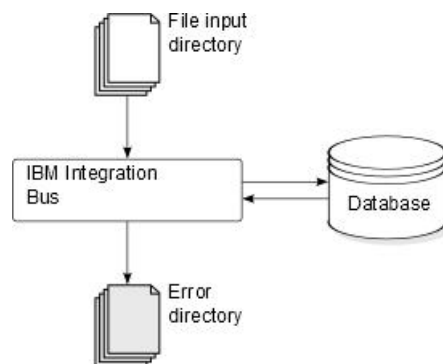


Figure 5. Master data process

Data organization

IBM Predictive Maintenance and Quality processes the following kinds of data:

- Master data supplies IBM Predictive Maintenance and Quality with information about the context in which events occur. Master data includes descriptions of the devices that produce events, the location where events occur, and the material that is used in an event.

- Metadata defines how IBM Predictive Maintenance and Quality processes received events. For more information, see “Metadata in the API” on page 255.
- Event data supplies IBM Predictive Maintenance and Quality with information that you want to measure about an event. For more information, see “How events are processed” on page 49.

The flat file application programming (API) interface

IBM Predictive Maintenance and Quality master data is supplied, accessed, modified, or removed using the flat file API. For more information, see Appendix B, “The flat file API,” on page 241.

File format and location

Master data and event data must be in a format that IBM Predictive Maintenance and Quality can recognize. The default file format is flat file, comma separated (.csv) format. Other file formats can be used, but you must create extra IBM Integration Bus flows.

File location

The file location is determined by the MQSI_FILENODES_ROOT_DIRECTORY environment variable. The file location is configured during the installation process.

This location contains the following sub folders:

- \masterdatain
used for loading master data and metadata files
- \eventdatain
used for loading event data files
- \error
used to report errors that occur while loading data
- \maximointegration
used for loading data files from IBM Maximo
- \control
- \restricted
- \properties

File names

Files must follow this naming convention:

record_name_operation.csv*

For example, a file that contains a set of resource records to be added to IBM Predictive Maintenance and Quality might be named:

resource_upsert_01.csv

File format

The .csv file format is used by default:

- Each line in a file is a record, and contains a sequence of comma-separated values. If a value contains a comma, the value must be contained within double quotation marks ",".
- Each record normally includes a code value (or combination of values) that uniquely identifies the record. These code values are sometimes known as business keys. Because this code value is a unique identifier for a row, it is used in other files as a way to reference that particular row. For example, in a file that contains a list of resources, the row for a resource can contain a location value. The location value is the code that is used to identify a location record.
- Sometimes a code value is required but is not applicable for a particular record. In this case, the special code **-NA-** must be used. For example, to avoid defining a location for a particular resource, use the code **-NA-** for the location value. The code value cannot be changed.
- In addition to a code value, a record typically has a name value. Both code and name value can hold the same value. However, while the code value must be unique for each row and is not normally shown to users, the name is visible in reports and dashboards. The name can be changed, unlike the code value.

The following example shows the format for a `location.csv` file. The command must be on a single line, not as shown here:

```
location_cd,location_name,region_cd,region_name,country_cd,country_name,
state_province_cd,state_province_name,city_name,latitude,longitude,
language_cd,tenant_cd,is_active
RAVENSWOOD,Ravenswood,NORTH AMERICA,North America,USA,United States,
CA,California,Los Angeles,34.0522,118.2428,,
TARRAGONA,Tarragona,EUROPE,Europe,UK,United Kingdom,ENGLAND,England,
London,51.5171,0.1062,,1
```

The following example shows codes that are used to identify records and used to reference other records. The codes that are used to identify a resource record are different from other records because a resource record is identified by both `Resource_CD1` and `Resource_CD2`, or by `operator_cd`.

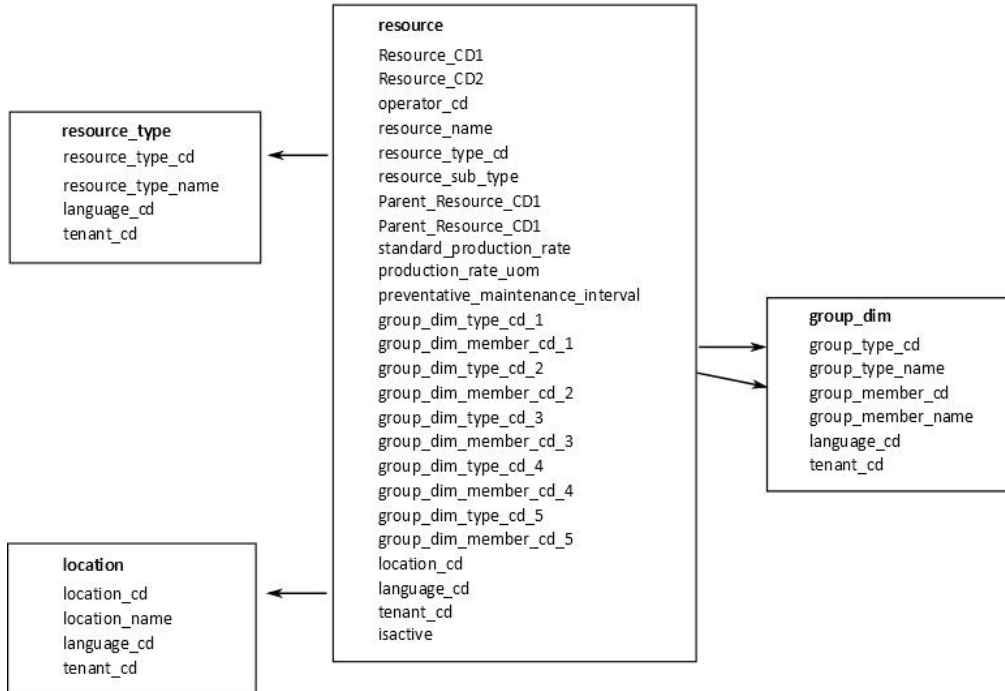


Figure 6. Codes used to identify and to reference records

Modifying a resource or process parent

If you must change a resource or process parent, you must reload the resource or process and all of its children. Modify the parent in a master data .csv file that contains all these rows, and resubmit the file.

Security

Implement security by restricting access to the directories used to supply files for the API.

Master data using InfoSphere MDM Collaboration Server

You can use IBM InfoSphere Master Data Management Collaboration Server to complete gaps in the data from external sources or consolidate data from multiple sources. You can also add attributes, create relationships between items, or define data that you do not have another source for.

For example, add hierarchy information to indicate which pieces of equipment belong to which site, in which location or classify resources into groups. In a report, the hierarchies and groups can be displayed as additional information or used as drill-downs and filters.

IBM InfoSphere Master Data Management Collaboration Server is model driven: you create a specification, and then define the fields. It automatically generates the user interface for the fields, for example, lookup tables, and date pickers. You can embed images in the data such as a picture of an asset.

A model for InfoSphere MDM Collaboration Server is provided with IBM Predictive Maintenance and Quality that simplifies the configuration. To use this model, you must do the following configuration steps.

1. Set the environment variable *PMQ_HOME* to the root of the IBM Predictive Maintenance and Quality installation directory.
2. Create a company for IBM Predictive Maintenance and Quality, see “Creating a company in IBM InfoSphere MDM Collaboration Server” on page 26.
3. Import the metadata (Company deployment), see “Importing metadata into InfoSphere MDM Collaboration Server” on page 29.
4. Configure the InfoSphere MDM Collaboration Server user interface, see “Configuring the IBM InfoSphere MDM Collaboration Server user interface” on page 27.

There are some specific guidelines that you must follow to ensure that you get the results that you expect. See “Guidelines for managing data in IBM InfoSphere MDM Collaboration Server” on page 27.

For additional information about using InfoSphere MDM Collaboration Server, see *Collaborative authoring with InfoSphere MDM Collaboration Server*. This is available from IBM Master Data Management Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSWSR9_11.0.0).

IBM Master Data Management Collaboration Server dynamic references

IBM Master Data Management Collaboration Server tasks use several dynamic references.

The following table describes the variables that are used in the InfoSphere MDM Collaboration Server tasks.

Table 5. Dynamic references

Reference	Description
<i>\$PMQ_HOME</i>	IBM Predictive Maintenance and Quality installation home directory.
<i>mdm_install_dir</i>	The root directory of the InfoSphere MDM Collaboration Server installation. \$TOP is an environment variable that is configured with InfoSphere MDM Collaboration Server by default, which points to this location.
<i>mdm_server_ip</i>	The IP address of InfoSphere MDM Collaboration Server, as seen by other IBM Predictive Maintenance and Quality servers, such as IBM Integration Bus.
<i>pmq_mdm_content_zip</i>	The full path to the content compressed file on the server file system.
<i>mdm_data_export_dir</i>	The directory, mount point, or symbolic link on the InfoSphere MDM Collaboration Server where data exports are configured to be written. The default is <i><\$PMQ_HOME>/data/export/mdm</i> .
<i>wmb_server_ip</i>	The IP address of IBM Integration Bus server, as seen by other IBM Predictive Maintenance and Quality servers.
<i>wmb_fileapi_input_dir</i>	The directory where input data files are to be placed for loading into the IBM Predictive Maintenance and Quality database. The directory can be local or remote. The file location is determined by the MQSI_FILENODES_ROOT_DIRECTORY environment variable. The file location is configured during the installation process.
<i>company_code</i>	The company code for InfoSphere MDM Collaboration Server. Make the code short and easy to remember because it must be entered during each login, for example, IBMPMQ.

Table 5. Dynamic references (continued)

Reference	Description
<i>company_name</i>	The display name of the company in InfoSphere MDM Collaboration Server, for example, IBMPMQ.

Creating a company in IBM InfoSphere MDM Collaboration Server

You must create a company before you can import IBM Predictive Maintenance and Quality metadata into IBM InfoSphere Master Data Management Collaboration Server. A company is similar to the concept of a project.

About this task

For information about the variables used, see “IBM Master Data Management Collaboration Server dynamic references” on page 25.

Procedure

1. Stop the InfoSphere MDM Collaboration Server service.
 - a. Change the directory to `cd <mdm_install_dir>/bin/go` where `<mdm_install_dir>` is the root directory of the InfoSphere MDM Collaboration Server installation.
 - b. Run the **stop_local.sh** command: `./stop_local.sh`
2. Run the company creation script.
 - a. Change the directory to `cd <mdm_install_dir>/bin/db`
 - b. Run the **create_cmp.sh** command: `./create_cmp.sh -code=<company_code> --name=<company_name>`
3. Start the InfoSphere MDM Collaboration Server service.
 - a. Change the directory to `cd <mdm_install_dir>/bin/go`
 - b. Run the **start_local.sh** command: `./start_local.sh`
4. Log in and verify the company. Open your web browser and enter the URL for the InfoSphere MDM Collaboration Server web server, for example: `http://<mdm_host_name>:7507/utills/enterLogin.jsp`
 The following default users are created for the new company:

Table 6. Default roles, users, and passwords created for a new company

Role	User name	Password
Administrator	Admin	trinitron
Basic User	Basic	trinitron

5. Change the default passwords for both the administrator, and for the basic user. You do this in the **Data Model Manager** module > **User Console**.

What to do next

The next step is to import the IBM Predictive Maintenance and Quality metadata into the InfoSphere MDM Collaboration Server.

Configuring the IBM InfoSphere MDM Collaboration Server user interface

Add the IBM Predictive Maintenance and Quality objects in the IBM Master Data Management Collaboration Server navigation area to make it easier to manage data.

Procedure

1. In InfoSphere MDM Collaboration Server, click **Please select a module to add**. A drop-down list is displayed.
2. Select all of the following modules from the **Catalog** module type.
 - **Asset**
 - **Locations**
 - **Material Types**
 - **Processes**
 - **Products**
 - **Suppliers**
3. Select **Groups by Type** from the **Hierarchy** module type.

What to do next

You can customize group types to suit the needs of the project.

1. In the **Groups by Type** hierarchy, choose a group type, and customize it as required with a new code or name.
2. Save the changes.
3. Update the **Group Hierarchy Lookup** by clicking **Product Manager > Lookup Tables, Lookup Table Console**.
4. Update the group type record with the new group type code.

Guidelines for managing data in IBM InfoSphere MDM Collaboration Server

You must follow these guidelines to manage data in IBM InfoSphere Master Data Management Collaboration Server to ensure that you get the results that you expect.

Assets

Define assets in the **Unassigned** category.

You can use the default hierarchy to organize items, but the hierarchy is not used by IBM Predictive Maintenance and Quality.

Group assignments:

- Can be assigned up to five groups from the **Groups by Type** hierarchy.
- Each assignment must be from a different group type.
- Must be assigned to Group (Level 2), not Group Type (Level 1.)

Groups

Groups are managed by using the group hierarchy rather than a catalog. Only categories are defined, not items.

The first level must be group type.

The second level must be groups.

Locations

Define the locations as follows:

- The first level must be **Region** (Location Type=Region).
- The second level must be **Country** (Location Type=Country).
- The third level must be **State** (Location Type=State / Province).

The location items must be defined only under State / Province (only on a leaf node).

Material types, processes, products, and suppliers

Define items in the **Unassigned** category.

You can use the default hierarchy to organize items, but the hierarchy is not used by IBM Predictive Maintenance and Quality.

Configuring and running data exports

To integrate IBM InfoSphere Master Data Management Collaboration Server into IBM Predictive Maintenance and Quality, data export files must be sent to the data input directory for the flat file API on the IBM Integration Bus server.

Before you begin

For information about the variables that are used, see “IBM Master Data Management Collaboration Server dynamic references” on page 25.

About this task

The IBM Integration Bus file location is determined by the `MQSI_FILENODES_ROOT_DIRECTORY` environment variable, and the folder is named `\masterdatain`. The file location is configured during the installation process.

Procedure

1. On the IBM Integration Bus server, ensure that the Network File System (NFS) is configured to run with the following command.

```
/sbin/chkconfig nfs on
```

2. Share the data input directory for the flat file API by adding the following line to `/etc/exports`. Create the directory if it does not exist.

```
<wmb_fileapi_input_dir> <mdm_server_ip>(rw)
```

3. Ensure that sufficient permissions are set on the data input directory.

The following example grants read and write permissions to all users and groups. If you require a more secure configuration, ensure that users, groups, and permissions are consistent with those on the InfoSphere MDM Collaboration Server so that NFS operates correctly.

```
chmod 777 <wmb_fileapi_input_dir>
```

4. Restart the NFS service for the settings to take effect.

```
service nfs restart
```

5. On the InfoSphere MDM Collaboration Server, ensure that the data export directory exists. If it does not, create the directory.

```
mkdir <mdm_data_export_dir>
```

6. Mount the remote flat file API input directory with NFS.

```
mount -t nfs -o rw wmb_server_ip:wmb_fileapi_input_dir mdm_data_export_dir
```

7. Test NFS sharing.

- a. Create a test file on the InfoSphere MDM Collaboration Server.

```
echo <"NFS Test File"> <mdm_data_export_dir>/nfstest.txt
```

- b. Check for the test file on the IBM Integration Bus server:

```
cat <wmb_fileapi_input_dir>/nfstest.txt
```

Results

If the file content is displayed, NFS is working. If you have problems, search for “Red Hat Linux NFS documentation” online for detailed information.

What to do next

To run a data export, in the InfoSphere MDM Collaboration Server Reports Console, select the export and click the **Run** icon. Data export files are written to `$PMQ_HOME/<mdm_data_export_dir>`. The default is `$PMQ_HOME/data/export/mdm`.

Importing metadata into InfoSphere MDM Collaboration Server

You must import IBM Predictive Maintenance and Quality data into IBM Master Data Management Collaboration Server before you can use MDM to manage data.

About this task

For information about the variables that are used, see “IBM Master Data Management Collaboration Server dynamic references” on page 25.

Procedure

Use the following command to import data into InfoSphere MDM Collaboration Server. The command must be on a single line, not as shown here.

```
<mdmce_install_dir>/bin/importCompanyFromZip.sh  
--company_code=<company_code>  
--zipfile_path=IBMPMQ.zip
```

Example

See the following example.

```
$TOP/bin/importCompanyFromZip.sh --company_code=IBMPMQ --zipfile_path  
=$PMQ_HOME/content/IBMPMQ.zip
```

`$TOP` is a built-in IBM InfoSphere Master Data Management Collaboration Server environment variable, which points to the root Master Data Management Collaboration Server directory.

Solution XML file

The solution XML file defines the master data. The master tables and supporting tables are defined so that database tables may be generated and upserts carried out.

The solution XML file defines the following types of tables:

- Master tables
- Event tables
- Profile or KPI tables

The LANGUAGE table and the columns are defined as shown in the following XML code:

```
<table table_cd="LANGUAGE" is_surrogate_primary_key="true"
      validator_class="com.ibm.pmq.master.validators.LanguageValidate">
  <column column_cd="LANGUAGE_CD" type="string" size="50" is_key="true"/>
  <column column_cd="LANGUAGE_NAME" type="string" size="200"/>
  <column column_cd="DEFAULT_IND" type="int"/>
</table>
```

The TENANT table and the columns are defined as shown in the following XML code:

```
<table table_cd="TENANT" is_surrogate_primary_key="true"
      validator_class="com.ibm.pmq.master.validators.TenantValidate">
  <column column_cd="TENANT_CD" type="string" size="100" is_key="true"/>
  <column column_cd="TENANT_NAME" type="string" size="200"/>
  <column column_cd="DEFAULT_IND" type="int"/>
</table>
```

The definitions of the LANGUAGE, TENANT, CALENDAR, EVENT_TIME, and KEYLOOKUP tables must not be modified and must be included in the solution XML file.

Master tables include language and tenant support. They are defined by using attributes of the table. For example, the following definition of the Master_Location table includes the attributes is_multilanguage, is_multitenant, and is_row_deactivateable. The value of "true" indicates that the table is multi-language, multi-tenant, and the table includes a column that indicates whether the row is enabled (active) or disabled (deactivated):

```
<table table_cd="MASTER_LOCATION"
      is_multilanguage="true" is_multitenant="true" is_row_deactivateable="true"
      is_surrogate_primary_key="true"
      validator_class="com.ibm.pmq.master.validators.LocationValidate">
  <column column_cd="LOCATION_CD" is_key="true" size="100"
type="string"/>
  <column column_cd="LOCATION_NAME" is_key="false" size="1024"
type="string"/>
  <column column_cd="REGION_CD" is_key="false" size="50"
type="string" is_nullable="true"/>
  <column column_cd="REGION_NAME" is_key="false" size="200"
type="string" is_nullable="true"/>
  <column column_cd="COUNTRY_CD" is_key="false" size="50"
type="string" is_nullable="true"/>
  <column column_cd="COUNTRY_NAME" is_key="false" size="200"
type="string" is_nullable="true"/>
  <column column_cd="STATE_PROVINCE_CD" is_key="false" size="50"
type="string" is_nullable="true"/>
  <column column_cd="STATE_PROVINCE_NAME" is_key="false" size="200"
type="string" is_nullable="true"/>
  <column column_cd="CITY_NAME" is_key="false" size="200"
type="string" is_nullable="true"/>
```

```

type="string" is_nullable="true"/>
  <column column_cd="LATITUDE" is_key="false" size="10,5"
type="decimal" is_nullable="true"/>
  <column column_cd="LONGITUDE" is_key="false" size="10,5"
type="decimal" is_nullable="true"/>
</table>

```

References

The tables defined in the solution XML file (event, master data, and profile) may also define references to master data tables. For example, Master_Product_Parameters references the Master_Product table. To reference a specific Master_Product row, the flows for Master_Product_Parameters take the business keys Product_Cd and Product_Type_Cd as input parameters in the CSV file. The following definition for Master_Product_Parameters is an example of how to define a reference. Product_Id is an identifier of the reference to the Master_Product table. The business keys of the Master_Product table, Product_type_cd, and Product_cd, along with Tenant_cd, are used to reference a Master_Product row:

```

<table table_cd="MASTER_PRODUCT_PARAMETERS"
  is_multilanguage="true" is_multitenant="true">
  <column column_cd="PARAMETER_NAME" type="string" size="50"
  is_key="true"/>
  <column column_cd="PARAMETER_VALUE" type="double"
  is_key="false"/>
  <reference reference_cd="PRODUCT_ID"
  table_reference="MASTER_PRODUCT" is_key="true"/>
</table>

```

The following example shows a more explicit table definition for Master_Product_Parameters. This method can be used to make the column names different than the business keys. That is, when table_column_cd is different from reference_column_cd. You must use this mapping to have unique reference_column_cd values when there is more than one reference to the same table:

```

<table table_cd="MASTER_PRODUCT_PARAMETERS"
  is_multilanguage="true" is_multitenant="true">
  <column column_cd="PARAMETER_NAME" type="string" size="50"
  is_key="true"/>
  <column column_cd="PARAMETER_VALUE" type="double"
  is_key="false"/>
  <reference reference_cd="PRODUCT_ID"
  table_reference="MASTER_PRODUCT" is_key="true">
  <column_mapping table_column_cd="PRODUCT_CD" reference_column_cd="PRODUCT_CD"/>
  <column_mapping table_column_cd="PRODUCT_TYPE_CD"
  reference_column_cd="PRODUCT_TYPE_CD"/>
  </reference>
</table>

```

Hierarchy table structures

The solution XML file manages the hierarchical structures that are used in IBM Predictive Maintenance and Quality. IBM Predictive Maintenance and Quality maintains hierarchical structures for two Master tables, Resource and Process.

Master_Resource_hierarchy is generated based on the solution XML. The following example shows the definition of Master_Resource in the solution XML file. The self_reference element means that there is a circular reference to the table. The circular reference is required to maintain the hierarchy. The number_of_levels

property defines the number of levels of hierarchy. The `duplicate_column_cd` element refers to the column names that appear across each level of the defined `number_of_levels` property:

```
<self_reference reference_cd="PARENT_RESOURCE_ID" number_of_levels="10">
  <column_mapping table_column_cd="RESOURCE_CD1"
reference_column_cd="PARENT_RESOURCE_CD1" />
  <column_mapping table_column_cd="RESOURCE_CD2"
reference_column_cd="PARENT_RESOURCE_CD2" />
  <duplicate_column_cd>RESOURCE_CD1</duplicate_column_cd>
  <duplicate_column_cd>RESOURCE_CD2</duplicate_column_cd>
  <duplicate_column_cd>RESOURCE_NAME</duplicate_column_cd>
</self_reference>
```

`Master_Process_Hierarchy` is generated based on the solution XML. The following example shows the definition of `Master_Process` in the solution XML file. For `Master_Process_Hierarchy`, hierarchical information for `Process_CD` and `Process_Name` is maintained across five levels:

```
<self_reference
reference_cd="PARENT_PROCESS_ID" number_of_levels="5">
  <column_mapping table_column_cd="PROCESS_CD"
reference_column_cd="PARENT_PROCESS_CD"/>
  <duplicate_column_cd>PROCESS_CD</duplicate_column_cd>
  <duplicate_column_cd>PROCESS_NAME</duplicate_column_cd>
</self_reference>
```

IBM Maximo Asset Management

Master data and event data can be supplied from IBM Maximo to IBM Predictive Maintenance and Quality. Recommended actions that are generated by IBM Predictive Maintenance and Quality can also be passed to IBM Maximo Asset Management.

IBM Maximo Asset Management is not installed as part of IBM Predictive Maintenance and Quality. If required, it must be purchased separately. However, IBM Predictive Maintenance and Quality contains adapters for IBM Maximo, which allow data integration.

How master data is mapped in IBM Maximo Asset Management

As an example, the following tables in IBM Predictive Maintenance and Quality can be populated from the default Maximo object model.

group_dim table

The records in the `group_dim` table provide classifications for resources. You can have up to five classifications for each resource. The classifications might vary.

Table 7. Fields in the group_dim table

Field	Type	Required or optional	Maximo objects/attributes
<code>group_type_cd</code>	string(50)	Required	"MXCLASSIFICATION"
<code>group_type_name</code>	string(200)	Required	"Maximo Classification"
<code>group_member_cd</code>	string(50)	Required	CLASSTRUCTURE.CLASSSTRUCTUREID
<code>group_member_name</code>	string(200)	Required	CLASSTRUCTURE.DESCRPTION

Location table

The location table contains the location of a resource or event, such as a room in a factory or a mine site. In Maximo, this information is stored as a LOCATIONS object and in its associated SERVICEADDRESS object.

Table 8. Fields in the location table

Field	Type	Required or Optional	Maximo objects/attributes
location_cd	string(50)	Required	SERVICEADDRESS.ADDRESSCODE
location_name	string(200)	Required	SERVICEADDRESS.DESCRPTION
region_cd	string(50)	Optional, region_cd and region_name must be supplied together	SERVICEADDRESS.REGIONDISTRICT
region_name	string(200)	Optional	SERVICEADDRESS.REGIONDISTRICT
country_cd	string(50)	Optional, country_cd and country_name must be supplied together	SERVICEADDRESS.COUNTRY
country_name	string(200)	Optional	SERVICEADDRESS.COUNTRY
state_province_cd	string(50)	Optional, country_cd and country_name must be supplied together	SERVICEADDRESS.STATEPROVINCE
state_province_name	string(200)	Optional	SERVICEADDRESS.STATEPROVINCE
city_name	string(200)	Optional	SERVICEADDRESS.CITY
latitude	float (in decimal degrees)	Optional	SERVICEADDRESS.LATITUDE
longitude	float (in decimal degrees)	Optional	SERVICEADDRESS.LONGITUDE

resource table

A resource defines resources of type asset or agent. An asset is a piece of equipment. An agent is the operator of the equipment. Some asset resources might form a hierarchy (for example, a truck is a parent of a tire). Asset information imported from Maximo includes the asset type, classification, and location.

Table 9. Fields in the resource table

Field	Type	Required or Optional	Maximo objects and attributes
Resource_CD1	string(50)	Either serial_no and model or operator_cd are required	ASSET.ASSETNUM
Resource_CD2	string(50)	Not applicable	ASSET.SITEID
resource_name	string(500)	Required	ASSET.DESCRPTION
resource_type_cd	string(50)	Required	Not applicable

Table 9. Fields in the resource table (continued)

Field	Type	Required or Optional	Maximo objects and attributes
resource_sub_type	string(50)	Optional	ASSET.ASSETTYPE
parent_resource_serial_no	string(50)	Optional (parent_resource_serial_no and parent_resource_model should be supplied together)	ASSET.PARENT
parent_resource_model	string(50)	Optional	ASSET.SITEID
parent_resource_operator_cd	string(50)	Optional	Not applicable
standard_production_rate	float	Optional	Not applicable
production_rate_uom	string(40)	Optional	Not applicable
preventative_maintenance_interval	float	Optional	Not applicable
group_dim_type_cd_1	string(50)	Group codes are required but a NA value can be specified for a corresponding type and a member	"MXCLASSIFICATION"
group_dim_member_cd_1	string(50)	Not applicable	ASSET.CLASSSTRUCTUREID
group_dim_type_cd_2	string(50)	Not applicable	Not applicable
group_dim_member_cd_2	string(50)	Not applicable	Not applicable
group_dim_type_cd_3	string(50)	Not applicable	Not applicable
group_dim_member_cd_3	string(50)	Not applicable	Not applicable
group_dim_type_cd_4	string(50)	Not applicable	Not applicable
group_dim_member_cd_4	string(50)	Not applicable	Not applicable
group_dim_type_cd_5	string(50)	Not applicable	Not applicable
group_dim_member_cd_5	string(50)	Not applicable	Not applicable
location_cd	string(50)	Required but a NA code can be specified	ASSET.SADDRESSCODE

Mapping master data in IBM Maximo Asset Management

IBM Predictive Maintenance and Quality includes sample flows that import assets, classifications, and ServiceAddress objects from the default Maximo object model. To enable these flows, master data must be exported out of IBM Maximo as XML files, and is later placed into the \maximointegration folder.

About this task

Asset data that is managed in IBM Maximo is mirrored in IBM Predictive Maintenance and Quality. When data is modified in IBM Maximo, it is automatically updated in IBM Predictive Maintenance and Quality. Data that comes from IBM Maximo must be updated and maintained in IBM Maximo. It is not possible for changes that are made in IBM Predictive Maintenance and Quality to be propagated back to IBM Maximo.

A Maximo Publish Channel is used to export assets, classifications, and the **ServiceAddress** attribute. You must invoke the channel manually initially to populate the IBM Predictive Maintenance and Quality database. After, the channel is automatically triggered whenever the contents of one of these objects changes.

For more information, see IBM Maximo Asset Management Knowledge Center (<http://www.ibm.com/support/knowledgecenter/SSWK4A>).

Procedure

1. Create an object structure in IBM Maximo based on the base object structures available in IBM Maximo Asset Management.

IBM Predictive Maintenance and Quality supports data mapping for three object structures: SPASSET, SPSERVICEADDRESS, and SPCLASSIFICATION.

These object structures are inherited from base object structures in IBM Maximo: ASSET, SERVICEADDRESS, and CLASSSTRUCTURE.

When the object structure is created, use the **Exclude/Include fields** option from the **Select Action** menu to include or exclude fields.

For more information, see *Object structures* in the IBM Maximo Asset Management, *Integrating data with external applications, Integration Components* online documentation.

2. Create the following publish channels:

- SPCLASSIFICATIONCHANNEL_R with object structure SPCLASSIFICATION
- SPPUBLISHCHANNEL_R with object structure SPASSET
- SPSAPUBLISHCHANNEL with object structure SPSERVICEADDRESS

For each publish channel, perform the following action:

- Configure the endpoint to be XML.

For more information, see *Publish channels* in the IBM Maximo Asset Management, *Integrating data with external applications, Integration Components, Channels and services* online documentation.

3. Create an external system and configure the corresponding endpoint for the external system as XML.

The name of the external system must be SPEXTSYSTEM.

Configure the location as the \maximointegration folder. The location of the folder is determined by the MQSI_FILENODES_ROOT_DIRECTORY environment variable.

When IBM Maximo and IBM Integration Bus are installed on different systems, this folder must be shared, or the exported files must be transferred to this folder.

4. Set up publish channels for the external systems.

- a. Name the publish channels as shown:

SPPUBLISHCHANNEL

For Asset.

SPCLASSIFICATIONCHANNEL

For Classification.

SPSAPUBLISHCHANNEL

For ServiceAddress.

- b. Select each publish channel in turn and click **Data Export** to export data.

The export screen supports a filter expression to export a subset of data. For example, if you want to export assets with a specific classification then you must enter a filter expression such as CLASSSTRUCTUREID='1012'.

To find the CLASSSTRUCTUREID that an asset belongs to, go to the **Specifications** tab of ASSET.

The **Specifications** tab contains classification information. The classification has a CLASSSTRUCTUREID associated with it, which you can see when you export the classification.

The exported XML is stored in the \maximo\integration folder.

5. Export the Object Structure schema:
 - a. Search and select the Object Structure for which the schema file must be generated.
 - b. Select **Generate Schema/View XML** action for that object structure. You can select the operation for which schema must be generated. Select the **Publish** operation.

The generated schema is stored in the same location as the data export XML files. These schema files correspond to the SPASSETService.xsd, SPCLASSIFICATIONService.xsd, and SPSERVICEADDRESSService.xsd files in the PMQMaximoIntegration IBM Integration Bus library.

Enabling master data loading in realtime mode

You can enable master data to load in realtime mode by creating publish channels and configuring their endpoints.

Procedure

1. Create new publish channel for real time master data loading.
 - a. Select **Integration > Publish Channels > New**.
 - b. Create the following publish channels:
 - SPCLASSIFICATIONCHANNEL_R, with object structure SPCLASSIFICATION
 - SPPUBLISHCHANNEL_R, with object structure SPASSET
 - SPSAPUBLISHCHANNEL, with object structure SPSERVICEADDRESS
 - c. For each publish channel, select **Action > Enable Event Listeners**, and then select the **Enable Listener** check box.
2. Configure the Web service endpoints.
 - a. Select **GoTo > Integration > Endpoint**.
 - b. Select **New Endpoint** and enter the following information:
 - In the **Endpoint Name** field, type AENDPOINT
 - In the **Handler type** field, type WEBSERVICE
 - In the **EndPointURL** field, type http://ESB_Node_IP_address:7800/meaweb/services/asset
 - In the **ServiceName** field, type asset
 - c. Select **New Endpoint** and enter the following information:
 - In the **Endpoint Name** field, type CENDPOINT
 - In the **Handler type** field, type WEBSERVICE
 - In the **EndPointURL** field, type http://ESB_Node_IP_address:7800/meaweb/services/classification
 - In the **ServiceName** field, type classification
 - d. Select **New Endpoint** and enter the following information:
 - In the **Endpoint Name** field, type SAENDPOINT
 - In the **Handler type** field, type WEBSERVICE
 - In the **EndPointURL** field, type http://ESB_Node_IP_address:7800/meaweb/services/serviceaddress

- In the **ServiceName** field, type serviceaddress
3. Configure the external system to associate the publish channels and endpoints to the external system for webservice event notification of workorders.
 - a. Select **GoTo > Integration > External Systems > filter** for EXTSYS2
 - b. Select **Publish channels > Add New Row**.
 - Enter SPCLASSIFICATIONCHANNEL : CENDPOINT
 - Select the **Enabled** check box.
 - c. Select **Publish channels > Add New Row**.
 - Enter SPPUBLISHCHANNEL : AENDPOINT
 - Select the **Enabled** check box.
 - d. Select **Publish channels > Add New Row**.
 - Enter SPSAPUBLISHCHANNEL : SAENDPOINT
 - Select the **Enabled** check box.

Importing event data from IBM Maximo Asset Manager

IBM Predictive Maintenance and Quality can be customized to import IBM Maximo work orders as events to record activities such as inspections and repairs.

You must do the following tasks:

1. Create a publish channel in IBM Maximo to export the work orders.
Take care not to import work orders that are created by IBM Predictive Maintenance and Quality.
Modify the WorkorderCreation flow to set the EXTREFID field to PMQ. When you import the work order, do not import work orders that have the EXTREFID field that is set to PMQ.
For more information, see IBM Maximo Asset Management Knowledge Center (<http://www.ibm.com/support/knowledgecenter/SSWK4A>).
2. Create a flow in IBM Integration Bus to consume these work orders, map them to the standard event format, and place them on the event processing queue.
3. Create profile variables to determine how these events are processed into key performance indicators (KPIs) and profiles. For more information, see “Profile variables” on page 53
4. Modify the event processing flow to ensure that these events trigger scoring for an appropriate predictive model. For more information, see “Event processing” on page 60.

Creating a work order service in IBM Maximo Asset Management

To create a work order, an enterprise service must be created in IBM Maximo. The enterprise service defines a web service with a WSDL file. The work order creation service is called by an IBM Integration Bus flow in IBM Predictive Maintenance and Quality.

Before you begin

You must configure a web service in IBM Maximo Asset Management to create work orders in IBM Predictive Maintenance and Quality.

Configure IBM Maximo to expose a web service corresponding to the service defined in the **MaximoWorkOrder.wsdl** file in the **PMQMaximoIntegration** IBM Integration Bus application.

For more information about creating an enterprise service, see IBM Maximo Asset Management Knowledge Center (<http://www.ibm.com/support/knowledgecenter/SSWK4A>).

Procedure

Create a Web Service from the default work order enterprise service (MXWOInterface).

1. In IBM Maximo Asset Management, go to the **Web Services Library**, **Select Action**, **Create Web Service**, **Create Web Service from Enterprise Service**.
2. Select **EXTSYS1_MXWOInterface** and click **Create**.
3. Click the generated web service name (**EXTSYS1_MXWOInterface**) and **Select Action**, **Deploy to Product Web Service Container**, **Deploy Web Service** and click **OK**.
4. Turn on the functionality in IBM Predictive Maintenance and Quality to create work orders in IBM Maximo based on recommendations from the default predictive models. In the IBM WebSphere® MQ Explorer, set the **MaximoTRIGGER** user-defined property for the **PMQIntegration** flow to **TRUE**.
 - a. In the IBM WebSphere MQ Explorer, go to **Brokers > MB8Broker > PMQ1**. Right-click the **PMQIntegration** node, and click **Properties**.
 - b. Click **User Defined Properties**.
 - c. Set the **MaximoTRIGGER** value to **TRUE**.
5. Set the server name in the **Web Service URL** property of the **InvokeWorkOrder** node to the name of the IBM Maximo host. This node is in the sample **WorkorderCreation.msgflow** flow in the **PMQMaximoIntegration** application.
 - a. In the IBM WebSphere MQ Explorer, go to **Brokers > MB8Broker > PMQ1 > PMQMaximoIntegration > Flows**, and click **Workordercreations.msgflow**.
 - b. In the graphical display, right-click the **InvokeWorkOrder** node and select **Properties**.
 - c. In the **Web Services URL** field, enter the URL of the IBM Maximo host.

Configuring work orders in Maximo

In Maximo, you can configure Maximo for OutBound work orders using either an XML file in batch mode or using a web service in realtime mode.

You can also configure Maintenance work orders to be updated with recommendations in IBM Predictive Maintenance and Quality (PMQ).

Configuring Maximo for OutBound work orders that use a web service

You can configure Maximo for OutBound work orders that use a web service in real-time mode.

Procedure

1. Define the object structure.
 - a. Edit the base object structures available in IBM Maximo Asset Management (MXWO) to add the Service Address object reference to it.

Tip: This ensures that work order events that are generated from Maximo contain the field reference that is related to the service address.

- b. Select **GoTo > Integration > Object Structure** and search for MXWO.
- c. Click the new row and enter the following information:
 - In the **Object** field, type WOSERVICEADDRESS.
 - In the **Parent Object** field, type WORKORDER.
 - In the **Object Location Path** field, type WOSERVICEADDRESS.
 - In the **Relationship** field, type SERVICEADDRESS.
2. To export the Object Structure schema for MXWO, select **Action > Generate Schema/View XML**.

The generated schema MXWOService.xsd is stored in the same location as the data export XML files. This schema is used for configuring in the mapping node of IIB for work order to event transformation.
3. Enable the Publish channel event listener.
 - a. Select **Publish Channel** and then select **MXWOInterface**. The work order publish channel appears.
 - b. Select **Action > Enable Event Listeners**.

The **Enable Listener** check box is enabled. See the following figure.
4. Add a processing rule for the publish channel MXWOInterface.
 - a. Select **New Row**.
 - b. Enter the following values:
 - In the **Rule** column, type PMQ.
 - In the **Description** column, type PMQ Maintenance related Rule.
 - In the **Action** column, specify SKIP.
 - In the **Enabled** column, select the check box.
 - c. Select **Add/Modify Conditions**.
 - d. Select **New Row**.
 - e. Specify the following values:
 - In the **Field** field, type DESCRIPTION.
 - In the **Evaluation Type** field, type NOTEQUALS.
 - In the **Evaluation When** field, type ALWAYS.
 - In the **Value** field, type MAINTENANCE.

A condition is added to skip the MAINTENANCE work order.
 - f. Select **New Row**.
 - g. Specify the following values:
 - In the **Field** field, select DESCRIPTION.
 - In the **Evaluation Type** field, select NOTEQUALS.
 - In the **Evaluation When** field, select ALWAYS.
 - In the **Value** field, select BREAKDOWN.

A condition is added to skip the BREAKDOWN work order.
5. Activate the JMS cron task.
 - a. Select **GoTo > System Configuration > Platform Configuration > Cron Task Setup**.
 - b. Filter on **JMSQSEQCONSUMER**.
 - c. Select the **SEQQOUT** cron task instance name.
 - d. Click **Active** and save the record.

The JMS cron task is activated.

6. Configure the web service endpoint.
 - a. Select **GoTo > Integration > Endpoint**.
 - b. Select **New Endpoint** and enter the following information:
 - In the **Endpoint Name** field, type MXWOENDPOINT.
 - In the **Handler type** field, type WEBSERVICE.
 - In the **EndPointURL** field, type `http://ESB_Node_IP_address:7800/meaweb/services/MXWOInterface`.
 - In the **ServiceName** field, type OutboundW0Service.
7. Configure the external system to associate publish channels and endpoints to the external system for web service event notification of work orders.
 - a. Select **GoTo > Integration > External Systems > New External System**
 - b. Enter the following information:
 - In the **System** field, type EXTSYS2.
 - In the **Description** field, type PMQ External System.
 - In the **EndPoint** field, type MXXMLFILE.
 - In the **Outbound Sequential Queue** field, type `jms/maximo/int/queues/sqout`.
 - In the **Inbound Sequential Queue** field, type `jms/maximo/int/queues/sqin`.
 - In the **Inbound Continuous Queue** field, type `jms/maximo/int/queues/cqin`.
 - Select the **Enabled** check box.
 - c. Select **Publish channels > Add New Row**.
 - Add a New Row to add MXWOInterface to the publish channel with Endpoint as MXWOENDPOINT.
 - Select the **Enabled** check box.

Configuring Maximo for OutBound work orders using an XML file

You can configure Maximo for OutBound work orders using an XML file in batch mode.

Procedure

1. Create a new publish channel SPWO.
 - a. Select **GoTo > Integration > Publish Channels**.
 - b. Enter the following information:
 - In the **Publish Channel** field, type SPWO.
 - In the **Description** field, type PMQ WorkOrder Publish Channel.
 - In the **Object Structure** field, type MXWO.
- See the following figure.

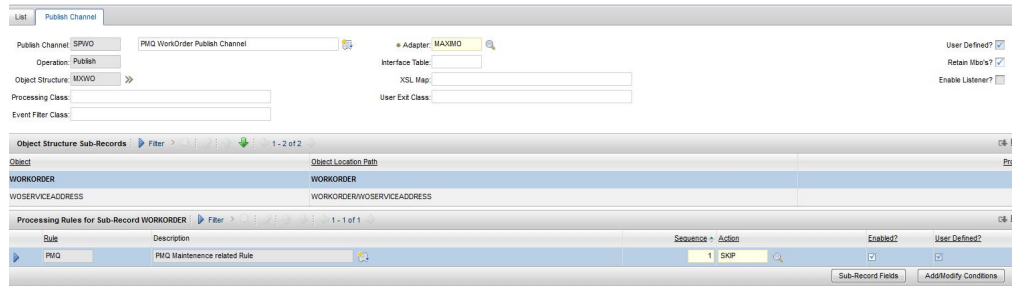


Figure 7. Create a new publish channel SPWO

2. Add a new processing rule for the publish channel SPWO.
 - a. Select **New Row**.
 - b. Specify the following values:
 - In the **Rule** column, type PMQ.
 - In the **Description** column, type PMQ Maintenance related Rule.
 - In the **Action** column, specify SKIP.
 - In the **Enabled** column, select the checkbox.
- See the following figure.

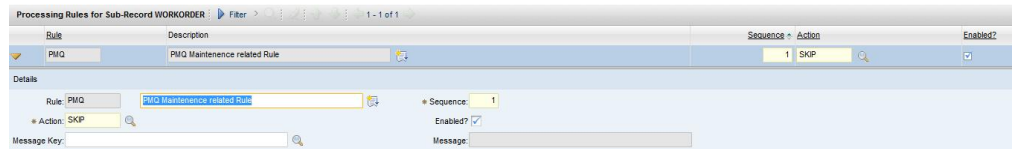


Figure 8. Adding a new processing rule for the publish channel SPWO

- c. Select **Add/Modify Conditions**.
- d. Select **New Row** under XML field evaluation.
- e. Specify the following values:
 - In the **Field** field, specify DESCRIPTION.
 - In the **Evaluation Type** field, specify NOTEQUALS.
 - In the **Evaluation When** field, specify ALWAYS.
 - In the **Value** field, specify MAINTENANCE.

A condition is added to skip the MAINTENANCE work order. See the following figure.

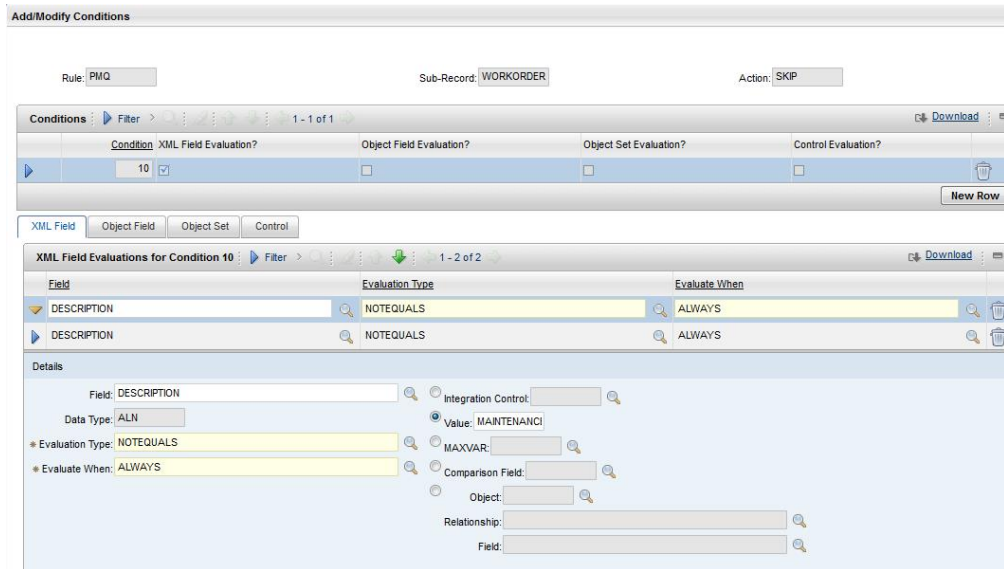


Figure 9. Adding a condition to skip the MAINTENANCE work order

- f. Select **New Row** under XML field evaluation.
- g. Specify the following values:
 - In the **Field** field, specify DESCRIPTION.
 - In the **Evaluation Type** field, specify NOTEQUALS.
 - In the **Evaluation When** field, specify ALWAYS.
 - In the **Value** field, specify BREAKDOWN.

A condition is added to skip the BREAKDOWN work order. See the following figure.

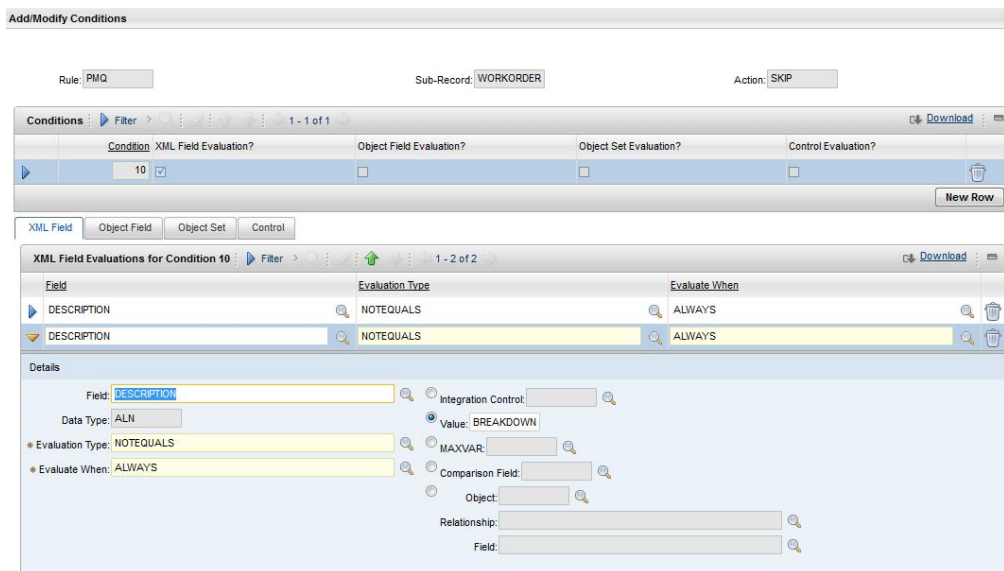


Figure 10. Adding a condition to skip the BREAKDOWN work order

3. Configure the external system to associate the publish channel and endpoint to the external system for the XML export of work orders.
 - a. Select **GoTo > Integration > External Systems**.

- b. Filter on SPEXTSYSTEM.
 - c. Select **Publish channels filter**.
 - d. Enter the following information:
 - In the **Publish Channel Name** field, type SPWO
 - In the **EndPoint** field, type MXXMLFILE
 - Enable the MXWOInterface for the external system SPEXTSYSTEM by selecting the **Enabled** check box.
 - Activate the external system (SPEXTSYSTEM) by selecting by selecting the **Enabled** check box.
- See the following figure.

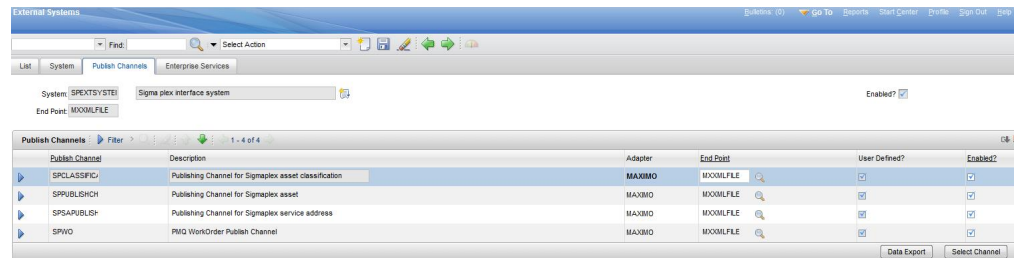


Figure 11. Enabling the external system SPEXTSYSTEM

Configuring Maximo to update recommendations in a work order

You can configure Maximo so that Maintenance work orders are updated in PMQ with PMQ recommendations.

The work order status is changed to CHANGED and Memo is updated to Refer LONGDESCRIPTION for PMQ recommendation. PMQ recommendation will get updated in the LONGDESCRIPTION field of PMQ.

The Maximo configuration described in this section creates the custom status CHANGED. The custom status CHANGED can be used to filter out all the work orders which were updated by PMQ with the recommendations.

Procedure

1. In Maximo, select **GoTo > System Configuration > Platform Configuration > Domains**.
2. Find the SYNONYM domain WOSTATUS to which you want to add a synonym value.

See the following figure.

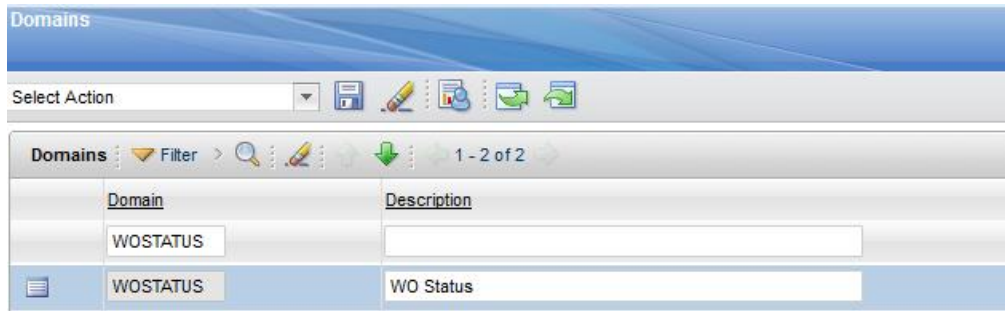


Figure 12. Finding the SYNONYM domain WOSTATUS

3. Click the **Edit details** icon.
4. Select **New Row** and specify the following values:
 - In the **Internal Value** field, specify WAPPR.
 - In the **Value** field, specify Change.
 - In the **Description** field, specify Recommendation Updated.
 See the following figure.

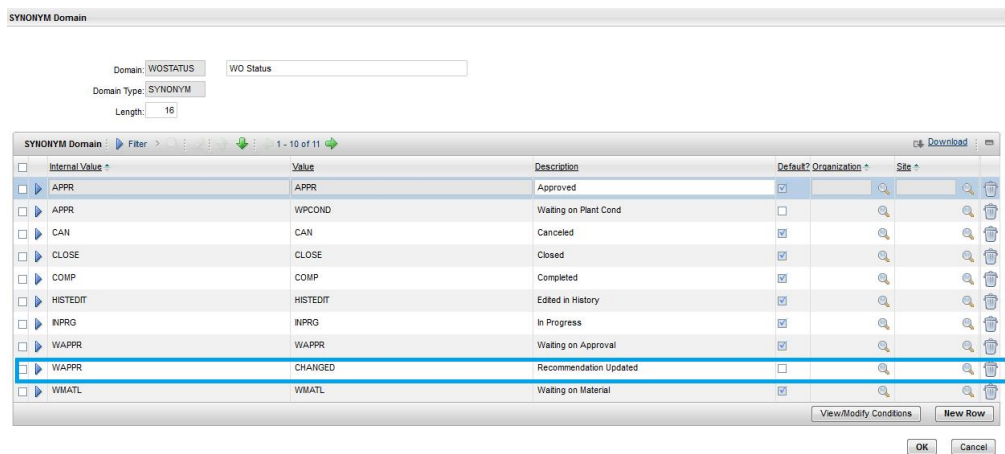


Figure 13. Specifying the values of the new row

Viewing work orders updated with PMQ recommendations

You can view work orders that have been updated with recommendations from IBM Predictive Maintenance and Quality.

Procedure

1. Select **Goto > Work Orders > Work Order Tracking**.
2. Select **Filter** and, in the **STATUS** field, specify **CHANGED**.
3. Open the work order and select the **Long description** button in the **Work Order** row.

See the following figure.

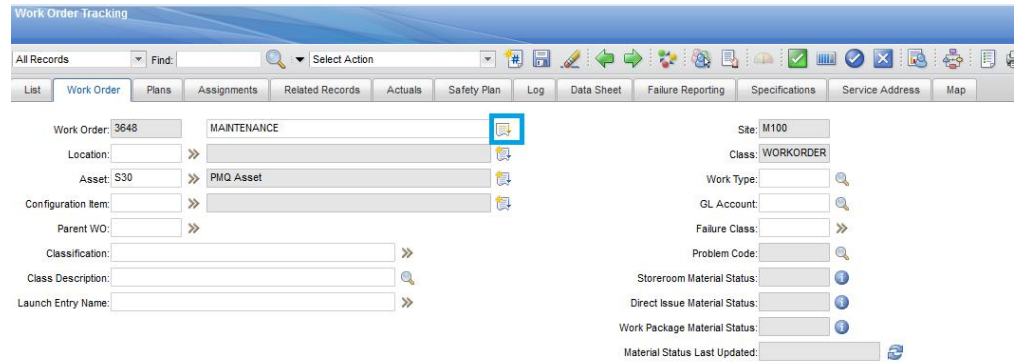


Figure 14. Opening the Long Description window

The PMQ recommendation appears, as shown in the following figure.

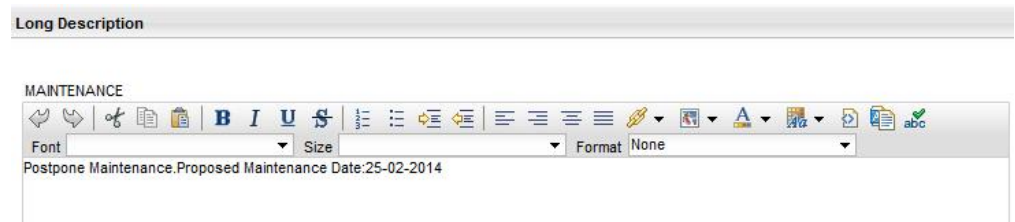


Figure 15. Viewing the PMQ recommendation

Creating a work order in Maximo

You can create a MAINTENANCE work order or a BREAKDOWN work order in Maximo.

Procedure

1. Select **Goto > WorkOrders > Work Order Tracking > New Work Order**.
2. Specify the following values:
 - In the **Description** field, specify either BREAKDOWN or MAINTENANCE.
 - In the **Site** field, specify the Model No of the resource.
 - In the **Asset** field, specify the Serial No of the resource.
 - In the **Service Address** field, specify the location.
3. If you are creating a MAINTENANCE work order, specify the following values:
 - In the **Scheduled Start** field, specify the scheduled maintenance start timestamp.
 - In the **Actual Start** field, specify the actual maintenance start timestamp, if applicable.
4. If you are creating a BREAKDOWN work order, specify the following values:
 - In the **Reported Date** field, specify the Breakdown timestamp.

Results

For an example of a BREAKDOWN work order, see the following figure.

Figure 16. Creating a BREAKDOWN work order

Mapping work orders for maintenance

You can map IBM Predictive Maintenance and Quality (PMQ) events to work orders for maintenance.

There are two types of work orders you can use for maintenance:

- Maintenance work orders
- Breakdown work orders

Mapping PMQ events to a Maintenance work order

Two PMQ events are generated from a maintenance work order: one event for scheduled maintenance (SM) and one event for actual maintenance (AM).

The event mapping is shown in the following table

Table 10. Mapping PMQ events to a Maintenance work order

PMQ event	Work Order	Remarks
incoming_event_cd	WONUM	Not applicable
event_type_cd	Not applicable	Hardcoded to "MAINTENANCE"
source_system_cd	Not applicable	Hardcoded to "MAXIMO"
process_cd	Not applicable	Not applicable
production_batch_cd	Not applicable	Not applicable
location_cd	WOSERVICEADDRESS. SADDRESSCODE	Not applicable
event_start_time	Scheduled Start	Timestamp field
event_end_time	Not applicable	Not applicable
event_planned_end_time	Not applicable	Not applicable
tenant_cd	Not applicable	Hardcoded to "PMQ"
operator_cd	Not applicable	Not applicable
Model	SITEID	Not applicable

Table 10. Mapping PMQ events to a Maintenance work order (continued)

PMQ event	Work Order	Remarks
serial_no	ASSETNUM	Not applicable
measurement_type_cd	Not applicable	Hardcoded to "SM" for scheduled maintenance event and "AM" for actual maintenance
observation_timestamp	Scheduled Start for scheduled maintenance Actual Start for actual maintenance	Timestamp field
value_type_cd	Not applicable	Hardcoded to "ACTUAL"
observation_text	DESCRIPTION_ LONGDESCRIPTION	Not applicable
Measurement	Not applicable	Not applicable
material_cd	Not applicable	Not applicable
multirow_no	Not applicable	Hardcoded to 1

Mapping PMQ events to a Breakdown work order

The event mapping is shown in the following table.

Table 11. Mapping PMQ events to a Breakdown work order

PMQ event	Work Order	Remarks
incoming_event_cd	WONUM	Not applicable
event_type_cd	Not applicable	Hardcoded to "MAINTENANCE"
source_system_cd	Not applicable	Hardcoded to "MAXIMO"
process_cd	Not applicable	Not applicable
production_batch_cd	Not applicable	Not applicable
location_cd	WOSERVICEADDRESS. SADDRESSCODE	Not applicable
event_start_time	Reported date	Timestamp field
event_end_time	Not applicable	Not applicable
event_planned_end_time	Not applicable	Not applicable
tenant_cd	Not applicable	Hardcoded to "PMQ"
operator_cd	Not applicable	Not applicable
Model	SITEID	Not applicable
serial_no	ASSETNUM	Not applicable
measurement_type_cd	Not applicable	Hardcoded to "BREAKDOWN"
observation_timestamp	Reported date	Timestamp field
value_type_cd	Not applicable	Hardcoded to "ACTUAL"

Table 11. Mapping PMQ events to a Breakdown work order (continued)

PMQ event	Work Order	Remarks
observation_text	DESCRIPTION_ LONGDESCRIPTION	Not applicable
measurement	Not applicable	Not applicable
material_cd	Not applicable	Not applicable
multirow_no	Not applicable	Hardcoded to 1

Migrating historical work orders from Maximo to PMQ

You can migrate historical work orders from Maximo to PMQ using the following process:

1. Perform a manual export of the work orders in Maximo.
2. In PMQ, import the work orders on the ESB node.
3. Work orders with a description of MAINTENANCE or BREAKDOWN are mapped with PMQ events and loaded into PMQ DataStore through a file processing flow.

Note: Loading historical work orders is a onetime activity.

Migrating real-time work orders from Maximo to PMQ

You can migrate real-time work orders from Maximo to PMQ using the following process:

1. In Maximo, a new work order is created with the description MAINTENANCE or BREAKDOWN.
2. A web service is invoked from Maximo to IBM Integration Bus (IIB).
3. When the work order is updated with the maintenance date, the web service sends the work order details to PMQ in the form of a SOAP XML message.
4. The SOAP message is mapped to PMQ events and loaded into PMQ DataStore.

Chapter 5. Event data

Event data is any data that you want to measure about an event. Data comes from many sources, and it must be transformed into a format that can be used by IBM Predictive Maintenance and Quality.

For example, if the event is recording an inspection result, you might want to record: who was the inspector, when did the event take place, which product lot was it based on, and what was the result of the inspection?

IBM Integration Bus transforms data into a format that can be used in IBM Predictive Maintenance and Quality.

IBM Integration Bus has a visual interface that you use to map the data structure of the source data into the expected format.

The loading of event data involves the following steps:

1. In IBM Integration Bus, define the content and format of the event information that is coming in.
2. Map the data into the format that is expected by IBM Predictive Maintenance and Quality. You can use the graphical mapper, or for more complicated mappings, you can use a programming language such as Java™.
3. A message flow is provided to load data from a file. To use this flow, specify the file and location, and set a predefined time interval for checking the location. The file can be in a comma separated value format, for more information, see “File format and location” on page 22. However, by modifying a message flow, other formats such as XML are supported.

The data is processed:

- The data structure is brought into the correct format, then ported into event tables in the data store.
- The KPI and profile tables are calculated. KPIs are used in predictive models or in reports.
- This information is used to call a scoring service to receive a recommendation that is based on the current state of the event.
- The predictive model to be used is defined.

For information about the file locations and names and file format, see “File format and location” on page 22.

How events are processed

You must connect event sources to IBM Predictive Maintenance and Quality to enable events to be processed.

Events are processed in IBM Integration Bus and stored in the database. The database has an event store to record events, tables for key performance indicators (KPIs), and profiles that are related to the event source. KPIs provide a history of performance over time. Profiles show the current state of the event, and also include recommended actions from predictive models. Profiles help to speed up scoring.

The following steps occur:

1. IBM Integration Bus receives the events and maps them into the format that is required by IBM Predictive Maintenance and Quality with a custom flow, if required.
2. Events go into a queue (PMQ.EVENT.IN) for further processing, either as single events or multiple events that are processed together for efficiency.
3. Processed events are inserted into the event store. The information in the events immediately updates KPIs for the current KPI period. A historical record of the KPI values for each period is maintained (a period is typically one day). The event data is also used to immediately update profiles, which contain information about the current state of the event source.

This diagram shows the flow of events into IBM Integration Bus and into the database.

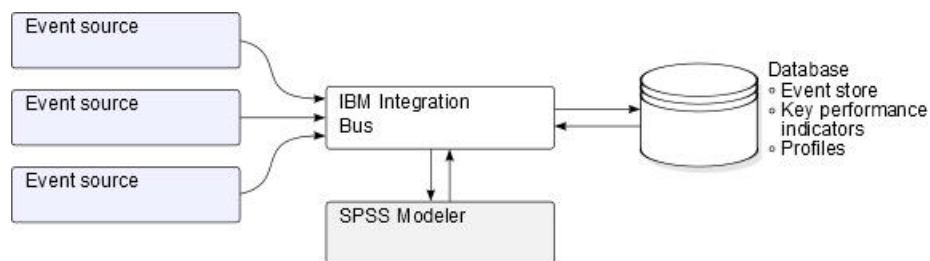


Figure 17. Flow of events into Integration Bus and into the database

The values in the event, KPI, and profile tables can be used as input to a predictive statistical model to generate recommended actions.

Processing events as they arrive, and immediately updating the aggregated values in the KPI and profile tables means that dashboards and reports are quickly updated with aggregated data.

Events must be loaded in chronological order. If events are not loaded in order then resulting KPIs and profiles may not be correct.

Event definition

Events are stored in the `event` and `event_observation` tables. An event can contain one or more event observations. Resource information is stored in the `event` table using `Resource_cd1` and `Resource_cd2`.

The calculated key performance indicators (KPIs) are stored in the `process_kpi` and `resource_kpi` tables. `Event_observations` update the values in the `process_kpi` and `resource_kpi` tables.

The calculated profile values are stored in the `process_profile`, `resource_profile`, and `material_profile` tables. The values in the row are updated as the events arrive. The tables contain values for the current period (day), prior period (prior day), and lifetime to date.

KPIs are calculated at the day level.

Flat file event input

Events can be in a flat file format (.csv) or .xml format that must conform to the format required by IBM Predictive Maintenance and Quality. Events can be in other forms, such as web services; however IBM Integration Bus flows must be modified and extended.

Each event contains information recorded by one or more measurements or observations. An event can be associated with one or more materials. Each event can also have an associated operator and or device.

However, each row of the input file can only define a single event, a single material, a single operator, and a single device. Therefore an event containing more than one of these must have more than one row.

The values supplied for `material_cd` associate these materials with the event.

An event requiring more than one observation row must set the optional `multi_row_no` to 1 in the first row of the event. Additional rows must be directly beneath this row and increase the value set in `multi_row_no` by 1 for each additional row.

If `Resource_cd1` has a value and `Resource_cd2` is blank or null, then this event must be associated with an Agent or Operator. If `Resource_cd1` and `Resource_cd2` both have non-blank values and have rows in the `Master_Resource` table with `Resource_type` being ASSET, then they are termed as events from a device or a resource.

Each row of a multi-row event usually has a different observation. The columns marked as observation in the following table have different values in each row of a multi-row event.

Ensure that events are pre-mapped into this format to enable them to be loaded via the application programming interface (API).

In the following table, the first ten fields, `incoming_event_cd` to `tenant_cd` are common to all of the rows of a multi-row event. Only the values in the first row are used. Many of these field are codes which reference values in the master data tables. See Appendix B, "The flat file API," on page 241.

Table 12. Fields in the Events table

Field	Type	Optional or required	Event or observation	Description
<code>incoming_event_cd</code>	string(50)	optional	event	A unique code that identifies the event.
<code>event_type_cd</code>	string(50)	required	event	The event type, such as measurement, alarm, inspection.
<code>source_system_cd</code>	string(50)	optional	event	The system generating the event.
<code>process_cd</code>	string(50)	optional	event	The production process related to the event.
<code>production_batch_cd</code>	string(50)	optional	event	The production batch related to the event.

Table 12. Fields in the Events table (continued)

Field	Type	Optional or required	Event or observation	Description
location_cd	string(50)	optional	event	The location of the event.
event_start_time	datetime	required	event	Time the event started in Coordinated Universal Time (UTC) format, for example 2002-05-30T09:30:10-06:00.
event_end_time	datetime	optional	event	Time the event ended in UTC format.
event_planned_end_time	datetime	optional	event	Time the event was planned to end in UTC format.
tenant_cd	string(50)	optional	event	The organization associated with the event.
Resource_cd1	string(50)	optional	event	The operator associated with the event.
Resource_cd2	string(50)	optional	event	The model number of the device associated with the event.
Resource_cd1	string(50)	optional	event	The serial number of the device associated with the event.
measurement_type_cd	string(50)	required	observation	The measurement type determines how the event observation will be processed.
observation_timestamp	datetime	required	observation	The time associated with the observation in UTC format.
value_type_cd	string(50)	optional	observation	The type of observation (actual, planned, or forecast).
observation_text	string(400)	optional (see note)	observation	A description associated with the event.
measurement	float	optional (see note)	observation	A measurement associated with the event.
material_cd	string(50)	optional	observation	The material used for an event.
multirow_no	integer	optional	Not Applicable	For multiple row events (more than one observation) use 1 to n for each row of the event.

Note: Either measurement or observation_text is required.

Schema definition of the event format

Events are processed in the event format that is shown in the following diagram. If you are extending IBM Predictive Maintenance and Quality to process external events from other sources, you must map those events to this internal event format.

The event schema is stored in the project PMQEventDataLibrary.

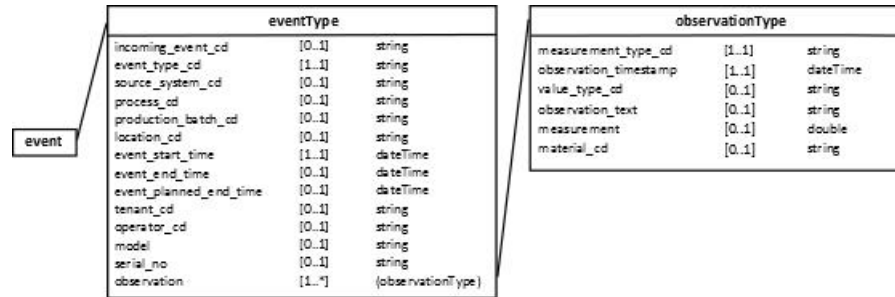


Figure 18. The event format used by IBM Predictive Maintenance and Quality

Error reporting

Errors can occur while processing events; during mapping to the required format or during the updating of the event, KPI, and profile tables.

You can add extra properties to the message to provide event source information for reporting while mapping to the IBM Predictive Maintenance and Quality format.

Profile and KPI tables

In addition to the event store and the master data, the IBM Predictive Maintenance and Quality database includes profile and KPI tables. The content of these tables is determined by a metadata driven aggregation mechanism that determines what calculations are performed when an event is processed.

The measurement_type and the resource_type or material_type values associated with the event and a particular event_observation, form the key that is used to look up the metadata.

Profile variables

The profile_variable table drives event processing in IBM Predictive Maintenance and Quality.

When an event_observation value arrives, its associated measurement_type value, and its associated resource_type value are used to find all of the profile_variable rows that are relevant for that observation, according to the orchestration defined for the event. Each of these rows indicates a calculation, which must be performed for the event. The calculation updates rows in the kpi and profile tables as indicated by the profile_variable. IBM Predictive Maintenance and Quality implements a standard set of calculations, but you can add a custom calculation and refer to it in a profile_variable row. The standard set of calculations include the following calculations:

- Measurement of type count
- Measurement text contains count

- Interval calculation
- Measurement above limit
- Measurement below limit
- Measurement delta

These calculations are described in “Profile calculations” on page 57.

To be able to process some events, you must load mandatory profile variables and measurement types. For more information, see “Mandatory profile variables and measurement types” on page 258.

For example, a temperature event with the measurement_type value “Ambient temperature” from a device can be aggregated by defining a profile_variable for the measurement_type “Ambient temperature” with the profile_calculation “Measurement of type” and adding a profile update for the measurement_type to the orchestration. This causes a row to be added to the resource_kpi table at each period for this device and profile_variable. This row aggregates the temperature values across each period (day). In addition, the defined profile_variable causes a row to be added to the resource_profile table for this device that is updated as each temperature event is processed.

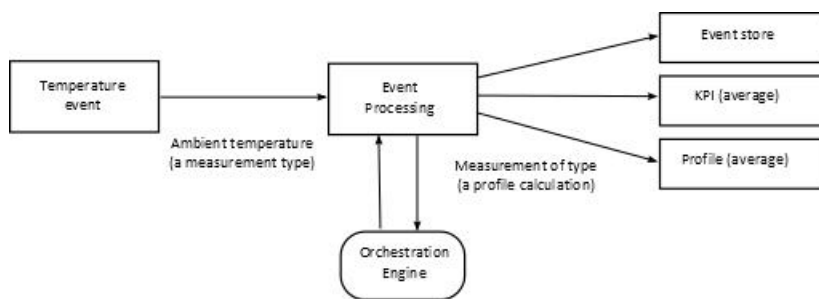


Figure 19. Temperature event workflow

Making a profile variable inactive

To make a profile variable inactive, for example, if you want to prevent a calculation from being performed, remove the profile update from the orchestration.

KPI tables

The IBM Predictive Maintenance and Quality key performance indicator (KPI) tables: resource_kpi and process_kpi hold aggregated values for each day.

In the resource_kpi table, the key for each row is determined by

- The profile_variable that triggered the calculation of the KPI
- The date
- The resource that is associated with the event
- The event code that is associated with the event observation
- The location that is associated with the event
- The process that is associated with the event
- The production batch that is associated with the event
- The tenant_id.

The fields in resource_kpi are described in the following table.

Table 13. Fields in the resource_kpi table

Field	Type	Description
kpi_date	date	The date for which the KPI is calculated. The time grain for KPI calculation is a single day.
profile_variable_id	integer	The profile variable that is the source of this KPI.
resource_id	integer	The resource that is associated with the event.
event_code_id	integer	The event code that is associated with the event observation. Event codes are codes for alarms, failures, issues, and so on. When an event arrives with a measurement_type value that has an event_code_indicator value of 1, the text from the event_observation_text field is assumed to contain an event_code value.
location_id	integer	The location that is associated with the event.
process_id	integer	The process that is associated with the event.
production_batch_id	integer	The production batch that is associated with the event.
actual_value	float	The actual value for this KPI. It is important to understand that for Business Intelligence reporting purposes, this value is typically divided by the measure count. Even if the value is meant to be an average, this value must be a sum of the values from the event and the measure_count must be the number of events. actual_value field supports the average calculation for dimensional reporting.
plan_value	float	The planned value for the KPI for this date.
forecast_value	float	The forecast value for the KPI for this date
measure_count	integer	The measure count for this date. Typically this value is used to divide the actual_value for reporting.
current_indicator	integer	Indicates that this row is the current row for a KPI. Typically the date of the current row is the current day.
tenant_id	integer	The tenant_id of the profile_variable that is the source of this KPI.

The fields in the process_kpi table are described in the following table.

Table 14. Fields in the process_kpi table

Field	Type	Description
process_id	integer	The process that is associated with the resource.
kpi_date	date	The date for which the KPI is calculated. The time grain for KPI calculation is a single day.
profile_variable_id	integer	The profile variable that is the source of this KPI.
material_id	integer	The material that is associated with the resource.
event_code_id	integer	The event code that is associated with the event observation. Event codes are codes for alarms, failures, issues, and so on. When an event arrives with a measurement_type value that has an event_code_indicator value of 1, the text from the event_observation_text field is assumed to contain an event_code value.

Table 14. Fields in the process_kpi table (continued)

Field	Type	Description
location_id	integer	The location that is associated with the resource.
production_batch_id	integer	The production batch that is associated with the event.
actual_value	float	The actual value for this KPI. It is important to understand that for Business Intelligence reporting purposes, this value is typically divided by the measure count. Even if the value is meant to be an average, this value must be a sum of the values from the resource and the measure_count must be the number of resources. actual_value field supports the average calculation for dimensional reporting.
plan_value	float	The planned value for the KPI for this date.
forecast_value	float	The forecast value for the KPI for this date.
measure_count	integer	The measure count for this date. Typically this value is used to divide the actual_value for reporting.
current_indicator	integer	Indicates that this row is the current row for a KPI. Typically the date of the current row is the current day.
tenant_id	integer	The tenant_id of the profile_variable that is the source of this KPI.

Profiles

Profiles provide pre-aggregated values to enable near real-time display in reports and dashboards.

The fields in the resource_profile table are described in the following table.

Table 15. Fields in the resource_profiles table

Field	Type	Description
resource_id	integer	The resource associated with this profile.
profile_variable_id	integer	The profile_variable that is the source of this profile.
value_type_id	integer	Value type of this profile. One of actual, plan, and forecast.
event_code_id	integer	The event code associated with the event observation. These are codes for alarms, failures, issues, and so on. When an event arrives with a measurement_type having an event_code_indicator of 1, the text from event_observation_text is assumed to contain an event_code.
location_id	integer	The location associated with the event.
profile_date	datetime	This date is based on the timestamp of the most recent event used to update the profile.
last_profile_date	datetime	Not Applicable
period_average	float	The average value for the period.
period_min	float	The minimum value for the period.
period_max	float	The maximum value for the period.

Table 15. Fields in the resource_profiles table (continued)

Field	Type	Description
period_total	float	The total value for the period.
period_std_dev	float	The standard deviation for the period.
period_msr_count	integer	The number of events contributing to this profile for the current period.
prior_average	float	The average value for the prior period.
prior_min	float	The minimum value for the prior period.
prior_max	float	The maximum value for the prior period.
prior_total	float	The total value for the prior period.
prior_std_dev	float	The standard deviation for the prior period.
prior_msr_count	integer	The number of events contributing to this profile for the prior period.
ltd_average	float	The average value lifetime to date.
ltd_min	float	The minimum value lifetime to date.
ltd_max	float	The maximum value lifetime to date.
ltd_total	float	The total value lifetime to date.
ltd_std_dev	float	The standard deviation lifetime to date.
ltd_msr_count	integer	The number of events contributing to this profile for the lifetime to date.
last_value	float	The most recent value in event_observation.measurement that updated this profile.
tenant_id	integer	The tenant_id of the profile_variable that is the source of this KPI.

Profile calculations

Profile calculations update the key performance indicator (KPI) and profile table (the `kpi_indicator` and `profile_indicator` values are updated). A profile variable specifies the profile calculations to perform for an observation with a given measurement type.

A profile variable maps a measurement type to a profile calculation. There may be zero or more profile variables for a given measurement type.

The following section describes the default profile calculations.

Note: Not all of the profile calculations are covered. Only the profile calculations used by BI and Analytics are covered as a part of Foundation porting.

Measurement of type

This calculation is based on the value of a particular `measurement_type`.

- KPI: the `actual_value` column contains the sum of all the `event_observation.measurement` values. The `measure_count` column contains a count of all the `event_observation` events.
- Profile: the average, minimum, maximum, total, and standard deviations are calculated for the present, prior (previous day), and lifetime to date periods. The

average value in the profile is the true average and, unlike KPI, is not divided by the corresponding `msr_count` value. These values can be calculated on a running basis for efficiency. The `msr_count` values record the count of all the `event_observation` events in the period. The `last_value` column contains the most recent `event_observation.measurement` values.

Measurement of type count

A count of the number of times an event with a particular `measurement_type` occurs.

- KPI: the `actual_value` and `measure_count` columns contain a count of the occurrences of the specified `event_observation`.
- Profile: the `msr_count` values record the count of the `event_observation` events in the period.

Measurement text contains count

A count of the number of times an event observation text contains a string. The string is the value of the `profile_variable.comparison_string`.

- KPI: the `actual_value` and `measure_count` columns contain a count of the occurrences of the specified `event_observation` events.
- Profile: the `msr_count` values record the count of the `event_observation` events in the period.

Measurement above limit

This is a count of the number of times the `event_observation.measurement` value falls above the value of the profile variable (`high_value_number`).

- KPI: the `actual_value` and `measure_count` columns contain a count of the occurrences of the specified `event_observation`.
- Profile: the `msr_count` values record the count of the `event_observation` events in the period.

Measurement below limit

This is a count of the number of times the `event_observation.measurement` value falls below the value of the profile variable (`low_value_number`).

- KPI: the `actual_value` and `measure_count` columns contain a count of the occurrences of the specified `event_observation` events.
- Profile: the `msr_count` values record the count of the `event_observation` events in the period.

Measurement delta

This is the change from one measurement value to the next.

- KPI: the `actual_value` column contains a sum of all the changes in measurement values. The `measure_count` column contains a count of all the `event_observation` events.
- Profile: The `msr_count` value should be 1 if the `event_observation` event occurs in the period. The `profile_date` value has the timestamp of the most recent `event_observation` event.

Custom calculations

You can modify the event processing flow to support extra calculations.

The custom calculation must be defined in the solution definition file. The custom calculation must be implemented as a Java Class implementing the `com.ibm.analytics.foundation.calculation.api.Calculation`.

Predictive scoring

To provide a health score for predictive models, code is required in the event processing flow.

A scoring service requires a defined set of inputs and returns a result. The score returns either a numerical value, a recommendation, or both. The sources of data for the input to the scoring service are the Event, KPI (key performance indicator), and Profile tables. Code transforms the data that is needed to supply the exact set of input parameters that are required by the scoring service. The scoring service is called by a web service call from IBM Integration Bus.

When the results are returned from the scoring service, they are written back as new events. Measurement types and profile variables can be defined for these events.

For example, a health score and recommendation can be recorded as an `event_observation.measurement` and `event_observation.observation_text`. In addition to being stored in the event tables, this score and recommendation can be aggregated for IBM Cognos Business Intelligence Reporting by defining two `profile_variables` and the corresponding profile updates in the profile adapter configuration of an orchestration.

To aggregate the health score, define a `profile_variable` and `profile_adapter` configuration for the Measurement of type calculation.

To aggregate the occurrences of a specific recommendation, one needs to define a `profile_variable` and `Profile_adapter` configuration for a Text contains calculation and set the `comparison_string` attribute of `profile_variable` and `profile_adapter` to the name of the recommendation.

The processing of an event that contains the result of a predictive scoring service can invoke a second scoring service.

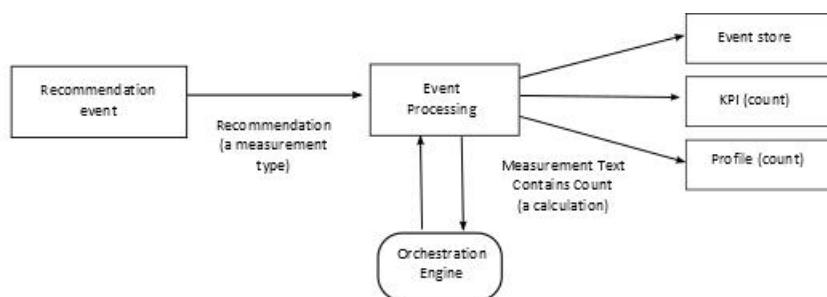


Figure 20. The flow of a scoring service

How scoring is triggered

Scoring for predictive models is triggered based on the service adapter configuration defined in the orchestration xml file. To construct any custom

scoring, the orchestration xml file must be defined accordingly.

Events and actual, planned, and forecast values

Typically, events contain actual values. Special events may contain planned values, and forecast values.

At least one event containing planned or forecast values, should be supplied for each KPI reporting period (day). This allows planned and forecast values to appear on IBM Cognos Business Intelligence reports along with actual values.

Event processing queue

Two queues are used to gather events for processing. One queue is for events read from .csv files, or transformation flows that you have developed. The other queue is for events that are generated from scoring results. You can use additional queues for processing but only one queue can contain events that update the same Key Performance Indicators (KPI) or profile rows. Typically, a queue supports events from an exclusive set of resources or processes.

A queue is used to hold events for processing in a single thread. The queue only contains events already mapped to the IBM Predictive Maintenance and Quality format.

Event processing

Event processing consists of the following steps.

1. Lookup of primary keys for supplied business keys.
2. Insertion of events.
3. Updating and inserting the KPI and Profile rows.
4. Scoring using an IBM SPSS predictive model.
5. Making a recommendation using IBM Analytical Decision Management.
6. Work order creation.

Recording and acting on predictive scores and recommendations

Profile variables are used to determine what key performance indicators (KPI) and profile calculations must be performed for an event. However, profile variables do not determine whether scoring or decision management is performed for an event. Scoring or decision management is determined by the service adapter definition in the Orchestration XML. This Orchestration XML must be modified to provide customization in scoring and decision making.

Scores that are returned by a predictive model and recommendations that are returned by decision management are processed and recorded in the same way as events that are received from a device. This means that scores and recommendation results are written to a content store, KPIs and profiles are calculated for these values, and the values are displayed in reports.

This reuse of the event processing mechanism is implemented by creating an event that uses the standard event format. An appropriate event type and measurement type are used for the event. The event is further processed, based on the service adapter definition defined in the orchestration xml file. Events on this internal event processing queue are processed by the same flow as external events. Profile

variables and profile updates in the profile adapter configuration are defined to control the processing of these internal events to calculate KPI and Profile values.

If IBM Predictive Quality and Maintenance is configured to work with IBM Maximo Asset Management, a recommendation can result in the creation of a work order in IBM Maximo. Customizing this behavior also requires modifying ESQL code.

For more information, see Chapter 9, “Recommendations,” on page 201

Threads

Events are processed by only one flow that runs in a single thread. If more than one flow is implemented to process events, these different flows must not update the same KPI or Profile rows. A single thread is required to ensure that only a single thread is calculating and updating a row in the KPI and Profile tables.

Batch processing

Event processing can occur faster by processing more than one event at the same time through batch processing. For example, if you want to process and load event data for one year, then you can do so by processing the events through multiple .csv files.

Use this approach only if the separate files contain events from separate devices.

- Create copies of `MultiRowEventLoad` flow and deploy on the broker. Each copy of message flow processes one .csv file at a time.
- Ensure that you do not set the `AdditionalInstances` property of the `MultiRowEventLoad` flow to greater than 0 for processing the batch concurrently.
- Ensure that the events from the same resource are combined into a single file in chronological order.

Parallel processing

Event processing can also occur faster by processing more than one event at the same time. However, it is important that only one thread at a time updates a row in the KPI or profile tables. Since the rows in these tables are related to resources and measurement types, achieve thread isolation by ensuring that events from an individual resource or of a particular measurement type are processed by a single thread. You can implement parallel processing by using multiple queues to manage the separation of events.

Event processing assumes that only one thread updates an individual row in the `resource_kpi`, `resource_profile`, `process_kpi`, `process_profile`, and `material_profile` tables. This is true for events from external devices and internal events that record recommendations. This means that parallelism can be achieved only by segmenting the events into groups that do not share resources, processes, or materials. To achieve parallelism, you must deploy multiple copies of event and integration flows, and ensure that each copy of the message flow uses a unique set of queues.

Remove events

Normally events are not deleted from the analytic database. During testing and development, events can be removed by deleting the appropriate rows from the `event`, `event_observation`, and `event_resource` tables.

As events are processed, extra internal events are added when predictive scoring and decision management are performed. You can also remove these events.

Sample remove event code

The following SQL code is an example and must be modified.

```
DELETE FROM SYSREC.EVENT_RESOURCE ER WHERE...
DELETE FROM SYSREC.EVENT_OBSERVATION EO WHERE...
DELETE FROM SYSREC.EVENT E WHERE...
```

Event processing also adds row to the KPI and profile tables, and you can remove those rows by modifying the following SQL.

```
DELETE FROM SYSREC.RESOURCE_KPI RK WHERE...
DELETE FROM SYSREC.RESOURCE_PROFILE RP WHERE...
DELETE FROM SYSREC.PROCESS_KPI PK WHERE...
DELETE FROM SYSREC.PROCESS_PROFILE PP WHERE...
DELETE FROM SYSREC.MATERIAL_PROFILE MP WHERE...
```

Configuring solution.xml for the event flow

The event definition, like the master data definition, is part of the solution XML file.

Within the solution.xml for event handling, there is one xml structure for a table where both event and event_observation is covered. The event_resource used in PMQ 1.0 is removed by having the resource_information defined within the event xml. Within the event definition, there is a separate tag called observation with the table_cd element.

```
<event_definition>
  <table table_cd="EVENT">
    <column column_cd="EVENT_START_TIME" type="timestamp" />
    <column column_cd="EVENT_END_TIME" type="timestamp" is_nullable="true" />
    <column column_cd="EVENT_PLANNED_END_TIME" type="timestamp" is_nullable="true" />
    <column column_cd="INCOMING_EVENT_CD" type="string" size="200" is_nullable="true" />
    <reference reference_cd="ASSET_ID" table_reference="MASTER_RESOURCE">
      <column_mapping reference_column_cd="SERIAL_NO" table_column_cd="RESOURCE_CD1"/>
      <column_mapping reference_column_cd="MODEL" table_column_cd="RESOURCE_CD2"/>
    </reference>
    <reference reference_cd="AGENT_ID" table_reference="MASTER_RESOURCE">
      <column_mapping reference_column_cd="OPERATOR_CD" table_column_cd="RESOURCE_CD1"/>
      <column_mapping reference_column_cd="OPERATOR_NA" table_column_cd="RESOURCE_CD2"/>
    </reference>
    <reference reference_cd="EVENT_TYPE_ID" table_reference="MASTER_EVENT_TYPE" />
    <reference reference_cd="SOURCE_SYSTEM_ID" table_reference="MASTER_SOURCE_SYSTEM" />
    <reference reference_cd="PROCESS_ID" table_reference="MASTER_PROCESS" />
    <reference reference_cd="PRODUCTION_BATCH_ID" table_reference="MASTER_PRODUCTION_BATCH" />
    <reference reference_cd="LOCATION_ID" table_reference="MASTER_LOCATION"/>
    <observation table_cd="EVENT_OBSERVATION">
      <column column_cd="OBSERVATION_TIMESTAMP" is_key="true" type="timestamp" />
      <column column_cd="OBSERVATION_TEXT" type="string" size="800" is_nullable="true" />
    </observation>
    <column column_cd="MEASUREMENT" type="double" is_nullable="true"/>
    <reference reference_cd="MEASUREMENT_TYPE_ID" is_key="true" table_reference="MASTER_MEASUREMENT_TYPE" />
    <reference reference_cd="VALUE_TYPE_ID" is_key="true" table_reference="MASTER_VALUE_TYPE" />
    <reference reference_cd="EVENT_CODE_ID" is_key="true" table_reference="MASTER_EVENT_CODE"/>
    <reference reference_cd="MATERIAL_ID" table_reference="MASTER_MATERIAL"/>
    <event_interval_column column_cd="OBSERVATION_DATE" type="date" />
    <event_interval_column column_cd="OBSERVATION_TIME" type="time" />
  </table>
</event_definition>
```

For handling the resource related information there are two references defined in the event xml.

```
<reference reference_cd="ASSET_ID" table_reference="MASTER_RESOURCE">
  <column_mapping reference_column_cd="SERIAL_NO" table_column_cd="RESOURCE_CD1"/>
  <column_mapping reference_column_cd="MODEL" table_column_cd="RESOURCE_CD2"/>
</reference>
<reference reference_cd="AGENT_ID" table_reference="MASTER_RESOURCE">
  <column_mapping reference_column_cd="OPERATOR_CD" table_column_cd="RESOURCE_CD1"/>
  <column_mapping reference_column_cd="OPERATOR_NA" table_column_cd="RESOURCE_CD2"/>
</reference>
```

If the referenced resource is a ASSET or AGENT.

The xml structure with in the event for handling the observation part is defined by a separate xml element called observation.

```
<observation table_cd="EVENT_OBSERVATION">
  <column column_cd="OBSERVATION_TIMESTAMP" is_key="true" type="timestamp" />
  <column column_cd="OBSERVATION_TEXT" type="string" size="800" is_nullable="true" />
  <column column_cd="MEASUREMENT" type="double" is_nullable="true"/>
  <reference reference_cd="MEASUREMENT_TYPE_ID" is_key="true" table_reference="MASTER_MEASUREMENT_TYPE" />
  <reference reference_cd="VALUE_TYPE_ID" is_key="true" table_reference="MASTER_VALUE_TYPE" />
  <reference reference_cd="EVENT_CODE_ID" is_key="true" table_reference="MASTER_EVENT_CODE"/>
  <reference reference_cd="MATERIAL_ID" table_reference="MASTER_MATERIAL"/>
  <event_interval_column column_cd="OBSERVATION_DATE" type="date" />
  <event_interval_column column_cd="OBSERVATION_TIME" type="time" />
</observation>
```

Chapter 6. Quality early warning system use cases

The quality early warning system (QEWS) in IBM Predictive Maintenance and Quality detects emerging quality problems sooner and with fewer false alarms than is typically achieved by traditional statistical process control. To achieve earlier detection, QEWS is sensitive to subtle changes in data values, such as shifts that are small in magnitude or trends that grow slowly over time. For a given level of statistical confidence, QEWS typically needs fewer data points than traditional statistical process control.

Early detection of quality problems is essential where delayed detection can have significant negative consequences, such as in the following scenarios:

- Building a large inventory of defective products results in high scrap costs.
- Shipping a large quantity of defective products to distribution channels or customers causes high warranty expenses.
- Having widespread quality or reliability problems in the field results in damage to brand value.
- Compromised production of supply-constrained materials or components prevents on-time shipment.
- Compromised production of products with long manufacturing times results in shipment delays.

Products are the subjects of QEWS analyses. A product is typically a part or a part assembly, but it can also be a process or a material. Products might be used in larger finished assemblies, which QEWS calls *resources*.

QEWS offers three use cases. *Quality inspection* detects unfavorable changes in the quality of components. *Warranty* detects warranty issues sooner than they might otherwise be detected, which results in fewer warranty claims and lower costs. *Parametric* detects unfavorable changes in variable-type data, and provides information that facilitates diagnostics and alarm prioritization.

Quality inspection

In a manufacturing environment, defects can occur in a manufacturing process because of variations in factors like process, raw materials, design, and technology. The resulting low quality of products creates a larger inventory of defective lots, which leads to increased inspection effort.

A small delay in detecting a quality problem can result in large costs, lost opportunity, and lost brand value.

The quality early warning system (QEWS) in IBM Predictive Maintenance and Quality evaluates evidence to determine whether the rate of failures is at an acceptable level. QEWS highlights combinations for which the evidence exceeds a specified threshold. QEWS can detect emerging trends earlier than traditional statistical process control, such as trend analysis. QEWS maintains a specified low rate of false alarms. Post-warning analysis of charts and tables identifies the point of origin, the nature and severity of the problem, and the current state of the process.

The QEWS quality inspection use case analyzes data from the inspection, testing, or measurement of a product or process operation over time. The data can be obtained from the following sources:

- suppliers (for example, the final manufacturing test yield of a procured assembly)
- manufacturing operations (for example, the acceptance rate for a dimensional check of a machined component)
- customers (for example, survey satisfaction ratings)

The quality inspection solution is not connected only to products. It is also connected to resource, process, material, and location entities. References to these entities are included in the PRODUCT_KPI and PRODUCT_PROFILE tables so that a product can be associated with any resource, process, material, location, or a combination of these entities during inspection analysis.

You can adjust the frequency at which data is captured and input to QEWS, and the frequency at which QEWS analyses are run, according to the requirements of each situation. For example, monitoring the quality levels of assemblies that are procured from a supplier might best be done on a weekly basis; monitoring the quality levels of units that are moving through a manufacturing operation might best be done on daily basis.

Business and technical challenges

You need the best techniques to examine quality data from tens of thousands of products and to provide proactive quality management.

You need to be able to detect process variability that is not visible through traditional means such as trend analysis. QEWS can evaluate trace data and predict with a tunable confidence whether the variability in the data is natural “noise” or a subtle indication of an impending problem. This ability is a significant improvement over traditional statistical process control.

Business challenges

Better analytical methods are available but are difficult to implement. This is due to complex computational challenges and constraints in software implementation.

Technical challenges

Manufacturing process variations can occur slowly. Gradual changes in product quality are not detected or they are detected too late, which leads to a large inventory of suspicious or defective lots. This results in increased inspection effort, more low quality products, and more waste.

Defining the quality inspection solution

To define the quality inspection solution, you must load master data, load event data, define messages flows, and define the output location of the inspection analysis.

Procedure

1. Load master data. For more information about loading master data, see Chapter 4, “Master data,” on page 21.

2. Load event data. You can load event data in batch mode or in real time. For more information about loading event data, see Chapter 5, “Event data,” on page 49.
3. Define message flows. For more information about message flows, see “Message flows” on page 13.

Results

IBM Cognos Business Intelligence uses data in the PRODUCT_KPI and PRODUCT_PROFILE tables to generate the inspection dashboards and reports.

Quality inspection solution details

You must consider some requirements when you are loading the master data and event data tables.

Master data tables are loaded by the master flows. The following tables are required to implement an inspection use case:

Master_Event_Type

You must define the following event types in the Master_Event_Type table:

PRODUCTION

Defines the products that are produced by the process.

INSPECTION

Defines the sample set of products that are being inspected.

The following text is an example of a CSV file that is used to load the Master_Event_type table:

```
event_type_cd,event_type_name,language_cd,tenant_cd
PRODUCTION,PRODUCTION,EN,PMQ
INSPECTION,INSPECTION,EN,PMQ
```

Master_Value_Type

There are three possible values for value_type_cd in the Master_Value_Type table: ACTUAL, PLAN, FORECAST. Usually, the data that is associated with PRODUCTION or INSPECTION events is ACTUAL.

The following text is an example of a CSV file that is used to load the Master_Value_Type table:

```
value_type_cd,value_type_name,language_cd,tenant_cd
ACTUAL,Actual,EN,PMQ
PLAN,Plan,EN,PMQ
FORECAST,Forecast,EN,PMQ
```

Master_Location

The Master_Location table contains information that is specific to the location where the event occurs, or the resource that produces the event.

The following text is an example of a CSV file that is used to load the Master_Location table:

```
location_cd,location_name,region_cd,region_name,country_cd,
country_name,state_province_cd,state_province_name,city_name,latitude,
longitude,language_cd,tenant_cd,is_active
Tokyo,Tokyo,AP,Asia Pacific,JP,Japan,TY,Tokyo,TokyoCity, 35.41,139.45,
EN,PMQ,1
```

Master_Measurement_Type

The Master_Measurement_Type table defines how the observation is read or used. For inspection, the measurement_type is INSPECT and FAIL. The INSPECT measurement defines how many units of product were inspected or tested for quality. The FAIL measurement defines whether the outcome of the inspection is successful or not, which is identified by a flag with the FAIL.

The following text is an example of a CSV file that is used to load the Master_Measurement_Type table:

```
measurement_type_cd,measurement_type_name,unit_of_measure,
carry_forward_indicator,aggregation_type,event_code_indicator,language_cd,
tenant_cd
INSPECT,INSPECTION,,0,AVERAGE,0,EN,PMQ
FAIL,FAIL QTY INDICATOR,,0,AVERAGE,0,EN,PMQ
INSP_LAM0,Inspection Acceptance Level,,0,SUM,0,EN,PMQ
INSP_LAM1,Inspection Unacceptance Level,,0,SUM,0,EN,PMQ
INSPECT_NO_DAYS,Inspect No of days,,0,SUM,0,EN,PMQ
INSP_PROB0,Inspection Confidence Probability,,0,SUM,0,EN,PMQ
```

Parameter names are loaded as measurement type. Parameter names such as **LAM0**, **LAM1**, and **PROB0** are all considered for measurement type, because their values are loaded by using event formats.

Master_Product

The Master_Product table contains the core data for the inspection use case. This table stores information that is related to a product and the product_type.

The following text is an example of a CSV file that is used to load the Master_Product table:

```
product_cd,product_name,product_type_cd,product_type_name,language_cd,tenant_cd,
is_active
WT2444,Wind Turbine,Type Turbine,Type Turbine,EN,PMQ,1
Prd_No_1,Product Name 1,Type1,Type1,EN,PMQ,1
Prd_No_2,Product Name 2,Type2,Type2,EN,PMQ,1
Prd_No_3,Product Name 3,Type3,Type3,EN,PMQ,1
Prd_No_4,Product Name 4,Type4,Type4,EN,PMQ,1
Prd_No_5,Product Name 5,Type5,Type5,EN,PMQ,1
Prd_No_6,Product Name 6,Type6,Type6,EN,PMQ,1
Prd_No_7,Product Name 7,Type7,Type7,EN,PMQ,1
Prd_No_8,Product Name 8,Type8,Type8,EN,PMQ,1
Prd_No_9,Product Name 9,Type9,Type9,EN,PMQ,1
Prd_No_10,Product Name 10,Type10,Type10,EN,PMQ,1
```

Master_Production_Batch

The Master_Production_Batch table contains information about each production batch that is used to produce a product. The information includes the product that is produced, the date it is produced, and the batch information.

The following text is an example of a CSV file that is used to load the Master_Product table:

```
production_batch_cd,
production_batch_cd,production_batch_name,product_cd,product_type_cd,
produced_date,language_cd,tenant_cd
T1,Turbine,WT2444,Type Turbine,2010-01-01,EN,PMQ
T2,Turbine,WT2444,Type Turbine,2011-01-01,EN,PMQ
PB 1,Production Batch 1,Prd_No_1,Type1,2011-12-08,EN,PMQ
PB 2,Production Batch 2,Prd_No_2,Type2,2011-03-18,EN,PMQ
PB 3,Production Batch 3,Prd_No_3,Type3,2012-01-04,EN,PMQ
PB 4,Production Batch 4,Prd_No_4,Type4,2012-06-06,EN,PMQ
PB 12,Production Batch 12,Prd_No_4,Type4,2012-06-06,EN,PMQ
```

```

PB 5,Production Batch 5,Prd_No_5,Type5,2012-10-26,EN,PMQ
PB 6,Production Batch 6,Prd_No_6,Type6,2013-07-07,EN,PMQ
PB 7,Production Batch 7,Prd_No_7,Type7,2011-11-28,EN,PMQ
PB 8,Production Batch 8,Prd_No_8,Type8,2011-12-19,EN,PMQ
PB 9,Production Batch 9,Prd_No_9,Type9,2012-08-17,EN,PMQ

```

Master_Profile_Variable

The Profile Parameters table uses parameters across the Product, Production Batch, Resource, Material, Process, and Location master entities. The Profile Parameter table entries are built by using the Profile Variable Code as one of the keys, along with related master keys (mainly Product, with either or all other master keys, such as Resource, Process, Material, and Location). To start loading the Profile Parameters table, the Master_Profile_Variable table must be prepared. The convention that is used for the Profile Variable Code is INSP_ combined with **Parameter Name**. For example, for the parameter name LAM0, the Profile Variable Code is INSP_LAM0.

The following text is an example of a CSV file that is used to load the Master_Profile_Variable table:

```

profile_variable_cd,profile_variable_name,profile_calculation_name,
measurement_type_cd,resource_type_cd,material_type_cd,profile_units,comparison_string,
low_value_date,high_value_date,low_value_number,high_value_number,kpi_indicator,
profile_indicator,data_type,aggregation_type,carry_forward_indicator,
process_indicator,variance_multiplier,language_cd,tenant_cd
INSP_LAM0,Inspection Acceptance Level,ASSIGN,INSP_LAM0,ASSET,-NA-
,,,,,1,1,INT,,0,0,1,EN,PMQ
INSP_LAM1,Inspection Unacceptable Level,ASSIGN,INSP_LAM1,ASSET,-
NA-,,,,,1,1,INT,,0,0,1,EN,PMQ
INSPECT_NO_DAYS,Inspection No of days,ASSIGN,INSPECT_NO_DAYS,ASSET,-
NA-,,,,,1,1,INT,,0,0,1,EN,PMQ
INSP_PROB0,Inspection Confidence Probability,ASSIGN,INSP_PROB0,ASSET,-
NA-,,,,,1,1,INT,,0,0,1,EN,PMQ

```

Event data loading

In addition to normal batch events, parameter values are loaded by using events. Parameters are loaded first, followed by normal inspection-related events. The following text is an example of the event format that is used to load parameter values for inspection, supported by the event type PARAMETERVI.

```

incoming_event_cd,event_type_cd,source_system_cd,process_cd,prod_batch_cd,
location_cd,event_start_time,event_end_time,event_planned_end_time,tenant_cd,operator_cd,
model,serialNo,measurement_type_cd,observation_timestamp,value_type_cd,observation_text,
measurement,material_code,multirow_no
1,PARAMETERVI,,-NA-,PP9-XX9-009,-NA-,2014-12-02 00:51:35,2014-12-02 00:51:35,2014-12-02
00:51:35,PMQ,,-NA-,-NA-,INSP_LAM0,2014-12-02 00:51:35,ACTUAL,INSP_LAM0,5,-NA-,1
2,PARAMETERVI,,-NA-,PP9-XX9-009,-NA-,2014-12-02 00:51:35,2014-12-02 00:51:35,2014-12-02
00:51:35,PMQ,,-NA-,-NA-,INSP_LAM1,2014-12-02 00:51:35,ACTUAL,INSP_LAM1,8.5,-NA-,1
3,PARAMETERVI,,-NA-,PP9-XX9-009,-NA-,2014-12-02 00:51:35,2014-12-02 00:51:35,2014-12-02
00:51:35,PMQ,,-NA-,-NA-,INSPECT_NO_DAYS,2014-12-02
00:51:35,ACTUAL,INSPECT_NO_DAYS,2000,-NA-,1
4,PARAMETERVI,,-NA-,PP9-XX9-009,-NA-,2014-12-02 00:51:35,2014-12-02 00:51:35,2014-12-02
00:51:35,PMQ,,-NA-,-NA-,INSP_PROB0,2014-12-02 00:51:35,ACTUAL,INSP_PROB0,0.99,-NA-,1
5,PARAMETERVI,,-NA-,PPB-XXY-003,-NA-,2014-12-02 00:51:35,2014-12-02 00:51:35,2014-12-02
00:51:35,PMQ,,-NA-,-NA-,INSP_LAM0,2014-12-02 00:51:35,ACTUAL,INSP_LAM0,5,-NA-,1

```

The example shows that the measurement type contains the parameter name. The parameter values are loaded into the Profile_Parameter table.

The orchestration definition file for inspection parameter events has a single orchestration step. For events with the PARAMETERVI event type code and the

value type code ACTUAL, ASSIGN and ASSIGN_DATE calculations are configured, and the profile adapter persists the calculation results to the PROFILE_PARAMETER table.

The next step that allows IBM Predictive Maintenance and Quality to perform the inspection is to store the events that are related to production and inspection. The events for inspection can be in the form of runtime or batch data. Runtime data is raw time series data and batch data is data that is aggregated by day, month, and other units of time. The events are stored in time series tables.

EVENT table

Contains information for master entities that is related to the event, for example, production batch, process, material, and resource.

EVENT_OBSERVATION table

Contains information that is related to the core event, for example, measurement, time of occurrence, and type of event.

Inspection and production events are processed through Predictive Maintenance and Quality Eventload message flows according to the orchestration definition file PMQ_orchestration_definition_inspection.xml.

The orchestration definition file for inspection events has a single orchestration step. For events with the INSPECT and FAIL measurement type code, the TOTAL calculation is configured, and the profile adapter persists the calculation results to the PRODUCT_KPI table.

The orchestration definition file for production events has a single orchestration step. For events with the QTY measurement type code, the TOTAL calculation is configured, and the profile adapter persists the calculation results to the PRODUCT_KPI table.

Event format for inspection loading

Inspection data that consists of production events to report the quantity that is produced, and inspection events to report the inspected and failed products, are loaded as Predictive Maintenance and Quality events.

The classification is based on the event type and the measurement.

For a PRODUCTION event type, the measurement type must be quantity (QTY), and the measurement holds the quantity value.

For an INSPECTION event type, the measurement type must be either INSPECT or FAIL.

- With the INSPECT measurement type, the number of products that underwent inspection is the measurement.
- With the FAIL measurement type, the number of products that failed inspection is the measurement.

The event type and the measurement type must be the key. Other columns that are used are production_batch_code, the location code, the event_start_time, observation_timestamp, and value_type_code. The event_start_time and observation_timestamp indicates the date and time of inspection.

Note: Each PRODUCTION event is followed by two INSPECTION events. Each INSPECTION event has the value 1 and 2 for multirow_no. The INSPECTION events must be in sequence and are not considered as a complete event unless both are included. An INSPECT measurement type must have one more INSPECTION event with FAIL measurement type to complete the action.

Inspection message flow and triggering mechanism

QEWS has two triggering modes, timer-based triggering and file-based triggering.

In the timer-based triggering mode, the QEWS invocation adapter is triggered at the configured time in the batch orchestration file `PMQ_orchestration_definition_batch.xml` once a day.

In the file-based triggering mode, a file with the run date as input is placed in the `batchdatain` directory, and the QEWS invocation adapter is triggered.

The timer-based and file-based trigger flows call the Process Inspection flow, which is shown in the following figure.

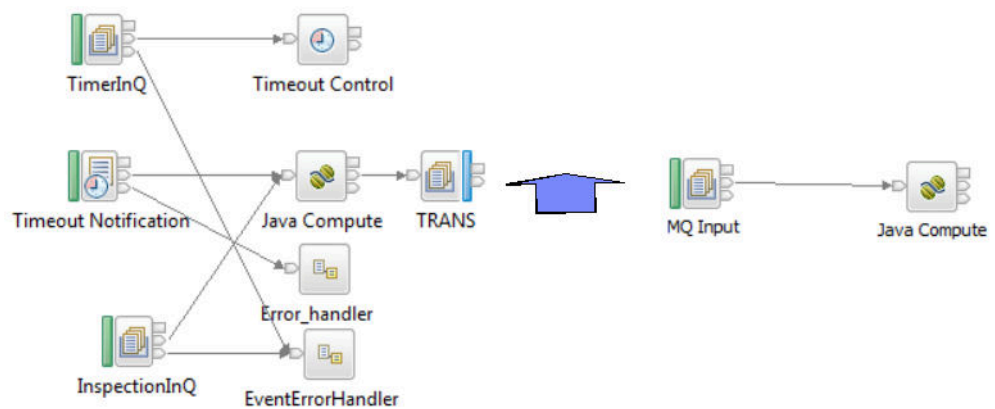


Figure 21. Process Inspection flow

The process inspection flow queries the `PRODUCT_KPI` table, and groups the combinations of masters that are referenced with tenant information that is filtered by the run date. Each combination is passed as a record to the TRANS queue.

The Java Compute node that is listening to the TRANS output queue picks up each combination message and retrieves the records from the KPI table, filtering the records by run date, `batch_flag (N)`, and the master combination. The records are then fed to the QEWS algorithm.

The invocation adapter also queries the `PROFILE_PARAMETER` table, and loads the parameters for the combination of masters (Product, Process, Material, Location, and Resource). The invocation adapter passes the set of parameters and the inspection data to the QEWS algorithm. The QEWS algorithm analyzes and persists the data to the KPI and Profile tables.

Output and reporting

Inspection analysis output is added to the `PRODUCT_KPI` and `PRODUCT_PROFILE` tables. The table structure includes the `Rundate` column. The

historical run analysis outputs in the table remain the same as in previous IBM Predictive Maintenance and Quality releases, and no purging policy is defined. The purging policy might be defined by the business requirements. In addition to the Rupdate column, the columns for referring to master entities like Process, Material, Resource, and Location are added to these two tables. These columns are included to group or not group a product, based on reasons such as the material that was used to prepare it, the resource that it uses, and the process that is used to prepare it.

IBM Cognos Business Intelligence includes the Rapidly Adaptive Visualization Engine (RAVE) that is used to build the Failure Rate chart and the Evidence chart. Cognos BI queries the PRODUCT_KPI and PRODUCT_PROFILE tables based on the run date value, and gathers the records that fit the run date value. The records are populated into a .json file at run time, and the .json file is used to prepare the charts.

Results and benefits

The quality early warning system (QEWS) in IBM Predictive Maintenance and Quality reduces cost by detecting problems and issues earlier and more accurately.

Results

Predictive Maintenance and Quality QEWS provides the following results:

- Improves production yields in the manufacturing line.
- Helps you to gain better understanding of root causes of manufacturing issues.
- Provides faster detection of manufacturing quality problems.

Benefits

Subtle changes in failure rates that indicate potential emerging quality problems are detected earlier. Early detection means quicker problem identification, faster problem resolution, and lower total costs.

The definitive nature of QEWS alerts eliminates the need for subjective judgment of statistical process control charts and other traditional tools, providing you with consistent and accurate direction.

QEWS can deliver insightful early warning signals even under variable lot size scenarios.

Warranty

Various conditions can lead to accelerated wear and replacement of manufactured products that are under warranty. Such conditions might include variations in the manufacturing process of the product, variations in the quality of vendors' materials that are used in the product, or the ways in which the product is used.

A small delay in detecting the conditions that lead to accelerated wear can cause more warranty claims and related losses. By understanding the factors that lead to warranty claims, you can take corrective actions such as the following actions:

- Improve manufacturing processes to prevent warranty claims.
- Set pricing for warranties and extended warranties.
- Evaluate vendors of the materials that are used in the product.

The quality early warning system (QEWS) warranty use case in IBM Predictive Maintenance and Quality provides detection that is based on excessive replacement rate and evidence of wear-out.

Replacement rate

QEWS alerts you when the product's random failure rate exceeds a computed threshold. The threshold can reflect product reliability goals (for example, the product population in the field must not exceed a specified failure rate) or financial liability goals (for example, the cost of reimbursing product warranty claims must not exceed a specified total amount).

Wear-out

QEWS alerts you when it finds evidence that product failures are not random in time, but are indicative of wear-out. Wear-out means that products that are in customer use for a longer time fail more often than products that are in customer use for a shorter time. Because wear-out can have serious consequences, QEWS alerts you when it detects evidence of wear-out regardless of how many product units contributed to the detection.

QEWS enables warranty models that are based on sales, production, and manufacturing dates.

Sales model

The Sales model identifies variations in product wear and replacement rates according to the date of sale. The date of sale might correlate with in-service conditions, seasonal climatic conditions, a particular customer, or other important similarities.

For example, a product carries a one-year warranty. In cold conditions, the product becomes brittle and wears prematurely. In certain geographies, products that are sold and enter service in winter initially suffer rapid wear, followed by slower wear during the latter part of the warranty period. The opposite is true for products that are sold and enter service in summer. These seasonal variations affect the product wear rates and weighted replacement rates, which are detected early by QEWS.

Production model

The Production model identifies variations in product wear and replacement rates according to the production date of the product, not the resource in which the product is used. The production date of the product might correlate with the manufacturing equipment operator, the manufacturing process, or other important similarities.

For example, a faulty batch of products is produced during a particular period. The products are installed in resources that have different manufacturing dates. Although the resource manufacturing dates and the product production dates are unrelated, QEWS makes it easier to identify and understand the real cause of the warranty claims.

Manufacturing model

The Manufacturing model identifies variations in product wear and replacement rates according to the manufacturing date of the resource in which the product is used. The resource manufacturing date might correlate with assembly problems that occurred during a particular period.

For example, due to a short-term problem with the manufacturing process of a resource, some of the products that are used in the resource fail

prematurely. Although the resource manufacturing dates and the product production dates are unrelated, QEWS makes it easier to identify and understand the real cause of the warranty claims.

You can adjust the frequency at which data is captured and input to QEWS, and the frequency at which QEWS analyses are run, according to the requirements of each situation. For example, monitoring data from a network of field service personnel might best be done on a monthly basis.

Business and technical challenges

Fast product cycles, high product volumes, and increasing cost pressure can all lead to increasing numbers of released defective products. The quality early warning system uses IBM technology to detect warranty claim trends earlier than they would otherwise be detected so that you can intervene with corrective action.

Business challenges

Statistical process control methods often overlook cumulative evidence that indicates a worsening quality problem. Better analytical methods are often difficult to implement due to complex computational challenges and constraints in software implementation.

Technical challenges

Premature product wear can have non-obvious causes such as source material variations, seasonal climatic conditions, or temporary manufacturing problems either with the product or with the resource in which the product is used. A small delay in detecting the conditions that lead to accelerated wear can cause more warranty claims and related losses.

Defining the warranty solution

To define the warranty solution, you must load master data, load event data, define message flows, and define the output location of the warranty analysis.

Procedure

1. Load master data. For more information about loading master data, see Chapter 4, “Master data,” on page 21.
2. Load event data. You can load event data in batch mode or in real time. For more information about loading event data, see Chapter 5, “Event data,” on page 49.
3. Define message flows. For more information about message flows, see “Message flows” on page 13.

Results

IBM Cognos Business Intelligence generates the warranty dashboards and reports.

Warranty solution details

There are requirements that you must consider when loading the master data and event data tables.

Master data tables are loaded by the master flows. The following tables are required to implement a warranty use case:

Master_Location

The Master_Location table contains information that is specific to the geography of the location where the event is produced, or the resource that produces the events.

The following text is an example of a CSV file that is used to load the Master_Location table:

```
location_cd,location_name,region_cd,region_name,country_cd,
country_name,state_province_cd,state_province_name,city_name,
latitude,longitude,language_cd,tenant_cd,is_active
Tokyo,Tokyo,AP,Asia Pacific,JP,Japan,TY,Tokyo,TokyoCity,35.41,139.45,
EN,PMQ,1
```

Master_Resource_Type

The Master_Resource_Type table maintains the Resource type classification. It supports two classification types: ASSET and AGENT. ASSET is a machine or machine part that is used in production. AGENT is the one who operates the machine or the system to ensure that the production procedure is carried out properly.

The following text is an example of a CSV file that is used to load the Master_Resource_Type table:

```
resource_type_cd,resource_type_name,language_cd,tenant_cd
ASSET,Asset,EN,PMQ
AGENT,Agent,EN,PMQ
```

Master_Resource

The Master_Resource table maintains all of the details pertaining to a Resource (ASSET or AGENT). The table maintains information such as which organization hierarchy the resource is lined to, the location at which the resource is installed, the tenant to which the resource is attached or leased, production rate, maintenance interval, and the manufactured date of the resource.

The following text is an example of a CSV file that is used to load the Master_Resource table:

```
resource_cd1,resource_cd2,resource_name,resource_type_cd,
resource_sub_type,parent_resource_cd1,parent_resource_cd2,
standard_production_rate,production_rate_uom,
preventive_maintenance_interval,group_type_cd_1,
group_member_cd_1,group_type_cd_2,group_member_cd_2,
group_type_cd_3,group_member_cd_3,group_type_cd_4,
group_member_cd_4,group_type_cd_5,group_member_cd_5,
location_cd,mfg_date,language_cd,tenant_cd,Is_active
-NA,-NA-,Not Applicable,ASSET,,,,,-NA,-NA,-NA,-NA,-NA-,
-NA,-NA,-NA,-NA,-NA,-TK,2014-01-01,EN,PMQ,1
RCD1,MOD1,RCMOD1,ASSET,,,,,,,,,,,,,TK,,,1
RCD2,MOD2,RCMOD2,ASSET,,,,,-NA,-NA,-NA,-NA,-NA-,
-NA,-NA,-NA,-NA,-NA,-TK,,,1
RCD3,MOD3,RCMOD3,ASSET,,,,,-NA,-NA,-NA,-NA,-NA-,
-NA,-NA,-NA,-NA,-NA,-TK,,,1
```

Master_Product

The Master_Product table contains the core data for the inspection and warranty use cases. This table stores information that is related to a product and the product_type.

The following text is an example of a CSV file that is used to load the Master_Product table:

```

product_cd,product_name,product_type_cd,product_type_name,
language_cd,tenant_cd,Is_active
AAA,TRUNK,B005,Body,EN,PMQ,1
AAB,TRUNK,B005,Body,EN,PMQ,
AAC,TRUNK,B006,Body,EN,PMQ,
AAD,TRUNK,B006,Body,EN,,
AAE,TRUNK,B006,Body,,,

```

Master_Production_Batch

The Master_Production_Batch table contains information about each production batch that is used to produce a product. The information includes the product that is produced, the date it is produced, and the batch information.

The following text is an example of a CSV file that is used to load the Master_Production_Batch table:

```

production_batch_cd,production_batch_name,product_cd,
product_type_cd,produced_date,language_cd,tenant_cd
B1001,FrameBatch,AAA,B005,2012-03-01,EN,PMQ
B1002,FrameBatch,AAB,B005,2012-03-01,EN,PMQ
B1003,FrameBatch,AAC,B006,2012-03-01,EN,PMQ
B1004,FrameBatch,AAA,B006,,,

```

Master_Resource_Production_Batch

The Master_Resource_Production_Batch table contains information about each production batch that is used to produce a resource.

The following text is an example of a CSV file that is used to load the Master_Resource_Production_Batch table:

```

resource_cd1,resource_cd2,production_batch_cd,qty,language_cd
RCD1,MOD1,B005,3,EN
RCD2,MOD2,B006,3,EN
RCD3,MOD3,B005,3,EN

```

Tip:

- If a product can have different parameters (such as LAM0, LAM1, PROB0, CW0, CW1, PROBW0), then you can assign a separate product code and production batch to each product variation. Reference each production batch in the Master_Resource_Production_Batch table.
- If a product has the same parameters but different manufacturing or production dates, then you can assign a separate production batch for each manufacturing or production date. Reference each production batch in the Master_Resource_Production_Batch table.

Master data in the Sales model

The following statements apply to the Sales model:

- When a resource is sold, the warranty is tracked from the date of sale until the end of the warranty period. Resources are tracked because, unlike products, in IBM Predictive Maintenance and Quality resources are serialized and can form a hierarchy.
- Each resource contains a number of products. Each product is tracked by a Master_Production_Batch table record.
- The Master_Resource_Production_Batch table handles the mapping between the Master_Resource and Master_Production_Batch tables and also maintains the quantity of products that go into a resource.

Master data in the Production model

The following statements apply to the Production model:

- The warranty for a product extends from the date of production until the end of the warranty period.
- Products are tracked by produced_date.
- The product produced_date is stored in the Master_Production_Batch table and is used as the vintage date.

Master data in the Manufacturing model

The following statements apply to the Manufacturing model:

- The warranty for a resource extends from the date of manufacture until the end of the warranty period.
- Resources are tracked by mfg_date.
- The mfg_date is stored in the Master_Resource table.

Master data and metadata loading

When parameter events are loaded, the event type PARAMETERVW is used. For warranty data events, the event types SALES and WARRANTY are used.

In addition to the measurements that are used in warranty events, measurements with parameter names must also be loaded.

For every measurement type, a unique profile variable is defined so that the Foundation orchestration engine can be leveraged to load parameters to the profile parameter table with multiple master grain levels supported. For parameters loading, profile variables are defined with the profile variable code ParameterName and with the ASSIGN profile calculation configured.

The following figure shows an example of a CSV file that contains the measurement type.

```
1 measurement_type_cd,measurement_type_name,unit_of_measure,carry_forward_indicator,aggregation_type,event
  _code_indicator,language_cd,tenant_cd
2 WARR_LAMO,Warranty Acceptance Level,,0,SUM,0,EN,PMQ
3 WARR_LAML,Warranty Unacceptance Level,,0,SUM,0,EN,PMQ
4 WARR_PROB0,Warranty Confidence Probability Level,,0,SUM,0,EN,PMQ
5 WARR_CW0,Warranty Wearout Acceptance Level,,0,SUM,0,EN,PMQ
6 WARR_CW1,Warranty Wearout Unacceptance Level,,0,SUM,0,EN,PMQ
7 WARR_PROBW0,Warranty Wearout Confidence Probability Level,,0,SUM,0,EN,PMQ
8
```

Figure 22. Example of a CSV file showing the measurement type

The following figure shows an example of a CSV file that contains the parameters for each product.

```

1 profile_variable_cd,profile_variable_name,profile_calculation_name,measurement_type_cd,resource_type_cd,
material_type_cd,profile_units,comparison_string,low_value_date,high_value_date,low_value_number,high_v
lue_number,kpi_indicator,profile_indicator,data_type,aggregation_type,carry_forward_indicator,process_in
dicator,variance_multiplier,language_cd,tenant_cd
2 WARR_LAM0,Warranty Acceptance Level,ASSIGN,WARR_LAM0,ASSET,-NA,,,,,,,,,1,1,INT,,0,0,1,EN,PMQ
3 WARR_LAM1,Warranty Unacceptance Level,ASSIGN,WARR_LAM1,ASSET,-NA,,,,,,,,,1,1,INT,,0,0,1,EN,PMQ
4 WARR_PROB0,Warranty Probability 0,ASSIGN,WARR_PROB0,ASSET,-NA,,,,,,,,,1,1,INT,,0,0,1,EN,PMQ
5 WARR_CW0,Warranty Wearout Acceptance Level,ASSIGN,WARR_CW0,ASSET,-NA,,,,,,,,,1,1,INT,,0,0,1,EN,PMQ
6 WARR_CW1,Warranty Wearout Unacceptance Level,ASSIGN,WARR_CW1,ASSET,-NA,,,,,,,,,1,1,INT,,0,0,1,EN,PMQ
7 WARR_PROBW0,Warranty Wearout Confidence Level,ASSIGN,WARR_PROBW0,ASSET,-NA,,,,,,,,,1,1,INT,,0,0,1,EN,PMQ
8

```

Figure 23. Example of a CSV file that contains parameters for each product

Event data loading

Warranty uses two types of event data: parameter data and sale and warranty data.

Predictive Maintenance and Quality parameters and their respective combinations covering the master entities Resource, Location, Product, Production Batch, Material, and Process are mapped to Predictive Maintenance and Quality events, and loaded to the PROFILE_PARAMETER table by the orchestration engine using the PMQEventLoad flow, based on the Master_Profile_Variable definition and orchestration definition. Parameter values are stored in the PARAMETER_VALUE column in the PROFILE_PARAMETER table, along with the profile variable and master data references mapped in the event.

The following figure shows an example of a CSV file that is used for loading parameter events.

```

1 incoming_event_cd,event_type_cd,source_system,process_cd,prod_batch_cd,location_cd,event_start_time,even
t_end_time,event_planned_end_time,tenant_cd,operator_cd,model,serial_no,measurement_type_cd,observation
timestamp,value_type_cd,observation_text,measurement_material,material_code,multirow_no
2 1,PARAMETERVW,,,PP9-XX9-009,,2014-10-08 00:00:00,,,PMQ,,,,,WARR_LAM0,2014-10-08 00:00:00,ACTUAL,,0.003,,1
3 1,PARAMETERVW,,,PP9-XX9-009,,2014-10-08 00:00:00,,,PMQ,,,,,WARR_LAM1,2014-10-08
00:00:00,ACTUAL,,0.05722,,1
4 1,PARAMETERVW,,,PP9-XX9-009,,2014-10-08 00:00:00,,,PMQ,,,,,WARR_PROB0,2014-10-08 00:00:00,ACTUAL,,0.95,,1
5 1,PARAMETERVW,,,PP9-XX9-009,,2014-10-08 00:00:00,,,PMQ,,,,,WARR_CW0,2014-10-08 00:00:00,ACTUAL,,1,,1
6 1,PARAMETERVW,,,PP9-XX9-009,,2014-10-08 00:00:00,,,PMQ,,,,,WARR_CW1,2014-10-08 00:00:00,ACTUAL,,1.2,,1
7 1,PARAMETERVW,,,PP9-XX9-009,,2014-10-08 00:00:00,,,PMQ,,,,,WARR_PROBW0,2014-10-08
00:00:00,ACTUAL,,0.99,,1

```

Figure 24. A CSV file that is used for loading parameter events

The orchestration definition XML file for parameter events has a single orchestration step. For events with the event type PARAMETERW and the value type code ACTUAL, the ASSIGN and ASSIGN_DATE calculations are configured, and the Profile adapter persists the calculation results into the PROFILE_PARAMETER table.

After the Master data load flows are completed, you must load the event flows. Event data is loaded on an event basis, where each event is associated with a number of observations. Each observation indicates a measurement type (for example, pressure in kilopascals) and a measurement reading.

The event flows load events such as SALES and WARRANTY that are predefined in the Master_Event_Type table. Every event is related to a particular Resource and the Production_Batch details.

The orchestration definition file for sales and warranty events, PMQ_orchestration_definition_warranty.xml, has a single orchestration step.

Events with the event type SALES and WARRANTY are persisted into the EVENT and EVENT_OBSERVATION tables by the EventStore adapter.

Event data loading in the Sales model

The Sales model event data is loaded in the following order:

1. The SALES event is loaded.
 - The measurement_type_cd field contains SALESDATE.
 - The event_start_time field and the observation_timestamp field contain the date of sale.
 - The observation_text field contains the warranty end date. By default, the value is three years but it can be changed as required.
 - The measurement field contains the number of warranty months.
2. Any number of WARRANTY events are loaded.
 - The measurement_type_cd field contains WARRANTYINDICATOR.
 - The event_start_time field and the observation_timestamp field contain the date when the claim was made.
 - The observation_text field and the measurement field are blank.

The following text is an example of a CSV file that is used for loading sales events.

```
incoming_event_cd,event_type_cd,source_system,process_cd,
prod_batch_cd,location_cd,event_start_time,event_end_time,
event_planned_end_time,tenant_cd,operator_cd,resource_cd2,
resource_cd1,measurement_type_cd,observation_timestamp,
value_type_cd,observation_text,measurement,material_code,multirow_no
1,SALES,,B1001,Tokyo,2006-12-19T12:00:00,,PMQ,,MOD1,RCD1,
SALESDATE,2006-12-19T12:00:00,ACTUAL,12/19/2009,35.9344262295082,,1
1,WARRANTY,,B1001,Tokyo,2013-06-17T12:00:00,,PMQ,,MOD1,RCD1,
WARRANTYINDICATOR,2013-06-17T12:00:00,ACTUAL,N,,1
1,SALES,,B1002,Tokyo,2006-11-20T12:00:00,,PMQ,,MOD2,RCD2,
SALESDATE,2006-11-20T12:00:00,ACTUAL,11/20/2009,35.9344262295082,,1
1,WARRANTY,,B1002,Tokyo,2009-05-04T12:00:00,,PMQ,,MOD2,RCD2,
WARRANTYINDICATOR,2009-05-04T12:00:00,ACTUAL,Y,,1
1,SALES,,B1003,Tokyo,2006-10-31T12:00:00,,PMQ,,MOD3,RCD3,
SALESDATE,2006-10-31T12:00:00,ACTUAL,10/31/2009,35.9344262295082,,1
```

Event data loading in the Production model

The Production model event data is loaded in the following order:

1. The SALES event is loaded.
 - The measurement_type_cd field contains SALESDATE.
 - The event_start_time field and the observation_timestamp field contain the Produced Date from the Master_Production_Batch table.
 - The observation_text field contains the warranty end date. By default, the value is 3 years but it can be changed as required.
 - The measurement field contains the number of warranty months.
2. Any number of WARRANTY events are loaded.
 - The measurement_type_cd field contains WARRANTYINDICATOR.
 - The event_start_time field and the observation_timestamp field contain the date when the claim was made.
 - The observation_text field and the measurement field are blank.

Event data loading in the Manufacturing model

The Manufacturing model event data is loaded in the following order:

1. The SALES event is loaded.
 - The measurement_type_cd field contains SALESDATE.
 - The event_start_time field and the observation_timestamp field contain mfg_date from the Master_Resource table.
 - The observation_text field contains the warranty end date. By default, the value is 3 years but it can be changed as required.
 - The measurement field contains the number of warranty months.
2. Any number of WARRANTY events are loaded.
 - The measurement_type_cd field contains WARRANTYINDICATOR.
 - The event_start_time field and the observation_timestamp field contain the date when the claim was made.
 - The observation_text field and the measurement field are blank.

Warranty triggering

There are two warranty triggering modes, timer-based triggering and file-based triggering.

In the timer-based triggering mode, the warranty IBM SPSS job is triggered at the scheduled time once a day, as configured in the batch orchestration file `PMQ_orchestration_definition_batch.xml`, with the current date as the run date. The default sub use case is Sales.

The Batch integration AutoTrigger flow accepts input parameters such as the sub use case name from the scheduler configuration (in the XML file), in addition to the schedule time, the queue name, and the duration. The AutoTrigger flow places the timer request to the queue that is configured to trigger the SPSSJobIntegration flow, which in turn triggers the SPSS job at the scheduled time, using the configurations and parameters defined in the batch orchestration.

In the file-based triggering mode, the warranty SPSS job is triggered by placing the run date file in the `batchdatain` directory with the run date and the sub use case parameter. The WarrantyDataPreparation flow accepts the run date file, and places an MQ request in the `PMQ.JOBINTEGRATION.IN` queue to trigger the SPSSJobIntegration flow, which in turn triggers the SPSS job using the configurations and parameters defined in the batch orchestration.

The following figure shows the batch orchestration configuration for warranty. Configurations can be modified at run time.

orchestration	
Identifier	SALES
scheduler	
scheduled_time	00:00:00
queue_name	PMQ.QEWS.WSTIMER.IN
duration_in_days	1
webservice	Webservice configuration for Warranty SALES
url	http://9.122.126.168:9080/process/services/ProcessManagement
jobLocationURI	spssc:///?id=5691007b90f455850000014a28e6e3bc939a
parameters	
parameter	
name	RunDateInFormatYYYYMMDDHyphenSeparated
value	StartDate
type	dynamic
parameter	
name	ServiceTabQtyMultiplier
value	1
type	static
parameter	
name	IsRunDateEqServerDate
value	0
type	static
parameter	
name	SubUseCase
value	SALES
type	static
notificationEnabled	true

Figure 25. Batch configuration for the warranty use case

Data in the event and event observation tables must be processed so that it can be provided to QEWS. Processing the tables involves calling the SPSS modeler stream, which picks data from Event, Event_Observation, Resource, Product, and Production_Batch and prepares the data in the following format:

```
Product_id | Produced Date | Service_Month | Parts under Warranty | Parts
replaced | tenant_cd | Use Case | RunDate
```

A Service table holds these records and forms as input from where the QEWSL invocation algorithm picks the data.

Once the SPSS modeler stream completes the transformation of Masters and Events into Service details, the QEWS invocation flow is triggered.

SPSS sends a status file with the run date to the integrationin directory of the ESB node, and the file is processed by the WarrantyFileIntegration flow. When the status of the warranty job is SUCCESS, it triggers the ProcessWarranty flow by placing a message in the input queue of ProcessWarranty.

The ProcessWarranty flow consumes the run date from the Status message, and queries the Service table in the Predictive Maintenance and Quality data mart. The flow prepares object structures comprised of the Parts under warranty (WPARTS), Parts replaced under warranty (WREPL), Produced Date, product_id, and the Product Parameters.

SPSS Modeler Streams

There are two SPSS Modeler streams and corresponding Collaboration and Deployment Services jobs for Warranty. The first stream is for the Manufacturing and Production models, in which the specific use case can be controlled by toggling across a parameter from MFG (Manufacturing) to PROD (Production). The second stream is for the Sales model.

The streams differ in the transformation logic for producing the Service table (for more information, see “Service tables”). The SPSS Modeling layer provides special logic for each of the models; all other processing and treatment is the same for all of the models.

The main difference between the models is in the aggregation and tracking of vintages. A vintage is a combination of the product ID (numbered product type) and a date (date of sale, date of production, or date of manufacture). The date at which the product was put in service is assumed to be the same as the date of sale of the resource in which the product is used. The models do take into account the differential tracking and treatment of products that are sold or shipped as replacements of other products that were shipped separately. Replacement products can either be excluded from the event structure or they can be included as a separate vintage.

You can choose between the Production and Manufacturing models by changing the `IsMFG_OR_PROD` variable of the `IBM_QEWSL_JOB` C&DS job to `PROD` or `MFG`. You can change the variable from either SPSS Collaboration and Deployment Services (during ad-hoc one time trigger) or IIB (during automated triggers).

The Sales model is controlled by a separate job named `IBMMPMQ_QEWSL_SALES_JOB`. The job can be run from IIB by using its job URI.

Customizable Parameters and Special Scenarios

Both SPSS Modeler streams contain some common parameters that can be used while running the SPSS models under special scenarios and requirements. These options can be altered from the SPSS Collaboration and Deployment Services Job Variable or from IIB. The preferred way of altering these parameters is through IIB. The description and uses of these parameters is as follows:

IsRunDateEqServerDate

This parameter determines whether the SPSS server system date (value = 1) or a custom run date (value = 0) is used in computation logic requiring a run date. The default value is 0 and it uses the custom run date that is supplied by IIB (corresponding to the IIB server system date during default runs).

RunDateInFormatYYYYMMDDHyphenSeparated

This parameter is used only if the value of `IsRunDateEqServerDate` parameter is 0. The parameter sets the custom run date. The required date format is `YYYY-MM-DD`.

ServiceTabQtyMultiplier

For performance reasons, sometimes you might need to run the `QEWSL` warranty engine on a sample of the complete data. `QEWSL` is a weighted algorithm, so by default it does not produce the same graphs or alerts for a sample as it would for the complete data. If the sample is a good true representative sample, this parameter helps to correct the scale of the weighted results or plots to give a representative output. The parameter is set with a value of multiplier as $1/number$.

Service tables

When the SPSS stream runs, it populates a DB2® table named `PMQSCH.SERVICE` (referred to as the Service table). After the table is populated, the processing is the same for all models.

The structure of the Service table is the same for all models. What changes is the computation and aggregation logic for the table fields by the different SPSS streams and models.

The Service table contains the following fields:

PRODUCED_DATE

This field contains the vintage date of the Sales or Manufacturing model. Together with the PRODUCT_ID field, this field represents the vintage of the record. Together with PRODUCT_ID and SVC_MNTHS fields, this field represents the composite unique key for the table.

PRODUCT_ID

This field represents the non-serialized product identifier (numeric product type) for the product whose replacement needs to be tracked.

SVC_MNTHS

This field represents the number of months that any of the products of that vintage (PRODUCED_DATE + PRODUCT_ID) were in service during their warranty period. For example, a three-year warranty period can contain up to 36 service months.

To have a consistent number of maximum service months across vintages in a computation lot, products with shorter warranty periods (for example, two years) can be given more SVC_MNTHS to match products with longer warranty periods (for example, 36 months). In this case, during the SVC_MNTHS that are outside of the warranty period, WPARTS and WREPL are both 0.

WPARTS

This field represents the number of products of that vintage (PRODUCED_DATE + PRODUCT_ID) that were in service without any warranty claims during the service month (SVC_MNTHS).

WREPL

This field represents the number of products of that vintage (PRODUCED_DATE + PRODUCT_ID) that failed (received a warranty claim) during the service month (SVC_MNTHS).

TENANT_ID

This field is an identifier to differentiate between tenant data in a multi-tenant environment.

Warranty message flow and triggering mechanism

When the SPSS modeler stream runs successfully, it invokes the warranty flow. A status message that is embedded with a date value is placed in the PMQ.QEWS.WARRANTY.IN queue. When the broker interface detects a message in the queue, it triggers the QEWSL algorithm. The embedded date value in the message is the rundate, which becomes the date of reference for the warranty flow. The Service table records and the parameters are passed to the QEWSL algorithm.

The same message flow is used to trigger all of the warranty models.

Output and reporting

The output from the algorithm is persisted in the Lifetime_KPI and Lifetime_Profile tables. In these tables, along with the analysis output, there are columns for Rundate and UseCase. In addition, columns that refer to the master

entities Resource, Process, Material, and Location are added to these two tables. Currently, the references to the master entities are not used. They refer to the NA rows for the default language.

The Replacement rate chart and the evidence charts are prepared by the Rapidly Adaptive Visualization Engine (RAVE) in IBM Cognos Business Intelligence, and they extract the records from the Lifetime KPI and Profile tables for a given rundate and use case.

Results and benefits

The quality early warning system (QEWS) warranty use case in IBM Predictive Maintenance and Quality reduces cost by detecting problems and issues earlier than they would otherwise be detected, and more accurately.

Results

IBM Predictive Maintenance and Quality QEWS delivers the following results:

- Shows you where to improve manufacturing processes to prevent warranty claims.
- Helps you to set pricing for warranties and extended warranties.
- Helps you to evaluate vendors of the materials that are used in products.

Benefits

Subtle changes in warranty claim rates indicative of potential emerging quality problems are detected earlier. This allows for quicker problem identification, faster problem resolution, and lower total costs.

The definitive nature of QEWS alerts eliminates the need for subjective judgment of statistical process control charts and other traditional tools, providing you with consistent and accurate direction.

QEWS can deliver insightful early warning signals even under variable lot size scenarios.

Parametric

The quality early warning system (QEWS) parametric use case in IBM Predictive Maintenance and Quality detects unfavorable changes in variable-type data, and provides information that facilitates diagnostics and alarm prioritization.

The QEWS parametric use case uses the Quality Early Warning System for Variable Data (QEWSV) algorithm to monitor variable-type data. This type of data is found in a number of industrial applications, including Supply Chain, Manufacturing, and Finance applications. QEWSV identifies unfavorable trends in the data process. The focus is on providing timely detection of unacceptable process behavior while maintaining a pre-specified low rate of false alarms.

Predictive Maintenance and Quality maintains variable data in the event store, and prepares data for the QEWSV algorithm. Variable values and evidence charts are plotted by using parametric results.

The QEWSV system is organized around three basic notions that are referred to as Variables, Operations, and Time slides. The behavior of selected variables is monitored based on acceptable and unacceptable characteristics of the underlying

process. These characteristics are converted to rules of the decision-making scheme that is used to decide whether a variable is flagged. Operations refer to points in the process that have a potential of influencing the stochastic behavior of the variables and are thus considered as likely problem areas when variables show unacceptable behavior. Time slides are data structures that organize measurements that pertain to a particular variable regarding an operation of interest.

Mapping

In Predictive Maintenance and Quality, Operation is synonymous to Process. Operation is the sequence of flow that is involved in arriving at the end product or intermediate product. Additionally, a tool is also considered a factor in identifying the behavior. In Predictive Maintenance and Quality, the tool is considered the Resource.

Time slides deal with the interval of time on which the measurement or observation on a variable is taken. The observation time stamp is more synonymous with time slides when an event observation is made.

Variables are defined for every operation per tool. In Predictive Maintenance and Quality, Variables are equated with Measurement type, whose measurements are read at different time intervals during the sequence of the operation flow.

Analysis performed in parametric use case

Based on observations with a specific measurement type along the time slides, the deviation or drift from targeted values are computed and analyzed to show whether the sequence of process is adhering to normal operation limits. This analysis impacts the quality of the end or intermediate product.

Sub use cases

Predictive Maintenance and Quality can handle a varied set of master data sets, from end products to manufacturing machinery to the raw materials used, as well as the environment or location-specific data. Predictive Maintenance and Quality identifies the following categories of analysis that is done on each of the master entities. Some categories might be a combination of different masters or a lone entity.

Process resource validation

This category is the default use case, where the process and the resource that takes part in the process is monitored based on a defined set of variables. These variables are associated with a set of parameters that define the target values, acceptable limit, unacceptable limit, standard deviation, false alarm rate, and unacceptable factor.

Resource validation

A resource is monitored based on the standard operation limits across a few measurement types (variables). This type of health check is essential in identifying any issues in the resource and correcting those issues to improve the performance and throughput.

Product validation

With Quality inspection, the product as a whole is checked, based on the failure rate. In variable data, given the set of variables whose targets are set for the product to meet, any deviation or drift beyond the allowed deviation highlights a flaw in the product.

Material validation

Raw materials purchased from a vendor are monitored for a defined set of guidelines as variables, and validated to check on the quality of the procured material.

Location suitability

With variable analysis, a location is analyzed to see whether it is suitable for a particular operation. Variables like pressure, temperature, humidity, and their time slide values can forecast the suitability of a location for carrying out any operation.

Each validation supports grains of Resource, Process, Material, Product, and Location. Given a combination of grains, a variable can be declared and provided with a target set of parameters to validate.

Business and technical challenges

The parametric use case has business and technical challenges.

Business challenges

The business challenges lie in identifying the rules to set up the quality norms for an end product or a raw material. When a rule is defined, if it does not identify a flaw in a product or material, it leads to noise and turbulence in the confidence of the quality. When there are more quality-related issues, there is more damage to the business, and the cost that is incurred in replacements and servicing is greater.

Applying complex statistical computation is a difficult task, and hard to implement with market available packages.

Technical challenges

Identifying a quality defect is difficult, unless the defect is treated under different variable conditions. Usually, in the quality inspection process, the quality is defined by a set of rules. However, defining those rules cannot identify subtle variations that occur during the manufacturing process. Therefore, treating defects under various conditional checks across different measurements, based on a target value, helps in forecasting the type of flaw that might cause a product to fail. Implementing conditional checks is a difficult task, as it requires complex statistical procedures.

Defining the parametric solution

To define the parametric solution, you must load master data, load event data, define message flows, and define the output location of the parametric analysis.

Procedure

1. Load master data. The master data includes master records for Process, Resource, Product, Material, and Location. For more information about loading master data, see Chapter 4, “Master data,” on page 21.
2. Load metadata. Metadata includes Measurement type (Variable), Event type, and Profile variable data.
3. Load event data. You can load event data in batch mode or in real time. Event data includes parameter data and measurement observations at every time scale for each defined event type. For more information about loading event data, see Chapter 5, “Event data,” on page 49.

Results

IBM Cognos Business Intelligence uses data in the PARAMETRIC_KPI and PARAMETRIC_PROFILE tables to generate the parametric dashboard and reports.

Parametric solution details

There are requirements that you must consider when you load the master data, metadata, and event data.

Master data and metadata loading

Master data loading involves the loading of all master entities, such as Process, Resource, Product, Material, and Location.

Separate from the entities, Measurement type that is specific to a resource, product, process, or location must be loaded. For example, if temperature (TEMP) is the variable that is monitored, then the measurement type code for resource validation can be TEMP_R, and the measurement type code for location suitability can be TEMP_L.

In addition to the Measurement type, Event types must be loaded. An event type is dedicated to each sub use case. The following table describes the mapping of sub use cases to event type.

Table 16. Sub use case to event type mapping

Sub use case	Event type
Process - Resource Validation	PRVARIABLE
Resource Validation	RVARIABLE
Product Validation	PVARIABLE
Material Validation	MVARIABLE
Location Suitability	LVARIABLE

Each sub use case is identified by the defined set of Event types. When a sub use case is triggered, then data for the event type is fetched from the Event and Event Observation tables.

For every measurement type, a unique profile variable is defined so that the Analytics Solutions Foundation orchestration engine is leveraged to load parameters to profile parameter tables, with multiple master grain levels supported. When parameters are loaded, profile variables are defined with the profile variable code *MeasurementTypeCd_ParameterName*, where *MeasurementTypeCd* is the variable code with the PROFILE_PARAMETER_ASSIGN calculation configured.

Before you can load parameters into tables, you must define measurement type entries. For example, to measure thickness, this measurement type must be loaded. To begin, you load the Measurement type table. The following text is an example of loading the Measurement type table.

```
measurement_type_cd,measurement_type_name,unit_of_measure,carry_forward_indicator,  
aggregation_type,event_code_indicator,language_cd,tenant_cd  
THICKNESS_P,Thickness,,0,SUM,0,EN,PMQ
```

When the Measurement type table is loaded, you must load the Master_Profile_Variable table. The parameter name has values such as THICKNESS_P_TARGET, THICKNESS_P_SIGMA, THICKNESS_P_LAM0, THICKNESS_P_LAM1, THICKNESS_P_CONTROL, THICKNESS_P_FALSEALARMRATE, THICKNESS_P_UNACCEPTFACTORSIGMA, and THICKNESS_P_NO_DAYS.

The following CSV file is an example of the Master_Profile_Variable table.

```
profile_variable_cd,profile_variable_name,profile_calculation_name,
measurement_type_cd,resource_type_cd,material_type_cd,profile_units,comparison_string,
low_value_date,high_value_date,low_value_number,high_value_number,kpi_indicator,
profile_indicator,data_type,aggregation_type,carry_forward_indicator,process_indicator,
variance_multiplier,language_cd,tenant_cd
THICKNESS_P_TARGET,Thickness Target,ASSIGN,THICKNESS_P,ASSET,-NA,,,,,,,,,1,1,INT,,0,0,1,
EN,PMQ
THICKNESS_P_SIGMA,Thickness Sigma,ASSIGN,THICKNESS_P,ASSET,-NA,,,,,,,,,1,1,INT,,0,0,1,
EN,PMQ
THICKNESS_P_LAM0,Thickness Lam0,ASSIGN,THICKNESS_P,ASSET,-NA,,,,,,,,,1,1,INT,,0,0,1,
EN,PMQ
THICKNESS_P_LAM1,Thickness Lam1,ASSIGN,THICKNESS_P,ASSET,-NA,,,,,,,,,1,1,INT,,0,0,1,
EN,PMQ
THICKNESS_P_CONTROL,Thickness Control,ASSIGN,THICKNESS_P,ASSET,-NA,,,,,,,,,1,1,INT,,0,0,1,
EN,PMQ
THICKNESS_P_FALSEALARMRATE,Thickness FalseAlarmRate,ASSIGN,THICKNESS_P,ASSET,
-NA,,,,,,,,,1,1,INT,,0,0,1,EN,PMQ
THICKNESS_P_UNACCEPTFACTORSIGMA,Thickness UacceptFactorSigma,ASSIGN,THICKNESS_P,ASSET,-
NA,,,,,,,,,1,1,INT,,0,0,1,EN,PMQ
THICKNESS_P_NO_DAYS,Thickness no of days,ASSIGN,THICKNESS_P,ASSET,-NA,,,,,,,,,1,1,INT,,0,0,
1,EN,PMQ
```

Note that the profile variable connects the measurement type to the parameter name.

Event data loading

Parametric data uses the following two types of event data.

parameter data

Parameters remain unique for a combination of master entities. Parameters are loaded to the PROFILE_PARAMETER table.

variable data

Variable and observation reading data are loaded to the Event and Event Observation tables, supporting different master data grains.

If parameter and variable data must be linked to process, product, resource, location, material, or a combination of these entities, appropriate grains in the event message must be set.

Parameter event loading

Before you load parameters, you must understand the parameters that are required for the parametric use case. The following table describes the parameters that Predictive Maintenance and Quality uses.

Table 17. Parametric parameters

PMQ parameter	QEWS parameter	Description
SIGMA	Sigma	The assumed standard deviation of the measurements. This value must always be greater than zero.

Table 17. Parametric parameters (continued)

PMQ parameter	QEWS parameter	Description
TARGET	Target	The most desirable value for the center of the measurement population. Typically interpreted as the best level for the mean of the measurements.
LAM0	Accept_Level	The level of mean of the measurements that is still considered acceptable. Typically, this level is close to the Target, and it reflects the amount of wiggle room that remains for the population mean around the target. In many cases involving low process capability, this level coincides with the Target, indicating that there is no wiggle room for the population mean.
LAM1	Unaccept_Level	The level of mean of the measurements that is considered unacceptable. This is the level for which good detection capability is desired. Generally, an unacceptable level should be further away from the Target than the acceptable level. It is also advisable to keep a certain degree of separation between the acceptable and unacceptable levels (for example, no lower than $0.2 * \text{Sigma}$, where possible).
CONTROL	Type_of_Control	1 means that the control is one-sided (you are only interested in detecting changes up or down). 2 means that the control is two-sided: both types of deviation from the Target are considered unacceptable. If Type_of_Control = 1 and Accept_Level < Unaccept_Level, then only changes of the process mean up are of interest. If Type_of_Control = 1 and Accept_Level > Unaccept_Level, then only changes of the process mean down are of interest. If Type_of_Control = 2, then you can specify either Accept_Level < Unaccept_Level or Accept_Level > Unaccept_Level, with the understanding that the acceptable and unacceptable levels of the two-sided procedure are positioned symmetrically around the target.
FALSEALARM RATE	False_Alarm_Rate	Default = 1000, which means that the detection procedure produces a rate of false alarms of 1 per 1000 points (that is, 1 per 1000 values in the *.tsd data file) when the population mean is located at the Accept_Level.
UNACCEPTFACTOR SIGMA	Unaccept_Factor_Sigma	Presently not used.

Parameters, and combinations of parameters for the Resource, Location, Product, Material, and Process master entities are mapped to IBM Predictive Maintenance and Quality events. By using the PMQEventLoad flow in the orchestration engine, the parameters are loaded to the PROFILE_PARAMETER table, based on the Master_Profile_Variable definition and the orchestration definition. Parameter values are stored in the PARAMETER_VALUE column of the PROFILE_PARAMETER table, along with the profile variable and master data references mapped in the event.

When parameters are loaded, profile variables are defined with the profile variable code *MeasurementTypeCd_ParameterName*, where *MeasurementTypeCd* is the variable code. The PROFILE_PARAMETER_ASSIGN profile calculation is used to load parameters to the PROFILE_PARAMETER table.

Parameter event mapping

The following table describes the mapping of parameters to events.

Table 18. Parameter to event mapping

Parameter	Predictive Maintenance and Quality event
incoming_event_cd	incoming_event_cd
Hard coded to PARAMETERV	event_type_cd
Not applicable	source_system_cd
process_cd (if applicable)	process_cd
production_batch_cd (if applicable)	production_batch_cd
Not applicable	location_cd
Parameter load time/Parameter update time	event_start_time
Not applicable	event_end_time
Not applicable	event_planned_end_time
tenant_cd	tenant_cd
Not applicable	operator_cd
Model (if applicable)	model
serial_no (if applicable)	serial_no
Variable code	measurement_type_cd
Parameter load time/Parameter update time	observation_timestamp
Hard coded to ACTUAL	value_type_cd
ProfileVariableCd (<i>MeasurementTypeCd_ParameterName</i>)	observation_text
Parameter value	measurement
material_cd (if applicable)	material_cd
multirow_no	multirow_no

Parameter events processing

Parameter events are processed through Predictive Maintenance and Quality Eventload message flows according to the orchestration definition file.

The orchestration definition file for parameter events is named *PMQ_orchestration_definition_parameter.xml*, and it has a single orchestration step. For events with the PARAMETERV event type code and the ACTUAL value type, the PROFILE_PARAMETER_ASSIGN calculation is configured, and the profile adapter adds parameters to PROFILE_PARAMETER tables.

The following text is an example of loading parameter event data.

```
incoming_event_cd,event_type_cd,source_system_cd,process_cd,prod_batch_cd,
location_cd,event_start_time,event_end_time,event_planned_end_time,tenant_cd,
operator_cd,model,serialNo,measurement_type_cd,observation_timestamp,value_type_cd,
observation_text,measurement,material_code,multirow_no
```

```

1,PARAMETERV,,-NA-,PP9-XX9-009,-NA-,2014-11-26 00:00:00,2014-11-26 00:00:00,
2014-11-26 00:00:00,PMQ,,-NA-,-NA-,THICKNESS_P,2014-11-26 00:00:00,ACTUAL,
THICKNESS_P_FALSEALARMRATE,1000,-NA-,1
2,PARAMETERV,,-NA-,PP9-XX9-009,-NA-,2014-11-26 00:00:01,2014-11-26 00:00:01,
2014-11-26 00:00:01,PMQ,,-NA-,-NA-,THICKNESS_P,2014-11-26 00:00:01,ACTUAL,
THICKNESS_P_LAM0,0.85,-NA-,1
3,PARAMETERV,,-NA-,PP9-XX9-009,-NA-,2014-11-26 00:00:02,2014-11-26 00:00:02,
2014-11-26 00:00:02,PMQ,,-NA-,-NA-,THICKNESS_P,2014-11-26 00:00:02,ACTUAL,
THICKNESS_P_LAM1,0.9,-NA-,1
4,PARAMETERV,,-NA-,PP9-XX9-009,-NA-,2014-11-26 00:00:03,2014-11-26 00:00:03,
2014-11-26 00:00:03,PMQ,,-NA-,-NA-,THICKNESS_P,2014-11-26 00:00:03,ACTUAL,
THICKNESS_P_CONTROL,2,-NA-,15,PARAMETERV,,-NA-,PP9-XX9-009,-NA-,2014-11-26
00:00:04,2014-11-26 00:00:04,2014-11-26 00:00:04,PMQ,,-NA-,-NA-,THICKNESS_P,
2014-11-26 00:00:04,ACTUAL,THICKNESS_P_SIGMA,0.04,-NA-,1
6,PARAMETERV,,-NA-,PP9-XX9-009,-NA-,2014-11-26 00:00:05,2014-11-26 00:00:05,
2014-11-26 00:00:05,PMQ,,-NA-,-NA-,THICKNESS_P,2014-11-26 00:00:05,ACTUAL,
THICKNESS_P_TARGET,0.8,-NA-,1
7,PARAMETERV,,-NA-,PP9-XX9-009,-NA-,2014-11-26 00:00:06,2014-11-26 00:00:06,
2014-11-26 00:00:06,PMQ,,-NA-,-NA-,THICKNESS_P,2014-11-26 00:00:06,ACTUAL,
THICKNESS_P_UACCEPTFACTORSIGMA,1.5,-NA-,1
8,PARAMETERV,,-NA-,PP9-XX9-009,-NA-,2014-11-26 00:00:07,2014-11-26 00:00:07,
2014-11-26 00:00:07,PMQ,,-NA-,-NA-,THICKNESS_P,2014-11-26 00:00:07,ACTUAL,
THICKNESS_P_NO_DAYS,2000,-NA-,1

```

Variable event loading

Variable data is made up of the measurements and observation readings that are taken at different time intervals during the sequence of the operation flow. Observation readings or variable data are mapped to Predictive Maintenance and Quality events, and loaded to event and event observation tables. The orchestration engine is used to load the observation readings or variable data by using the PMQEventLoad flow, based on the orchestration definition. Master data references in the event vary, depending on the type of event and sub use case.

Variable event mapping

The following table describes the parameter to event mapping for variables.

Table 19. Parameter to event mapping for variables

Parameter	Predictive Maintenance and Quality event
incoming_event_cd	incoming_event_cd
Hard coded to event type, based on sub use case chosen (PRVARIABLE, RVARIABLE, PVARIABLE, MVARIABLE, LVARIABLE)	event_type_cd
Not applicable	source_system_cd
process_cd (if applicable)	process_cd
production_batch_cd (if applicable)	production_batch_cd
location_cd (if applicable)	location_cd
event_start_time	event_start_time
Not applicable	event_end_time
Not applicable	event_planned_end_time
tenant_cd	tenant_cd
Not applicable	operator_cd
Model (if applicable)	model
serial_no (if applicable)	serial_no

Table 19. Parameter to event mapping for variables (continued)

Parameter	Predictive Maintenance and Quality event
Variable code	measurement_type_cd
observation_timestamp	observation_timestamp
Hard coded to ACTUAL	value_type_cd
Not applicable	observation_text
Observation reading or variable value	measurement
material_cd (if applicable)	material_cd
multirow_no	multirow_no

Master data references in the event vary, depending on the type of event or sub use case.

The following text is an example of variable event data.

```
incoming_event_cd,event_type_cd,source_system_cd,process_cd,prod_batch_cd,location_cd,
event_start_time,event_end_time,eventlanned_end_time,tenant_cd,operator_cd,model,serialNo,
measurement_type_cd,observation_timestamp,value_type_cd,observation_text,measurement,
material_code,multirow_no
1,PBVARIABLE,,-NA-,PP9-XX9-009,-NA-,2014-11-28 01:10:59,2014-11-28 01:10:59,
2014-11-28 01:10:59,PMQ,,-NA-,-NA-,THICKNESS_P,2014-11-28 01:10:59,ACTUAL,,0.75,-NA-,1
2,PBVARIABLE,,-NA-,PP9-XX9-009,-NA-,2014-11-28 02:10:59,2014-11-28 02:10:59,
2014-11-28 02:10:59,PMQ,,-NA-,-NA-,THICKNESS_P,2014-11-28 02:10:59,ACTUAL,,0.79,-NA-,1
3,PBVARIABLE,,-NA-,PP9-XX9-009,-NA-,2014-11-28 03:10:59,2014-11-28 03:10:59,
2014-11-28 03:10:59,PMQ,,-NA-,-NA-,THICKNESS_P,2014-11-28 03:10:59,ACTUAL,,0.79,-NA-,1
4,PBVARIABLE,,-NA-,PP9-XX9-009,-NA-,2014-11-28 04:10:59,2014-11-28 04:10:59,
2014-11-28 04:10:59,PMQ,,-NA-,-NA-,THICKNESS_P,2014-11-28 04:10:59,ACTUAL,,0.77,-NA-,1
```

Variable events processing

Variable events are processed through Predictive Maintenance and Quality Eventload message flows according to the orchestration definition file.

The orchestration definition file for parametric events is named `PMQ_orchestration_definition_parametric.xml`, and it has a single orchestration step. The event store adapter stores variable raw events into event and event observation tables. The event type is used to differentiate between the events of different sub use cases.

Parametric orchestration and triggering mechanism

The parametric use case has the following batch triggering modes:

- Timer-based triggering, which passes the run date as the current date, and passes the sub use case name to the algorithm invocation flow.
- File-based triggering, which passes the run date and the event type code (the sub use case) as input.

Depending on the parametric use case, the `SubUseCase` value is configured:

- RVALIDATION for resource health check
- PRVALIDATION for process resource validation
- MVALIDATION for material validation
- PBVALIDATION for product validation
- LVALIDATION for location suitability

Timer-based triggering

The orchestration definition file `PMQ_orchestration_definition_batch.xml`, in the `properties` directory on the Integration Bus node computer, is configured to trigger the parametric timer at the configured scheduled time once per day. The `AutoTrigger` flow of the batch integration process flow accepts an input parameter, such as the `SubUseCase` name, from the scheduler configuration, in addition to the schedule time, the queue name, and the duration. The `AutoTrigger` flow places the timer request to the configured queue, which triggers the `ProcessParametric` flow at the scheduled time. The `ProcessParametric` uses the `SubUseCase` name and the run date (as the current date) as inputs. The `ProcessParametric` flow invokes the `QEWS` algorithm.

You can change the `SubUseCase` name and timer configuration in the orchestration definition file at run time.

File-based triggering

In file-based triggering, a file with the naming convention `parametric_rundate*.txt` must be placed in the `batchdatain` folder. The contents of the file must have the following format:

```
rundate=2014-12-01  
subusecase=PBVALIDATION
```

The file is picked up by the `ParametricDataPreparation` flow, which is included in the `PMQQEWSIntegration` application. The `ParametricDataPreparation` flow converts the file into an WebSphere MQ message, and places the message in the `PMQ.QEWS.PARAMETRIC.IN` queue. The `ProcessParametric` flow is triggered, with the `SubUseCase` name and the run date as inputs.

Parametric algorithm invocation

The following diagram shows how the parametric algorithm is invoked.

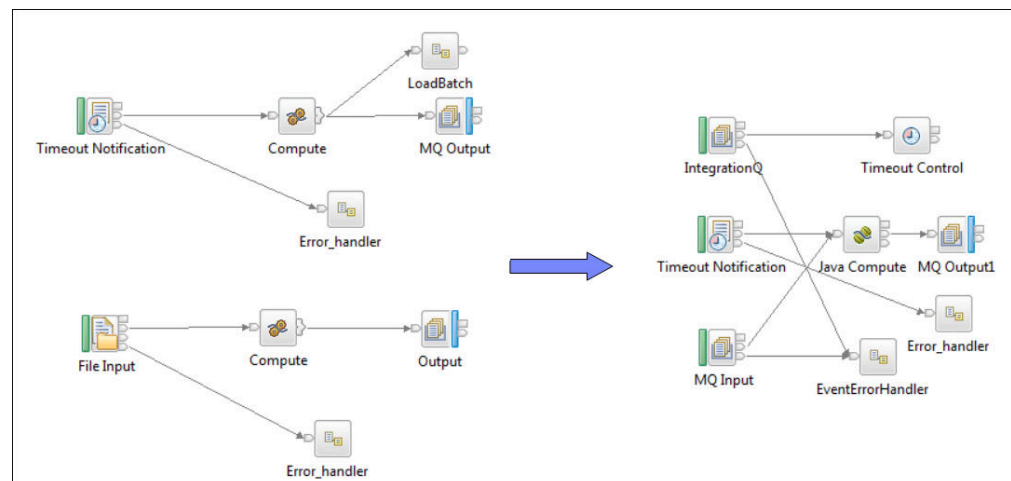


Figure 26. Parametric algorithm invocation

Output and reporting

After the parametric algorithm is invoked by using the timer-based or file-based triggering mode, the Event and Event Observation tables are queried for a specific

event type and date range. The date range covers a specified number of days backward from the run date. Each sub use case has a specific event type.

The specified number of days is taken from the NO_DAYS parameter. When the grain combination is identified, the parameters are retrieved from the PROFILE_PARAMETER table.

The data set from the Event and Event Observation tables and the parameters are passed to the parametric algorithm. The results of the parametric analysis are persisted into the PARAMETRIC_KPI and PARAMETRIC_PROFILE tables. These two tables include the Rundate and Event type ID columns. The Rundate column holds the date when the QWESV algorithm was triggered and when the execution began. The Event Type ID column refers to the Master Event Type table, where the Event type code for each of the sub use cases Process, Resource Validation, Material Validation, Location Validation, and Product Validation is stored.

The Rundate column provides the option of storing the processed data based on the execution date, which maintains the processed records of earlier runs. Currently, there is no purging policy set for the KPI and Profile tables. The purging policy may later be defined based on the business requirements.

IBM Cognos Business Intelligence includes the Rapidly Adaptive Visualization Engine (RAVE) that is used to build the Parametric chart. CognosBI queries the PARAMETRIC_KPI and PARAMETRIC_PROFILE tables based on the run date value, and gathers the records that fit the run date value. The records are populated into a .json file at run time, and the .json file is used to prepare the charts.

Results and benefits

The quality early warning system for variables (QEWSV) in IBM Predictive Maintenance and Quality reduces cost by detecting problems and issues earlier and more accurately.

Results

Predictive Maintenance and Quality QEWSV delivers the following results:

- Improves production yields in the manufacturing line.
- Helps you to gain a better understanding of root causes of manufacturing issues by dealing directly with Variables and Operations.
- Provides faster detection of manufacturing quality problems.

Benefits

Subtle changes in failure rates that indicate potential emerging quality problems are detected earlier. Early detection means quicker problem identification, faster problem resolution, and lower total costs. The definitive nature of QEWSV alerts eliminates the need for subjective judgment of statistical process control charts and other traditional tools, providing you with consistent and accurate direction. QEWSV can deliver insightful early warning signals, even under variable lot size scenarios.

Chapter 7. Situational Awareness and UI and Service Framework

Situational Awareness is an application that helps operators monitor asset status changes in real time with different views. The Situational Awareness application also provides the Standard Operating Procedure feature to help users define procedure templates, and trigger the procedures if needed, usually when an asset is in abnormal status. The Situational Awareness application is based on a UI and Service framework that is a programming model that enables developers to easily customize existing applications or create new applications. The UI framework helps accelerate application development on the front end, and the REST service framework provides an extension mechanism that enables developers to develop services quickly.

Managing the Standard Operating Procedures

A Standard Operating Procedure (SOP) is a set of instructions that describes all the relevant steps and activities of a process or procedure.

When you define an SOP, you define activities that are included in the SOP. SOP enables an administrator to organize personnel, information, and tasks in response to events and incidents in order to achieve a comprehensive control of the operation. A SOP comprises of these components:

Standard Operating Procedure Definition

An SOP definition is the template that is used when a SOP is instantiated in response to a particular occurrence. A SOP Definition is made up of activities that are described by Activity Definitions.

Activity Definition

A SOP Definition contains one or more Activity Definitions. An activity definition sets the individual instructions that need to be performed as part of the SOP.

SOP Instance

A single Instance of an SOP in response to a particular event or occurrence. One SOP Definition can be used for many SOP Instances. An SOP Instance can be in one of these states.

- Active
- Started
- Stopped
- Completed
- Canceled

Activity Instance

An Activity Instance is the instantiation of a single Activity Definition. A single Activity Definition can be used to create multiple Activity Instances. An Activity Instance can be in a number of states:

- Active
- Waiting
- Started
- Skipped

- Completed

References

Supplemental information which is relevant to a Standard Operating Procedure or Activity. References can also be used to define e-mail templates.

Roles There are two abilities, Owners and Readers. These can be set against administrative and user roles.

- A Reader can monitor the activities that are associated with a standard operating procedure.
- An Owner can monitor and complete the activities that are associated with the standard operating procedure.

Activity Type

The Activity Type describes the response to the activity. The activities can be of different types and execution models. Any combination of different activities in an SOP is allowed.

- Manual: This type of activity must be manually carried out by the owner of the SOP.
- If-Then-Else Activity: A conditional activity that allows branching based on specific criteria. The user can choose which of the SOP definitions to instantiate when starting the activity. Either enter or select values for Then and Else.
- Alert Activity: This activity displays an e-mail template for the SOP owner to complete and send an email notification to predefined personnel.
- REST Activity: An activity that creates a REST service call. The user can specify the service URL and any required authentication information to be invoked when the activity is started.
- SOP Activity: An activity that starts another standard operating procedure.

Roles for Standard Operating Procedures

The abilities for each of the roles for SOPs are as follows:

SOP Administrator roles

- View and delete an SOP definition
- Launch, view, and edit an SOP instance
- Start and complete activities in an SOP instance

SOP author roles

- Create, edit, view, and delete an SOP definition
- Create an SOP draft
- View, edit, and delete an SOP activity
- Submit an SOP draft for approval
- Approve an SOP draft

Reference Librarian Role

Create shared references

Owner Roles (SOP definition)

- Create an SOP draft
- View, edit, and delete an SOP definition

- Edit and delete an SOP activity
- Submit an SOP draft for approval
- Approve an SOP draft
- Launch, view, and edit an SOP instance

Reader roles (SOP Definition)

- View an SOP definition
- View an SOP instance from My Activities
- View an SOP activity, provided the user has Reader role in the Activity definition

Owners roles (SOP activity)

- View an SOP instance from My Activities
- Start and complete activities in an SOP instance for their own activities from My Activities

Reader roles (SOP activity)

View SOP instance from My Activities

Approval life cycle for an Standard Operating Procedure

An SOP definition can assume different status during its life cycle.

- Draft: When the SOP is first created, a draft version is saved initially. From an approved version of an SOP, it is also possible to create another draft version, when it is necessary to change the SOP definition using the approved version as a base. A draft can be edited, submitted for approval, or discarded.
- Pending approval: This is a draft SOP definition submitted for approval, ready to be approved or disapproved. The name of the version is defined in this status and it will name the SOP definition version if approved. If this version is not approved, the SOP definition goes back to the draft version status.
- Approved: When an SOP definition is approved it is ready to be launched.

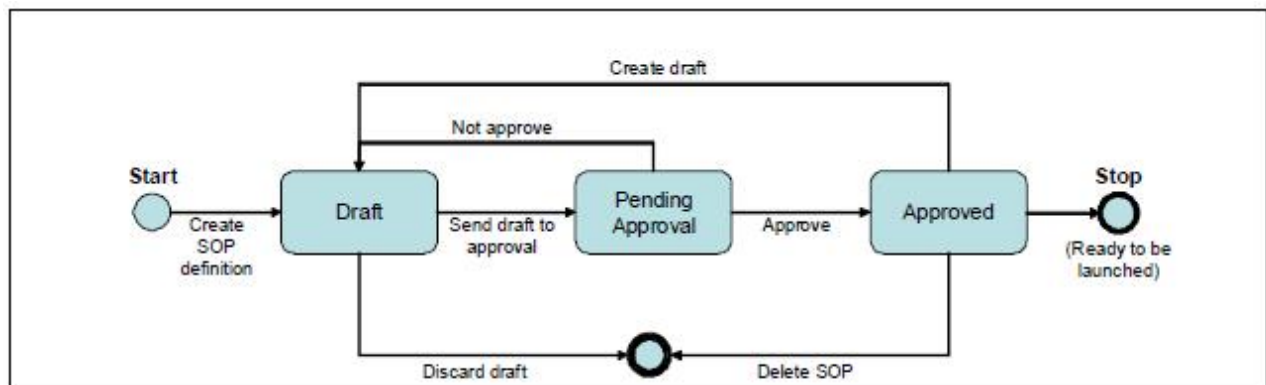


Figure 27. SOP Life Cycle

Configuring Standard Operating Procedures for different activities

One of the important tasks for when defining a Standard Operating Procedures (SOP) is to define the set of activities that composes the procedure.

The activities can be of different types and execution models. Any combination of different activities in an SOP is allowed.

Configuring activities to start in sequence

When you specify that activities are done in sequence, you will not be able to start an activity until the predecessor activity is completed.

Whether the activities of an SOP are executed sequentially or not is specified in the General Settings section of the SOP definition.

Specify that activities must be executed sequentially where the activities must be executed in a chronological order or where activities depend on the result of previous activity.

Examples of sequential activities are:

- MA1 - Approve an operation.
- SA2 - Collect information about an incident area and the number of victims.
- SA3 - Prepare personnel for standby.
- SA4 - Redirect traffic to clear an incident area.

The Sequential activities figure shows an activity flow with activities that run in sequential order.

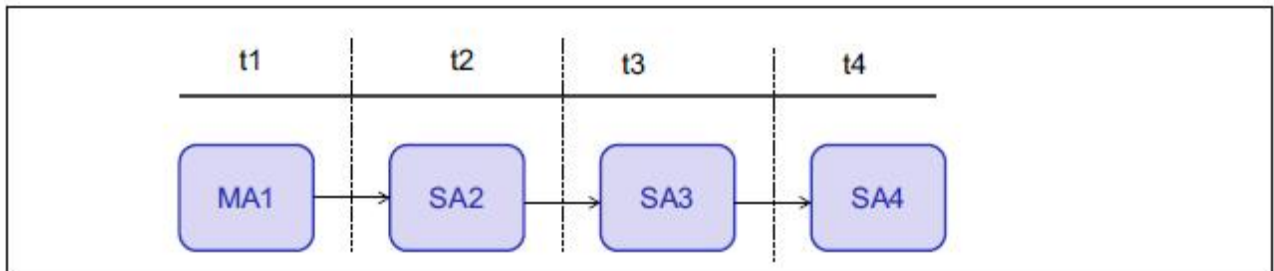


Figure 28. Sequential activities

Note:

- The user can start SA2 only if MA1 is complete, SA3 after SA2 is complete, and SA4 after SA3 is complete. The sequential property of an SOP applies to all activities, either all activities are sequential or none for the entire SOP.
- t1, t2, t3 and t4 represents the duration of the activity.

Configuring required activities

A required activity is one that is mandatory in a Standard Operating Procedure (SOP). You cannot skip over an activity that is specified as a required activity in the SOP definition.

Any of the activity types can be configured as required. Required is an attribute of an activity therefore an SOP can have activities that are required and activities that are optional (they can be skipped over). Examples of required activities are:

- MA1 - Approve operation. Mandatory.
- A2 - Collect information about incident area and number of victims. Optional.
- A3 - Prepare personnel for standby. Optional.
- A4 - Redirect traffic to clear incident area. Optional.

The mandatory activities figure shows an activity flow with activities that have mandatory and optional activities.

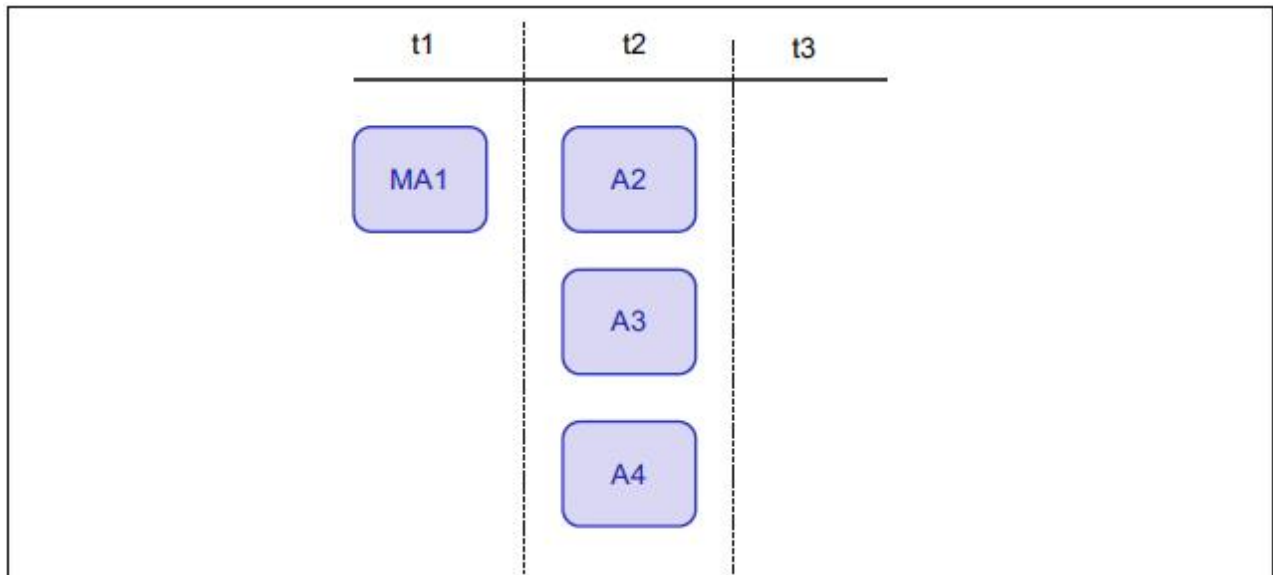


Figure 29. Mandatory activities

Note:

- MA1 is mandatory. All the other activities are optional and do not need to be performed sequentially.
- A2, A3, and A4 can be run in parallel.
- t1, t2, t3 represents the duration of the activity.

Configuring manual activities

A manual activity type is an activity that must be manually performed by the owner once the Standard Operating Procedure is launched.

Manual activities are the most basic and essential type. Examples of manual activities are:

- A1 - Collect information about the incident area and number of victims.
- A2 - Prepare personnel for standby.
- A3 - Redirect traffic to clear incident area.

The manual activities figure shows an activity flow with activities that are started manually. In the figure, A1, A2, and A3 are all manual activities. t1, t2, t3 represents the duration of each activity.

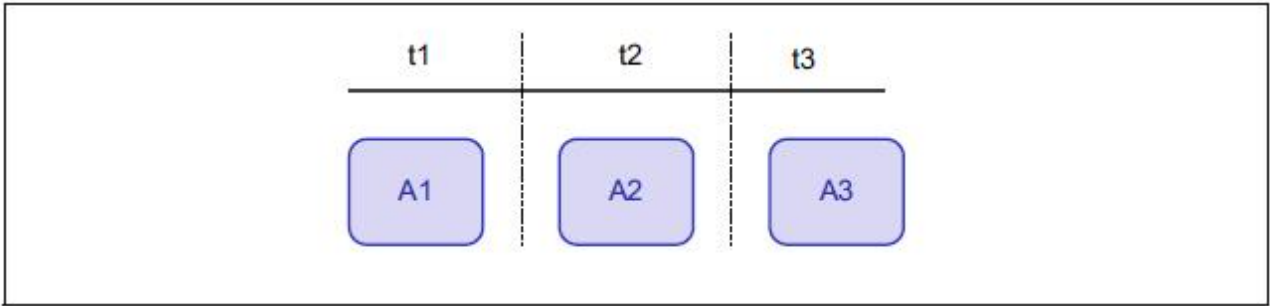


Figure 30. Manual activities

Configuring conditional activities

Conditional activities are conditional activities that allow branching based on If-Then-Else criteria.

When the activity is launched, the operator decides whether to execute one set of actions (Then) or another set (Else).

The conditional activity is useful in situations where you want to use the results from other activities to make a decision and launch a new SOP to respond to the situation. Examples of a conditional activity is:

- MA1 - Approve operation.
- SA2 - Collect information about incident area and number of victims.
- SA3 - Prepare personnel for standby.
- SA3 If (team is not available = true) Then initiate contingency plan. Conditional activity.
- SA4 - Redirect traffic to clear incident area.

The conditional activity figure shows a conditional activity flow.

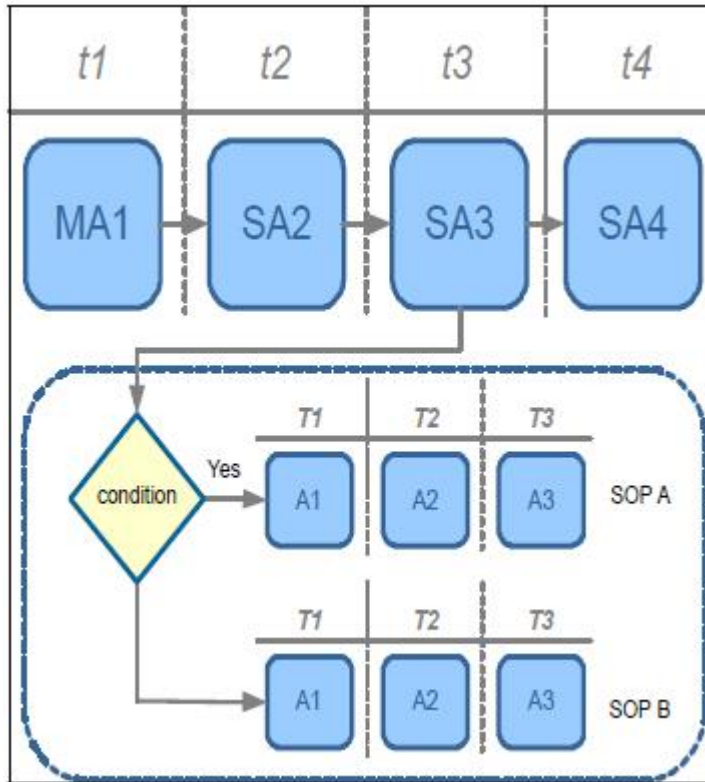


Figure 31. Conditional activity

When the SOP instance runs, the operator decides whether to launch the Then SOP or the Else SOP, if one is available, by selecting **Start Then** or **Start Else** options in my activity page.

Configuring notification activities

Notification activities enable the operator to complete an email and send it. The email notification occurs as part of the required activity.

You need to set up an SMTP server to send the email notifications before configuring and launching a notification activity.

To set up an SMTP server, go to the `sysprop.json` table in the IFE database and update the following system properties in the SOP group:

- MailServerHostname: hostname of the SMTPserver
- MailServerPort: SMTP server port
- MailSender: sender of emails to be sent by the activity in the **From** field.

Email templates can be created and then reused for a notification activity. All email templates are stored as **References**.

Here are example templates There are some email examples created as references, please access them using these urls where <liberty server ip> is the ip address of the IFE server:

https://<liberty server ip>:9443/ibm/ife/sop/app/js/ife/sop/email/emailTemplate_fireEvent.txt

https://<liberty server ip>:9443/ibm/ife/sop/app/js/ife/sop/email/
emailTemplate_bombThreat.txt

https://<liberty server ip>:9443/ibm/ife/sop/app/js/ife/sop/email/
emailTemplate_disturbanceEvent.txt

https://<liberty server ip>:9443/ibm/ife/sop/app/js/ife/sop/email/
emailTemplate_evacuation.txt

https://<liberty server ip>:9443/ibm/ife/sop/app/js/ife/sop/email/
emailTemplate_prep_for_power_loss.txt

https://<liberty server ip>:9443/ibm/ife/sop/app/js/ife/sop/email/
emailTemplate_radiationHazard.txt

https://<liberty server ip>:9443/ibm/ife/sop/app/js/ife/sop/email/
emailTemplate_transitionToNewERlevel.txt

https://<liberty server ip>:9443/ibm/ife/sop/app/js/ife/sop/email/
emailTemplate_weatherEvent.txt

To create a notification template:

1. Create a text file that contains the email template. The To, Subject, and Body fields are optional. Here is an example template:

```
-----  
To:  
Subject:Attention: Emergency incident detected  
Body:Emergency incident detected. Please contact administrator for  
further Information.  
-----
```

2. Publish the template in a location on the application server. Usually, the web content directory of the Situational Awareness application, which is accessible by the Situational Awareness application.
3. Create a reference in the Standard Operating Procedures References and point the address to the URL where the template is published.
4. Make sure the first line of the description field contains only the keyword NOTIFICATION.
You can add further description details in subsequent lines.

Note: If you receive a 401 error when you try to load the template into the email window, you need to include your credentials in the URI, for example:https://user_name:password@<host>:<port>/test-url/notification.

Defining a Standard Operating Procedure

Define an SOP with all activities.

Procedure

1. On the user interface, click **Administration > Standard Operating Procedures > SOP Definition**.
2. In the Basics section, define the basic information for the SOP.
 - a. Click **Create**.
 - b. In the Basics section, type a name and description for the SOP.

- c. If the activities need to be carried out in a particular order, select **Activities are done in order** in the **General Settings** field.
- d. Click **Add References** to add supplemental information.
- e. Click **Next** to go to the Roles section.
3. In the Roles section, define the roles for the SOP.
 - a. Select the **Roles** that will be assigned as **Owners** and **Readers** for the SOP created with this definition.
 - If you want the role to be able to monitor activities that are associated with the standard operating procedure, select **Reader**.
 - If you want a role to be able to monitor and complete activities that are associated with the standard operating procedure, select **Owner**.
 - b. Click **Next** to go to the Activities section.
4. In the Activities section, define the activities for the SOP.
 - a. Click **Add**.
 - b. If the activity is required, select **Required**. If not, the SOP can move on to the next activity during instantiation.
 - c. Select **Autostart** if this activity is to start automatically without owner operation.
 - d. You can select the roles for the Owners and Readers. If not, the roles are inherited from the SOP definition.
 - e. Set the duration of the activity. The duration is the length the activity takes once started.
 - f. Type a description of the activity.
 - g. Set the Activity type. The options are:
 - Manual Activity
 - If-Then-Else Activity
 - Alert Activity
 - REST Service
 - SOP Activity
 - h. To add more activities click **Add** and define the next activity for the SOP.
 - i. Click **Next** to go to the Summary section.
5. Review the SOP.
 - a. Review the information in the summary.
 - b. Click **Save**.

Important: The SOP is in now draft state. The SOP must be approved before it can be instantiated.

In the draft state, the SOP can be edited or deleted if not required.

Creating a reference for Standard Operating Procedures

References are supplemental information relevant to an SOP or an activity. References can also be used to define e-mail templates.

About this task

A user can create references for Standard Operating Procedures.

Procedure

1. On the user interface, click **Administration > Standard Operating Procedures > References**.
2. Click **Add**.
3. In the **Name** field, type the name of the reference.
4. In the **URI** field, type or paste the web accessible address.
5. In the **Description** field, type a description of the reference.
6. If you want to restrict the use of the reference, select the **Private** check box. Otherwise the reference can be shared to other users.

Editing a Standard Operating Procedure

Edit a Standard Operating Procedure.

About this task

To be able to edit an approved Standard Operating Procedure (SOP), the SOP must be returned to a draft version.

Note: You can always edit an SOP in a draft version.

Procedure

1. On the user interface, click **Administration > Standard Operating Procedures > SOP Definition**.
2. From the list of SOP definitions, select the SOP to be edited.
3. From the **Version** field, select the version to be edited.
4. Click **Create Draft**. The SOP state is now in draft.
5. Click the SOP you want to edit.
 - To add an activity, click **Add** in the Activities tab.
 - To edit an activity, click the activity and then click **Edit**.
 - To delete an activity, click the activity and then click **Delete**.
 - To edit a role, select the Roles tab and then click **Edit**.
 - To edit references, select the References tab, click **Edit** or **Delete** for a specific reference or click **Add** to add a new reference.
6. When the edits are complete, click **Save**.

What to do next

Before the edits are accepted, you must submit the draft version for approval.

Submitting a draft Standard Operating Procedure for approval

You can submit a draft version of an SOP for approval or discard it.

Before you begin

The SOP must be set to Draft version, and you must have the administrator role.

Procedure

1. On the user interface, click **Administration > Standard Operating Procedures > SOP Definition**.

2. Select the SOP for submission that is in **Draft** state, and click **Submit for Approval**.

Attention: An SOP that has been submitted for approval cannot be edited. The administrator can either Approve or Disapprove the SOP.

Testing a Standard Operating Procedure

Test an approved version of a Standard Operating Procedure (SOP).

About this task

An SOP must be in an approved state before it can be tested.

Procedure

1. On the user interface, click **Administration > Standard Operating Procedures**.
2. Click on the approved SOP Definition that you want to test.
3. Click **Definition Actions** and from the drop-down list select **Launch**.

Once launched, the SOP Definition list will update to indicated the number of Active SOPs based on this definition.

The **My Activities** banner indicator shows that there are one or more activities that require attention.

4. You can perform the SOP and check for completeness.
5. If the SOP needs further work, click **Create Draft** to return the SOP to a draft state.

Exporting a Standard Operating Procedure

You can export your Standard Operating Procedure definitions as an XML file. This capability is useful for migration purposes.

About this task

To export an SOP definition performs the following steps:

Procedure

1. On the user interface, click **Administration > Standard Operating Procedures > SOP Definition**.
2. Select **Standard Operating Procedures Definition** administration page.
3. From the **Definition Actions** drop-down list, select **Export All**.
4. In the pop-up window that is displayed, click **Save File > OK**.
5. Navigate to the directory where you want to save the SopDefinitions.xml file and click **Save**.

Importing a Standard Operating Procedure

You can import a previously exported Standard Operating Procedure definitions file. This capability is useful in cases where the organization has predefined SOPs that were implemented using a different tool.

About this task

To import an SOP definition performs the following steps:

Procedure

1. On the user interface, click **Administration > Standard Operating Procedures > SOP Definition**.
2. Select **Standard Operating Procedures Definition** administration page.
3. From the **Definition Actions** drop-down list, select **Import**.
4. In the Import Standard Operating Procedures window, select the XML file to upload.

The imported definitions are displayed in the list of SOP definitions. You can import an SOP definition that has the same name as an existing SOP definition. In this case, the existing SOP definition is not overwritten. Instead, two SOP definitions with the same name are displayed in the list of definitions.

5. Click **Import**.

Reverting to a particular version of a Standard Operating Procedure

You can revert back to a particular version of a Standard Operating Procedure (SOP).

About this task

The Standard Operating Procedures shows the latest version of a definition. If you need to revert to an earlier version of a definition, you can use these steps.

Procedure

1. On the user interface, click **Administration > Standard Operating Procedures > SOP Definition**.
2. From the list of SOP definitions, select the SOP to be edited.
3. From the **Version** field, select the version to be reverted to.
4. Click **Create Draft**. The SOP state is now in draft.

What to do next

Before the revert is accepted, you must submit the draft version for approval.

Viewing a Standard Operating Procedure

You can view a Standard Operation Procedure (SOP) as an administrator.

There are two ways to view an SOP instance:

- From the SOP Administration page.
- From the My Activities widget.

Viewing a Standard Operating Procedure as an administrator

You can view a Standard Operation Procedure (SOP) as an administrator.

About this task

To view the SOP instance from the SOP Administration do the following steps:

Procedure

1. On the user interface, click **IFE Administration > Standard Operating Procedures**.

2. Click on the approved SOP Definition that you want to view.
3. Click **Definition Actions** and from the drop-down list select **Launch**.
Once launched, the SOP Definition list will update to indicated the number of Active SOPs based on this definition.
4. Click the approved SOP Definition.
5. Click the Instances tab.
6. Click the SOP instance to view the details.

Viewing a Standard Operating Procedure as a user

You can view a Standard Operation Procedure (SOP) as a user.

Before you begin

The Standard Operating Procedure must have been launched before a user can view the instance.

About this task

Users that are not authorized to view the Standard Operation Procedures Definition administration page, can view an SOP instance from My Activities.

Procedure

1. On the user interface, click **Administration**.
2. Click the **My Activities** icon.
3. Click the SOP instance you want to view.

Extend situation awareness

The Situation Awareness Application consists of a presentation Layer, core Services, a core data model, data integration and an application content pack. The application is consists of two parts: The front end and back end. The front end runs on the UI Framework and back end is part of the application content pack, that includes the service and data model.

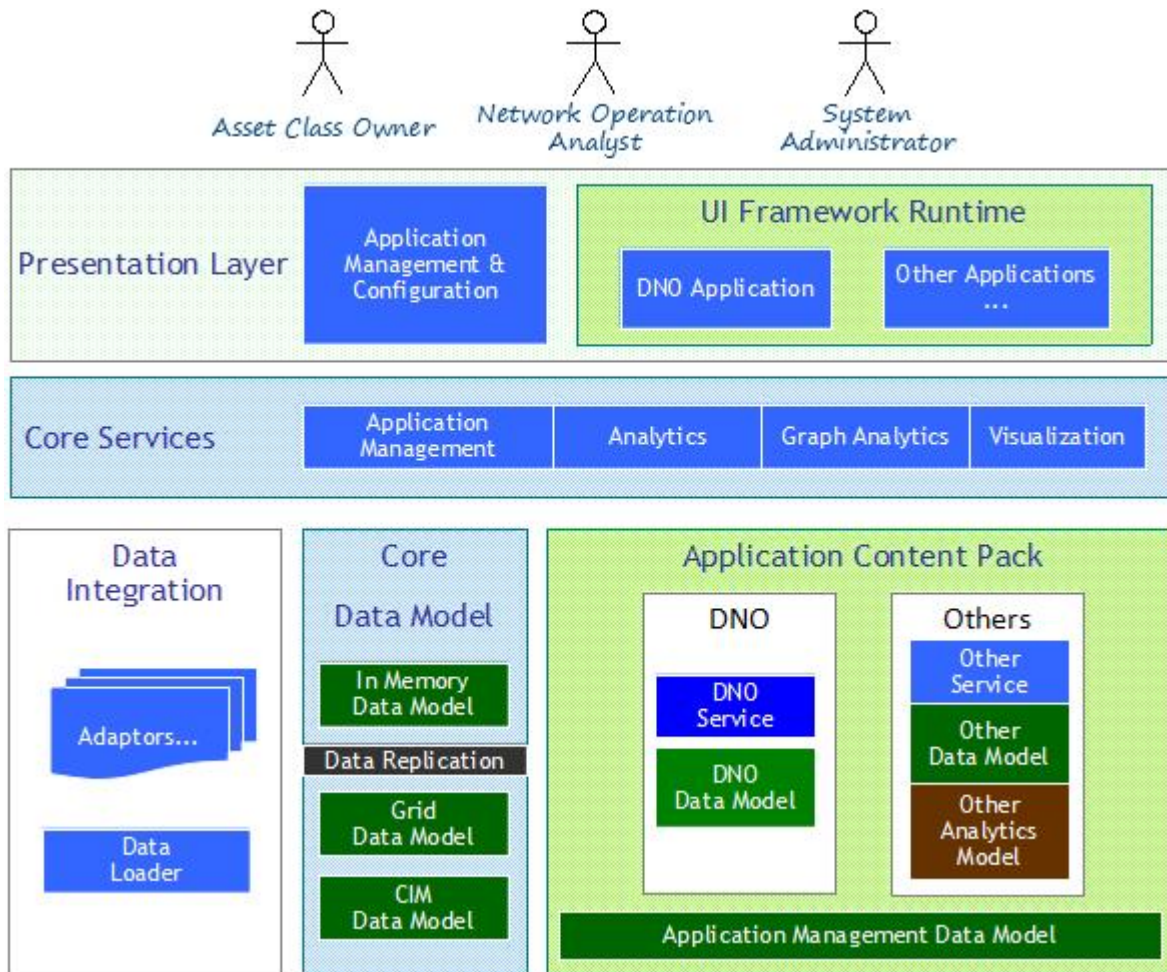


Figure 32. Situation Awareness overview

Configuration of the front end development

The configuration based methodology for the front end development achieves a rapid development for the application.

The entry configuration file `main.json` includes a series of sub-configuration files depending of the functionality required. The JSON format configuration files are responsible for:

- Create widget instances
- Configure widget parameters
- Configure widgets containment
- Configure dependency injection among widgets
- Configure binding mechanism between source and target widgets so that the changes happening in source widget can be notified to target widget

The list of the configuration files include:

main.json

entry configuration

layout.json

page layout relevant configuration

header.json

header bar relevant configuration

filter.json

filter relevant configuration

content.json

page content relevant configuration

map.json

map view relevant configuration

list.json

list view relevant configuration

previewCard.json

preview card relevant configuration

assetDetails.json

asset details relevant configuration

model.json

model relevant configuration

hierarchyModel.json

hierarchy model relevant configuration

bind.json

widget interaction relevant configuration

Model driven back end implementation

The back end implementation includes the REST service and the data model, where the data model is the base of the whole back end implementation.

Application REST Services

All the REST services are configured in dno.xml. The implemented services are:

Table 20. Implemented Asset services

Service URL	Create	Update	Read
/overheadline	Yes	Yes	Yes
/substation	Yes	Yes	Yes
/wind_farm	Yes	Yes	Yes

Table 21. Implemented Measurement services

Service URL	Create	Update	Read
/measurement	Yes	Yes	Yes
/measurement_thresho	Yes	Yes	Yes
/measurement_timeout	Yes	Yes	Yes

Table 22. Implemented Reading services

Service URL	Create	Update	Read
/health_index	Yes	Yes	Yes
/health_index/ byYear/ health_index/ byMonth/ health_index/byDay/ health_index/ byHour/ health_index/ byMinute	No	No	Yes
/load_index	Yes	Yes	Yes
/load_index/byYear/ load_index/byMonth/ load_index/byDay/ load_index/byHour/ load_index/byMinute	No	No	Yes
/impact_level	Yes	Yes	Yes
/impact_level/ byYear/ impact_level/ byMonth/ impact_level/byDay/ impact_level/ byHour/ impact_level/ byMinute	No	No	Yes

Extending the application

Three types of assets are implemented: overhead line, substation and wind farm. These are the steps to follow to add a new asset type called underground cable.

The procedures show an example of how to add underground cable to extend the application.

Extending the data model

To extend the data model you need to identify the table and the relationships, create the objects and then load them into the database.

Procedure

1. Identify the table and the relationship to be created. The table is called DNO.UndergroundCable.
2. Provide the database object creation script:

```
CREATE TABLE DNO.UndergroundCable (
    ASSET_ID INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY
    (START WITH 1 INCREMENT BY 1 NO CYCLE CACHE 20 NO ORDER),
    SERIAL_NUMBER VARCHAR(128) NOT NULL,
    NAME VARCHAR(128),
    DESCRIPTION VARCHAR(512),
    INSTALLATION_DATE DATE,
    STATUS SMALLINT,
    LOCATION DB2GSE.ST_GEOMETRY
```

```
)
DATA CAPTURE NONE
COMPRESS YES ADAPTIVE
VALUE COMPRESSION;
```

3. Prepare the data to be loaded into database.

- a. Prepare the asset data for the new asset type underground cable and put the following example content into a file called `undergroundcable.csv` and save it in the directory `/opt/IBM/energy/data/dno_sample/csv` on the application server.

```
ASSET_ID,SERIAL_NUMBER,NAME,DESCRIPTION,INSTALLATION_DATE,STATUS,LOCATION
3001,UGC_N1,Underground Cable N1,Underground Cable N1,2014-10-01,,
"LINESTRING (-3.0894280 55.3783630, -3.1038610 55.3800519)"
3002,UGC_N1,Underground Cable N1,Underground Cable N1,2014-10-01,1,
"LINESTRING (-1.0903370 53.9502490, -1.1012380 53.9519670)"
3003,UGC_N2,Underground Cable N2,Underground Cable N2,2014-10-01,1,
"LINESTRING (-1.1012380 53.9519670, -1.1108510 53.9547951)"
3004,UGC_N3,Underground Cable N3,Underground Cable N3,2014-10-01,1,
"LINESTRING (-1.1108510 53.9547951, -1.1202921 53.9559570)"
3005,UGC_N4,Underground Cable N4,Underground Cable N4,2014-10-01,1,
"LINESTRING (-1.1202921 53.9559570, -1.1320510 53.9545429)"
```

- b. Prepare the measurement data for the underground cable asset and append the example content into the file `measurement.csv` in the folder `/opt/IBM/energy/data/dno_sample/csv` on the application server.

```
1201,undergroundcable,3001,UGC_3001_Impact_Level,
Impact_Level,Impact_Level,Impact_Level,
1202,undergroundcable,3002,UGC_3002_Impact_Level,
Impact_Level,Impact_Level,Impact_Level,
1203,undergroundcable,3003,UGC_3003_Impact_Level,
Impact_Level,Impact_Level,Impact_Level,
1204,undergroundcable,3004,UGC_3004_Impact_Level,
Impact_Level,Impact_Level,Impact_Level,
1205,undergroundcable,3005,UGC_3005_Impact_Level,
Impact_Level,Impact_Level,Impact_Level,
```

- c. Prepare measurement threshold data for the underground cable asset and append the example content into the file `measurement_threshold.csv` in the folder `/opt/IBM/energy/data/dno_sample/csv` on the application server.

```
1201,0.0,3.0,0
1201,3.0,5.0,1
1201,5.0,6.0,2
1202,0.0,3.0,0
1202,3.0,5.0,1
1202,5.0,6.0,2
1203,0.0,3.0,0
1203,3.0,5.0,1
1203,5.0,6.0,2
1204,0.0,3.0,0
1204,3.0,5.0,1
1204,5.0,6.0,2
1205,0.0,3.0,0
1205,3.0,5.0,1
1205,5.0,6.0,2
```

- d. Prepare the reading data for the underground cable asset and append the example content as an example into the file `impact_level.csv` in the folder `/opt/IBM/energy/data/dno_sample/csv` on the application server.

```
1201,"2014-01-01 08:58:00",4.0
1201,"2013-01-01 08:59:00",4.0
1201,"2012-01-01 09:00:00",3.0
1202,"2014-01-01 08:58:00",2.0
1202,"2013-01-01 08:59:00",3.0
1202,"2012-01-01 09:00:00",4.0
1203,"2014-01-01 08:58:00",2.0
```

```

1203,"2013-01-01 08:59:00",3.0
1203,"2012-01-01 09:00:00",4.0
1204,"2014-01-01 08:58:00",2.0
1204,"2013-01-01 08:59:00",3.0
1204,"2012-01-01 09:00:00",4.0
1205,"2014-01-01 08:58:00",2.0
1205,"2013-01-01 08:59:00",3.0
1205,"2012-01-01 09:00:00",4.0

```

4. Update the list of csv files in the config.properties in the folder /opt/IBM/energy/data/dno_sample/config.

```

csv.1=substation
csv.2=overheadline
csv.3=wind_farm
csv.4=undergroundcable
csv.5=measurement
csv.6=measurement_threshold
csv.7=health_index
csv.8=impact_level
csv.9=load_index
csv.10=measurement_status
csv.11=connection

```

5. Run the command: /opt/IBM/energy/data/runCSVLoader.sh /opt/IBM/energy/data/dno_sample/config/config.properties.

Extending the REST service

Add the new resource to the service configuration file.

Procedure

Add a new section to the REST service xml configuration file.

```

<resource>
    <url>/undergroundcable</url>
    <properties idProperty="ASSET_ID">
        <property name="ASSET_TYPE" type="String" />
        <property name="ASSET_ID" type="Number" />
        <property name="SERIAL_NUMBER" type="String" />
        <property name="NAME" type="String" />
        <property name="DESCRIPTION" type="String" />
        <property name="INSTALLATION_DATE" type="Date" />
        <property name="STATUS" type="Number" />
        <property name="LOCATION" type="Geometry" />
    </properties>
    <query>select 'undergroundcable' as ASSET_TYPE,
ASSET_ID,SERIAL_NUMBER,NAME,DESCRIPTION,INSTALLATION_DATE, STATUS,LOCATION
from dno.undergroundcable</query>
    <create>
        MERGE INTO dno.undergroundcable T
        USING TABLE(VALUE(CAST(:ASSET_ID AS INTEGER),
CAST(:SERIAL_NUMBER AS VARCHAR(128)),
CAST(:NAME AS VARCHAR(128)), CAST(:DESCRIPTION AS VARCHAR(512)),
CAST(:INSTALLATION_DATE AS DATE),
CAST(:STATUS AS SMALLINT), db2gse.st_geomFromText(:LOCATION, 1003)))
        TMP(ASSET_ID, SERIAL_NUMBER, NAME, DESCRIPTION,
INSTALLATION_DATE, STATUS, LOCATION)
        ON T.ASSET_ID = TMP.ASSET_ID
        WHEN NOT MATCHED THEN INSERT(ASSET_ID, SERIAL_NUMBER, NAME, DESCRIPTION,
INSTALLATION_DATE, STATUS, LOCATION)
        VALUES(TMP.ASSET_ID, TMP.SERIAL_NUMBER, TMP.NAME, TMP.DESCRPTION,
TMP.INSTALLATION_DATE, TMP.STATUS, TMP.LOCATION)
        WHEN MATCHED THEN UPDATE SET (SERIAL_NUMBER, NAME, DESCRIPTION,
INSTALLATION_DATE, STATUS, LOCATION)

```



```

        = (TMP.SERIAL_NUMBER, TMP.NAME, TMP.DESCRPTION,
TMP.INSTALLATION_DATE, TMP.STATUS, TMP.LOCATION)
        </create>
</resource>

```

Extending the user interface Procedure

1. Extend the data model. Add following json objects into model.json.

```

{
    "id": "undergroundCableModel",
    "module": "ifeef/model/Model",
    "properties": ["criteria", "store"],
    "parameters": {
        "timer": "@{timer}",
        "storeModule": "ifeef/model/Store",
        "target": "/ibm/ife/sample/dno/api/undergroundcable",
        "idProperty": "ASSET_ID"
    }
}

```

2. Extend the filter. Add a new filter for an underground cable asset type into the file filter.json.

```

{
    "id": "undergroundcable_filter",
    "module": "ifeef/widget/filter/Filter",
    "container": "group1",
    "parameters": {
        "label": "Underground Cable",
        "style": "width: 100%;margin-top: 10px;",
        "enabled": true,
        "pattern": "and",
        "searchFields": [
            {
                "name": "NAME", "type": "string", "label": "@{dno_nls.NAME}"
            },
            {
                "name": "SERIAL_NUMBER", "type": "string",
                "label": "@{dno_nls.Serial_Number}", "regExp": "([A-Za-z0-9][A-Za-z0-9-_]*)"
            },
            {
                "name": "INSTALLATION_DATE", "type": "date",
                "label": "@{dno_nls.Installation_Date}"
            },
            {
                "name": "STATUS",
                "label": "@{dno_nls.STATUS}",
                "type": "number",
                "multiple": true,
                "options": [
                    {
                        "label": "@{dno_nls.Critical}",
                        "icon": "/ibm/ife/widges/icons/critical.png",
                        "value": 2
                    },
                    {
                        "label": "@{dno_nls.Caution}",
                        "icon": "/ibm/ife/widges/icons/critical.png",
                        "value": 1
                    },
                    {
                        "label": "@{dno_nls.Acceptable}",
                        "icon": "/ibm/ife/widges/icons/critical.png",
                        "value": 0
                    }
                ]
            },
            {
                "label": "@{dno_nls.NoReading}",
                "icon": "/ibm/ife/widges/icons/critical.png",
            }
        ]
    }
}

```

```

        "value":3
      }
    ]
  }
}

```

3. Extend the map layer and the preview card.

- a. Add a new map layer for the underground cable into the file map.json.

```

{
  "id": "undergroundCableLayer",
  "module": "ifef/widget/map/DataLayer",
  "container": "map",
  "properties": ["store", "selected", "criteria"],
  "parameters": {
    "index":2,
    "map": "@{map}",
    "keyFields": ["ASSET_TYPE", "ASSET_ID"],
    "styles": [ {
      "condition": "STATUS==0",
      "style": {
        "stroke":{
          "type":"ol.style.Stroke",
          "parameters": {
            "color": "#699037",
            "width": 2
          }
        }
      }
    },
    {
      "condition": "STATUS==1",
      "style": {
        "stroke":{
          "type":"ol.style.Stroke",
          "parameters": {
            "color": "#FDBA1A",
            "width": 2
          }
        }
      }
    },
    {
      "condition": "STATUS==2",
      "style": {
        "stroke":{
          "type":"ol.style.Stroke",
          "parameters": {
            "color": "#C32E14",
            "width": 2
          }
        }
      }
    },
    {
      "condition": "STATUS==3",
      "style": {
        "stroke":{
          "type":"ol.style.Stroke",
          "parameters": {
            "color": "#EE3D96",
            "width": 2
          }
        }
      }
    },
    {
      "condition": "STATUS!=0 && STATUS!=1 && STATUS!=2 && STATUS!=3",
      "style": {

```

```

        "stroke":{
        "type":"ol.style.Stroke",
        "parameters": {
            "color": "#BBBBBB",
            "width": 2
        }
    }
}
]]
}
}
}

```

- b. Add a new map preview card for the underground cable type into the file `previewCard.json`.

```

{
    "id": "undergroundCableMapPreviewCard",
    "module": "ifef/widget/previewcard/PreviewCard",
    "properties":["data"],
    "parameters": {
        "titleProperty": "NAME",
        "position": "_position",
        "properties": [
            {"label":"@{dno_nls.Asset_ID}", "name":"ASSET_ID", "isKey": true},
            {"label":"@{dno_nls.Serial_Number}", "name":"SERIAL_NUMBER", "isKey": true},
            {"label":"@{dno_nls.NAME}", "name":"NAME", "isKey": true},
            {"label":"@{dno_nls.Description}", "name":"DESCRIPTION", "isKey": true},
            {
                "label":"@{dno_nls.STATUS}",
                "name":"STATUS",
                "isKey": true,
                "render":{
                    "templates": [
                        {
                            "condition":"STATUS==0",
                            "content":"<span style='background-color:green'>${Acceptable}"
                            "variables":{
                                "Acceptable":"@{dno_nls.Acceptable}"
                            }
                        },
                        {
                            "condition":"STATUS==1",
                            "content":"<span style='background-color:orange'>${Caution}"
                            "variables":{
                                "Caution":"@{dno_nls.Caution}"
                            }
                        },
                        {
                            "condition":"STATUS==2",
                            "content":"<span style='background-color:red'>${Critical}"
                            "variables":{
                                "Critical":"@{dno_nls.Critical}"
                            }
                        }
                    ]
                }
            },
            {"label":"@{dno_nls.Location}", "name":"LOCATION"}
        ]
    }
}
}
}

```

4. Extend the list and list Container including preview card.

- a. Add a new list for the underground cable asset type in the file `list.json`.

```

{
    "id": "undergroundCableList",
    "module": "ifef/widget/list/List",
    "container": "listContainer",

```

```

"properties": ["store"],
"parameters": {
  "keyFields": ["ASSET_TYPE", "ASSET_ID"],
  "title": "@{undergroundCable_filter.label}",
  "pageSize": 100,
  "paginationBar": [100,150,200,1000,0],
  "baseSort": [
    {
      "attribute": "STATUS",
      "descending": true
    },
    {
      "attribute": "SERIAL_NUMBER",
      "descending": true
    }
  ],
  "column": [
    {
      "name": "@{dno_nls.STATUS}",
      "field": "STATUS",
      "sortable": true,
      "style": "@{statusFormatter.style}",
      "decorator": "@{statusFormatter.decorator}"
    },
    {
      "name": "@{dno_nls.Serial_Number}",
      "field": "SERIAL_NUMBER",
      "sortable": true
    },
    {
      "name": "@{dno_nls.NAME}",
      "field": "NAME",
      "sortable": true
    },
    {
      "name": "@{dno_nls.Installation_Date}",
      "field": "INSTALLATION_DATE",
      "sortable": false
    }
  ]
}
}

```

- b. Add a new preview card for the underground cable type into the file previewCard.json.

```

{
  "id": "undergroundCablePreviewCard",
  "module": "ifef/widget/previewcard/PreviewCard",
  "properties": ["data"],
  "parameters": {
    "titleProperty": "NAME",
    "position": "_position",
    "properties": [
      {"label": "@{dno_nls.Asset_ID}", "name": "ASSET_ID", "isKey": true},
      {"label": "@{dno_nls.Serial_Number}", "name": "SERIAL_NUMBER", "isKey": true},
      {"label": "@{dno_nls.NAME}", "name": "NAME", "isKey": true},
      {"label": "@{dno_nls.Description}", "name": "DESCRIPTION", "isKey": true},
      {
        "label": "@{dno_nls.Installation_Date}",
        "name": "INSTALLATION_DATE",
        "render": {
          "dateFormatOptions": { "selector": "date", "datePattern": "yyyy-MM" }
        }
      }
    ],
    {
      "label": "@{dno_nls.STATUS}",
      "name": "STATUS",

```

```

        "isKey": true,
        "render": {
            "templates": [
                {
                    "condition": "STATUS==0",
                    "content": "<span style='background-color:green'>${Acc",
                    "variables": {
                        "Acceptable": "@{dno_nls.Acceptable}"
                    }
                },
                {
                    "condition": "STATUS==1",
                    "content": "<span style='background-color:orange'>${Ca",
                    "variables": {
                        "Caution": "@{dno_nls.Caution}"
                    }
                },
                {
                    "condition": "STATUS==2",
                    "content": "<span style='background-color:red'>${Criti",
                    "variables": {
                        "Critical": "@{dno_nls.Critical}"
                    }
                }
            ]
        },
        {
            "label": "@{dno_nls.Location}",
            "name": "LOCATION"
        },
    ],
    "moreActions": [
        {
            "label": "@{nls.HighlightOnMap}",
            "func": "@{highlightOnMap.highlightOnMap}"
        }
    ]
}
}
}}

```

- c. Modify `highlightOnMap` to include `undergroundcable` in the file `previewCard.json`

```

{
    "id": "highlightOnMap",
    "module": "ifef/widget/map/HighlightOnMap",
    "parameters": {
        "mapContainer": "@{mapTab}",
        "typeProperty": "ASSET_TYPE",
        "mappings": [
            {
                "type": "overheadline",
                "targetLayer": "@{overheadlineLayer}",
                "dataIdProperty": "ASSET_ID",
                "maxZoom": 16
            },
            {
                "type": "undergroundcable",
                "targetLayer": "@{undergroundCableLayer}",
                "dataIdProperty": "ASSET_ID",
                "maxZoom": 16
            },
            {
                "type": "substation",
                "targetLayer": "@{substationLayer}",
                "dataIdProperty": "ASSET_ID",
                "maxZoom": 16
            },
            {
                "type": "wind_farm",

```

```

        "targetLayer": "@{windFarmModelLayer}",
        "dataIdProperty": "FARM_ID",
        "maxZoom": 16
    }
}
]
}
}

```

5. Extend the logical Map: refer to section of New Application Development for the Logical Map.
6. Extend the binding: modify the file bind.json as in the example:

```

[
  {
    "id": "filterBind",
    "module": "ifef/behavior/Bind",
    "parameters": {
      "bindings": [
        ...
        { "source": "@{undergroundcable_filter}", "sourceProp": "criteria", "target": "@{undergroundcable_filter}" },
        ...
      ]
    }
  },
  {
    "id": "mapBind",
    "module": "ifef/behavior/Bind",
    "parameters": {
      "bindings": [
        ...
        { "source": "@{undergroundCableModel}", "sourceProp": "store", "target": "@{undergroundCableModel}" },
        ...
        { "source": "@{undergroundCableLayer}", "sourceProp": "selected", "target": "@{undergroundCableLayer}" },
        ...
      ]
    }
  },
  {
    "id": "listBind",
    "module": "ifef/behavior/Bind",
    "parameters": {
      "bindings": [
        ...
        { "source": "@{undergroundCableModel}", "sourceProp": "store", "target": "@{undergroundCableModel}" },
        ...
        { "source": "@{undergroundCableList}", "sourceProp": "selected", "target": "@{undergroundCableList}" },
        ...
      ]
    }
  },
  {
    "id": "logicMapBind",
    "module": "ifef/behavior/Bind",
    "parameters": {
      "bindings": [
        { "source": "@{logicMapPreviewCardHelper}", "sourceProp": "selected", "target": "@{logicMapPreviewCardHelper}" }
      ]
    }
  },
  {
    "id": "logicalMapLayoutBind",
    "module": "ifef/behavior/Bind",
    "parameters": {
      "bindings": [
        { "source": "@{logicalMap}", "sourceProp": "layoutData", "target": "@{logicalMapLayout}" }
      ]
    }
  },
  {
    "id": "assetDetailBind",
    "module": "ifef/behavior/Bind",
    "parameters": {
      "bindings": [
        ...
        { "source": "@{undergroundCableLayer}", "sourceProp": "criteria", "target": "@{undergroundCableLayer}" },
        ...
        { "source": "@{undergroundCableList}", "sourceProp": "criteria", "target": "@{undergroundCableList}" },
        ...
      ]
    }
  }
]

```

```
    }  
  }  
]
```

Customizing the solution using the UI and Service Framework

You can customize the solution to suit your business requirements.

Use the Administration Console view to register custom user interface components and to customize the user interface to suit your particular operation.

Customizing the user interface

You can use the extension capabilities that are provided by the user interface framework to build application user interfaces that meet your operational requirements.

UI Framework provides a number of reusable user interface components including pages, styles, layouts, and widgets. You can also add custom widgets and layouts to your solution by using the user interface extension framework. Use the Administration Console view to register new user interface components, to add new pages, and to configure the style, layout, widgets, access controls, and Representational State Transfer (REST) services for the pages in your solution.

Configuring pages

If you are an administrator, you can add and configure custom pages. Each page has a style and a layout to manage the widgets on the page.

About this task

Use the Administration Console view to add and configure custom pages. Each page is defined by a set of properties, and each property value that you enter is validated by the solution upon entry. You can view pages that are provided with the solution, but you cannot delete them, and you can edit only the layout properties for these pages.

Procedure

1. In the Administration Console view, click **Page**.
Adding a custom page
2. Click **Create**.
3. In the Create a Page window, enter the details for your custom page.
 - a. In **Title**, enter a unique page title.
 - b. Optional: In **Description**, enter a description of the page.
 - c. In **URI**, enter a valid URI, for example `/ibm/ife/sample/index/HTML`.
 - d. Click **Save**. The new page is listed under the **Create** button.
 - e. For a newly created page, select **Groups** in the **Access control** field if you want those groups to access this page. Click **Save**.

Editing information for a custom page

4. Edit the page title, description, URI and access control for a custom page.
5. Click **Save** to save the changes to the page's configuration.

Deleting a custom page

6. Click the delete icon for the custom page, and then in the confirmation window, click **Yes**.

Note: You cannot delete a custom page if the page is included in the configuration of a page hierarchy.

Configuring page hierarchies

If you are an administrator, you can add and remove page hierarchies, and you can configure the contents of page hierarchies. Page hierarchies that contain one or more pages are displayed in the main navigation bar.

About this task

Use the Administration Console view to add, edit, and remove page hierarchies. Each hierarchy has a label and can contain one or more pages or page hierarchies. Each element of a page hierarchy is defined by a set of properties, and each property value that you enter is validated by the solution upon entry. The position of a page hierarchy in the main navigation bar is determined by the value of the hierarchy's sequence property. You cannot change the properties of system page hierarchies that are provided with the solution, but you can remove them or change their contents.

Procedure

1. In the Administration Console view, click **Page Hierarchy**.
Adding a page hierarchy
2. Add a label to create a new page hierarchy. You can add a top-level page hierarchy or you can add a page hierarchy to an existing page hierarchy. The label for a page hierarchy is displayed in the main navigation bar.
 - To add a top-level hierarchy, click **Create**.
 - To add a hierarchy in an existing page hierarchy, click the add icon for the page hierarchy.
3. In the Create an Item window, enter the label details for your custom page hierarchy.
 - a. In **Type**, select **Label** in the drop-down list to add a page hierarchy.
 - b. In **Name**, enter a name for the page hierarchy. The name is displayed in the main navigation bar.
 - c. Optional: In **Description**, enter a description of the page hierarchy.
 - d. In **Sequence**, enter a number greater than or equal to 0 as the sequence number for the page hierarchy. The sequence number is a relative value that determines the position of a page hierarchy in relation to other page hierarchies in the main navigation bar. For example, if you have two top-level page hierarchies, then the page hierarchy with the lower sequence value is positioned to the left in the main navigation bar.
 - e. Click **Save**.

Note: A page hierarchy is not displayed in the main navigation bar until it contains one or more pages.

Adding a page to a hierarchy

4. Click the add icon for the page hierarchy.
5. In the Create an Item window, enter the details for your custom page.
 - a. In **Type**, select **Page** from the drop-down list.
 - b. In **Name**, enter the page name to display in the page hierarchy.
 - c. Optional: In **Description**, enter a description of the page.

- d. In **Sequence**, enter a number greater than or equal to 0. The sequence number is a relative value that determines the position of a page in a hierarchy in relation to other pages or page hierarchies.
- e. In **Page definition**, select the page to add to the hierarchy from the drop-down list.
- f. Click **Save**.

Note: You must refresh the browser to display the new page in the main navigation bar.

Editing a page hierarchy

6. Select a page hierarchy to edit the name, description, and sequence for the page hierarchy.
7. Select a page in a page hierarchy to edit the name, description, sequence, and page definition for the page.
8. Click **Save**.

Deleting a page hierarchy

9. Click the delete icon for the page hierarchy, and then in the confirmation window, click **Yes**.

Deleting a page from a page hierarchy

10. Click the delete icon for the page, and then in the confirmation window, click **Yes**.

Configuring REST services

If you are an administrator, you can register and configure custom Representational State Transfer (REST) services.

About this task

Use the Administration Console view to register and configure custom REST services. Each service is defined by a set of properties, and each property value that you enter is validated by the solution upon entry.

Procedure

1. In the Administration Console view, click **Service**.

Registering a custom REST service

2. Click **Create**.
3. In the Create a Service window, enter the details for your custom service.
 - a. In **Name**, enter the name of the service.
 - b. In **URI**, enter the URI for the resource. For example, `/ibm/ife/api/ui-service/style`.

Note: You can append `/*` to the base URI for the service to include all the resources that are managed by the service. For example, `/ibm/ife/api/ui-service/*`.

- c. Optional: In **Description**, enter a description of the service.
- d. Click **Save**. The new service is listed under the **Create** button. Edit the service to assign access rights to the service to user groups in your solution.

Editing a custom REST service

4. Select the service and edit the values.
 - a. Edit the values for the name, URI, and description.

- b. For **Access Control** , assign access rights to the service for the user role groups in your solution. For each user role group, you can select one or more of the access controls that are labeled **Create**, **Read**, **Update**, and **Delete**. For more information about user role groups, see the related link.
- c. Click **Save** to save the changes.

Deleting a custom REST service

5. Click the delete icon for the service, and then in the confirmation window, click **Yes**.

Configuring notifications

To be able to receive notifications, the user must configure the notifications in the administration page.

After Notifications are set up, the user is then able to receive notifications via the end user page, email or an SMS text message on a mobile communications system.

Making a subscription for Notification in Situational Awareness

After Notification is set up you can receive Notification on your system dashboard. There are two extra methods for the user to receive notification, email and SMS text message.

About this task

On the Notification admin page, you can make a subscription for the system notification .

Procedure

1. On the user interface, click **Administration > Notification > Allert Settings**.
2. Click **System Message > Item Correlation**.
3. In the fields System Dashboard, email, SMS, you can select the check-box Receive notifications for the method for receiving system notifications. You can also select how long a message shows in your System Dashboard.
4. Click **Save**.

Configuring the application

UI Framework provides the capability to configure the user interface, focusing on the filter bar and the connectivity filter.

Configuring Time in the filter bar

You can configure the options for the time filter in the table AHA.ANALYSIS_YEAR.

The options for the time filter are derived from the configuration of year analysis table AHA.ANALYSIS_YEAR.

TIME_BUCKET

Integer, is the yearly interval to analyze the given scope.

ANALYSIS_DURATION

Integer, the number of years to be analyzed from the start year.

ANALYSIS_YEAR

Integer, the start year to take analysis as nnnn.

UI Framework

The user interface framework accelerates the development of an application on the front end.

The UI framework consists of several parts:

- Bootstrap
 - Is based on HTML/JSP bootstrap
 - Calls runtime to load the configuration files.
 - Responsible for loading the javascript libraries and the cascading style sheets.
- Configuration
 - Is a JSON format configuration file.
 - Configures the widget parameters.
 - Configures the widget containment.
 - Configures the dependency injection.
- Runtime Library
 - Is a lightweight javascript library.
 - Enables the API to load configuration file and initialize UI widget.
 - Enables the API to dynamically add/remove widgets.
- Ready-to-use widgets - consists of:
 - Container widgets (BorderLayout, TabContainer, ContentPane,)
 - Functional widgets (Map, List, Chart,)
 - Data Model widgets
 - Property binding widgets
 - Behavior widgets
- Ready-to-use application template - The template includes html bootstrap code, the configuration file and custom widgets.

Bootstrap

Bootstrap is an jsp file, that loads the style sheet, loads and initializes the JSP libraries, defines the page level style and calls the runtime to load the configuration.

User can start from below default bootstrap as a template:

- Import custom javascript libraries (e.g. jquery, react.js, etc...)
- Import custom stylesheet

Configuration

The configuration is managed by one JSON file that is loaded by Runtime and includes other required JSON files.

Each item inside configuration file is composed of:

- id - a unique id of the widget.
- module - the module name the widget.
- container - an optional container of the widget.
- regions - an optional list of regions if the widget is a container.
- parameters - optional parameters for the widget.

Example one - Configure Widget Containment

The widget containment relationship represents a parent - child relationship between the widgets. There is a root container that contains everything to be displayed. The root container widget is contained by a DOM node in the Bootstrap file. The example shows that div id = "root" is the division and can be used as the root container.

```
<body class="oneui">
  <div id="root"></div>
  <script>
    require(["ifef/Runtime.js"], function(Runtime){
      window.rt = Runtime;
      Runtime.load("config3.json").then(function(widgets){
        //console.debug("Success",widgets);
      });
    });
  </script>
</body>
```

Figure 33. The division as the root container

Children widgets are contained by a container widget in a specified region. The example shows the left widget as part of the main widget and is located in the left region.

```
10 [
11 {
12   "id": "main",
13   "module": "idx/layout/BorderContainer",
14   "container": "root",
15   "regions": ["top", "left", "right", "center", "trailing", "bottom"],
16   "parameters": {
17     "style": "width: 100%; height: 100%;"
18   }
19 },
20 {
21   "id": "left",
22   "module": "dijit/layout/ContentPane",
23   "container": "main",
24   "parameters": {
25     "region": "left",
26     "splitter": "toggle",
27     "minSize": 20,
28     "style": "width: 20%;padding: 0;"
29   }
30 },
31 {
32   "id": "center",
33   "module": "dijit/layout/ContentPane",
34   "container": "main",
35   "parameters": {
36     "region": "center",
37     "minSize": 20,
38     "style": "width: 60%;padding: 0;"
39   }
40 }
41 ]
```

Figure 34. Region container widget

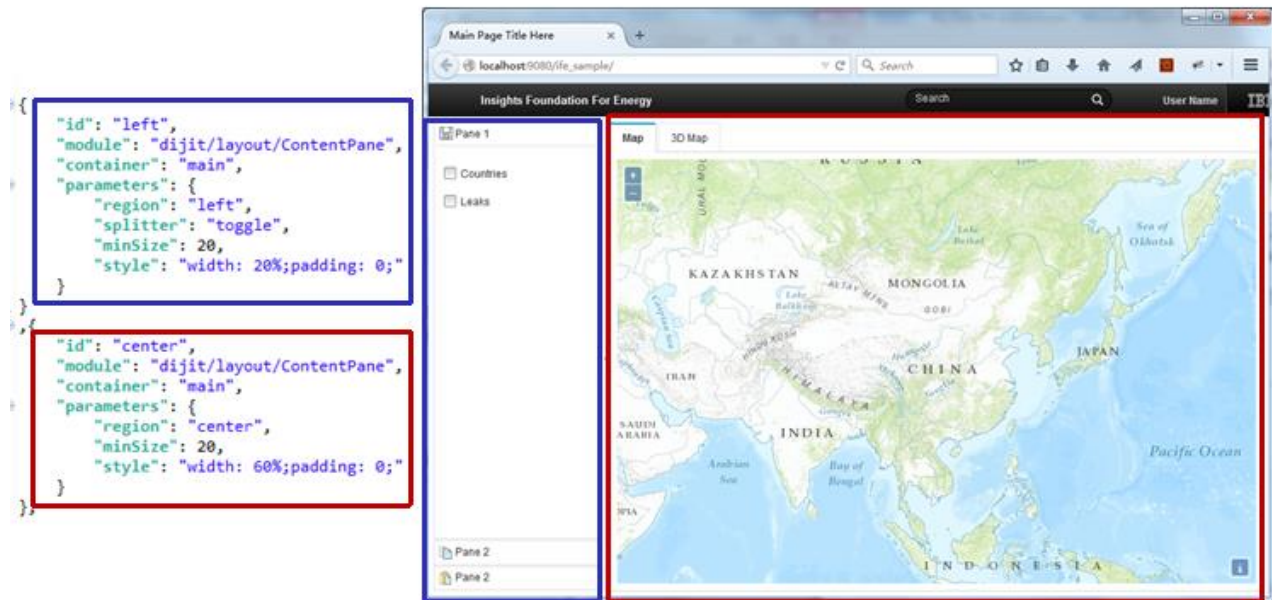


Figure 35. Result for left and center containment

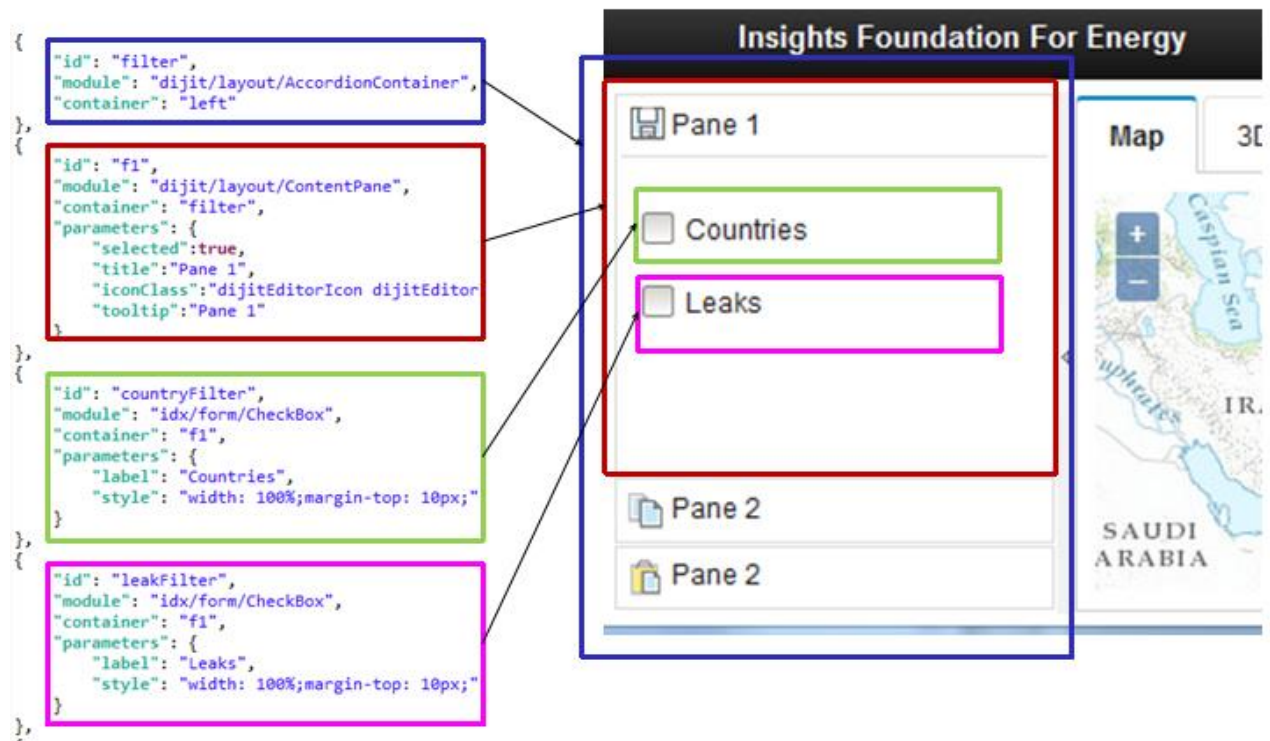


Figure 36. Filter Panes are contained by left widget

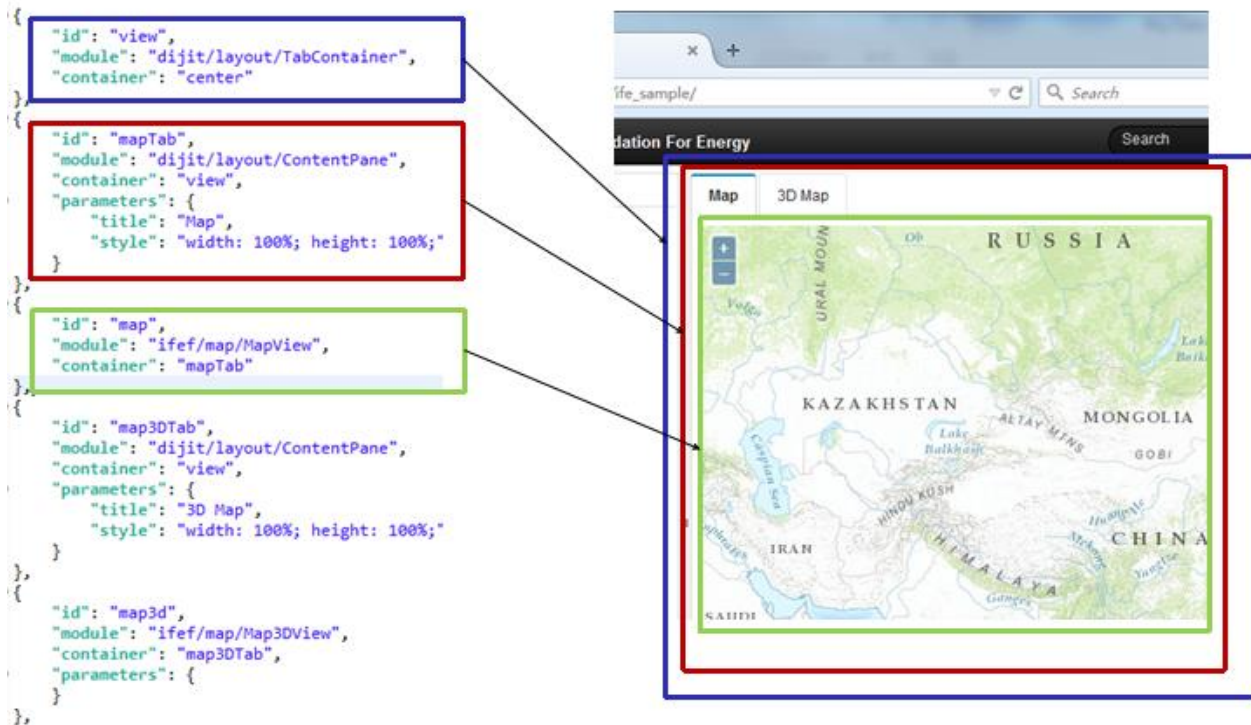


Figure 37. Map & 3D Map contained by center widget

Example two - Configure Dependency Injection

Dependency represents the interaction relationship among different widgets. Dependency injection is the way of UI framework to configure this kind of relationship in JSON format configuration file. The example shows that the countryLayer module has the map and the countryFilter dependency is injected.

```

{
  "id": "countryLayer",
  "module": "ifef/behavior/GeoJsonLayer",
  "parameters": {
    "data": "data/countries.geojson",
    "map": "@{map}",
    "filter": "@{countryFilter}"
  }
}

```

```

{
  "id": "map",
  "module": "ifef/map/MapView",
  "container": "mapTab"
},

```

```

{
  "id": "countryFilter",
  "module": "idx/form/CheckBox",
  "container": "f1",
  "parameters": {
    "label": "Countries",
    "style": "width: 100%;margin-top: 10px;"
  }
},

```

Figure 38. The countryLayer module has {map} and {countryFilter} is the dependency

Runtime library

The Runtime library enables the API to dynamically manipulate UI configurations.

The commands for the Runtime Library are:

add: add new widget(s)

Parameters:

- widget: a widget configuration or an array of widget configurations, a widget instance or array of widget instances.
- container: optional, the ID of the container where the widgets are added.
- region: optional, the name of region of the container.

Returns: a dojo/Deferred object.

remove: remove widget

Parameter:

- id: The ID of the widget to be removed.

Returns: the array of the specified widget and its children that are removed. The user can reattach them to another container or region by calling an add command, or totally remove the widgets.

get: get widget instance

Parameter:

- id: the ID of the widget.

Returns: the widget instance of the specified ID.

load: load widget from file

Parameters:

- `url`: the address of the configuration file.

Returns:

- `dojo/Deferred` object.

Note:

- The Runtime library also maintains a global registry of widgets, the contents of the registry changes when user calls an `add`, `remove`, or `load` command.
- The Runtime library can be used in both bootstrap and custom widgets.

```
<script>
  define([
    "dojo/_base/declare", // declare
    "dojo/_base/lang",
    "dojo/Stateful",
    "ifef/Runtime"
  ], function(declare, lang, Stateful, Runtime){
  });
</script>
```

Figure 39. Bootstrap and custom widgets

Ready-to-use widgets

Ready-to-use widgets are widgets which can be used directly.

There are several types of widgets listed:

- Container widgets
 - Dojo and IDX container widgets are available to use. The following list contains the most common container. For other IDX container, please refer to IDX documentation.
- Reusable functional widgets: IFE provided widgets for a specific functionality:
 - Filter
 - Preview Card
 - Map
 - Data Layer
 - Logical Map
 - List
 - List Container
 - LineChart
 - BarChart
 - HeaderButton
 - Timer
- Behavior widgets
 - Property Binding Widgets. Property binding depends on the mechanism of getting, setting, and watching for property changes, we leverage **dojo/Stateful** and **dijit/_WidgetBase** to provide this mechanism. For further information refer to Stateful and WidgetBase in the Dojo Toolkit Reference Guide.
The example shows the binding from the **checked** property of **countryFilter** to **filterSelected** property of **countryLayer**. When user checks the checkbox,

countryLayer is notified: the `_filterSelectedSetter` method of countryLayer will be called to execute the corresponding actions:

```
118 {
119   "id": "bind1",
120   "module": "ifef/behavior/Bind",
121   "parameters": {
122     "bindings": [
123       {"source": "countryFilter", "sourceProp": "checked", "target": "countryLayer", "targetProp": "filterSelected"}
124     ]
125   }
126 }
127 ]
```

Figure 40. The `filterSelected` method

```
1 define(["dojo/_base/declare", "dojo/_base/lang", "dojo/Stateful"],
2 function(declare, lang, Stateful){
3   // module:
4   //   ifef/behavior/CountryLayer
5   return declare("ifef.hebehavior.GeoJsonLayer", [Stateful], {
6     data: null,
7
8     _filterSelectedSetter: function(selected/*boolean*/){
9       if(selected){
10        this.layer = this.layer ? this.layer : new ol.layer.Vector({
11          source: new ol.source.Vector({
12            format: new ol.format.GeoJSON(),
13            url: this.data
14          })
15        });
16        this.map.map.addLayer(this.layer);
17      } else {
18        if(this.layer){
19          this.map.map.removeLayer(this.layer);
20        }
21      }
22    },
23  },
24  ])
```

Figure 41. Execute action

- Special Usage for Binding Widget
bindOnlyIfUnequal: only notify the target if the old value is not equal to new value.

```

{
  "id": "bind1",
  "module": "ifef/behavior/Bind",
  "parameters": {
    "bindings": [
      {
        "source": "filter1",
        "sourceProp": "checked",
        "target": "countryLayer",
        "targetProp": "filterSelected",
        "bindOnlyIfUnequal": true
      }
    ]
  }
}

```

Figure 42. Notifies the target if the two values do not agree

converter and func: function of injected converter will be called before setting the value to the target. The functions give a conversion between different formats.

```

{
  "id": "bind1",
  "module": "ifef/behavior/Bind",
  "parameters": {
    "converter": "@{converter1}",
    "bindings": [
      {
        "source": "filter1",
        "sourceProp": "checked",
        "target": "countryLayer",
        "targetProp": "filterSelected",
        "bindOnlyIfUnequal": true,
        "func": "conver"
      }
    ]
  }
}

```

Figure 43. Converter and function

- Data Model widgets
 - Model
 - The Data Model widget wraps the service created by the service framework. Two parameters should be specified for it:
 - target: The URL address of the service.
 - idProperty: The ID property of the response value.

```

{
  "id": "modell",
  "module": "ifef/model/Model",
  "properties": ["criteria", "store"]
  "parameters": {
    "target": "/url/to/rest/service",
    "idProperty": "id"
  }
}

```

Figure 44. ID property of the response value

The Data Model widget provides two properties for binding:

criteria: the filter criteria.

store: the dojo store that provides access to data that is applied to the filter criteria, where filter1 is the filter widget which provide filter criteria, and layer1 is the map layer widget that consumes the dojo store:

```

{
  "id": "bind1",
  "module": "ifef/behavior/Bind",
  "parameters": {
    "bindings": [
      { "source": "filter1", "sourceProp": "criteria", "target": "modell", "targetProp": "criteria" },
      { "source": "modell", "sourceProp": "store", "target": "layer1", "targetProp": "store" }
    ]
  }
}

```

Figure 45. Provides access to data

– ModelSelector

The ModelSelector widget is used to dynamically determine the model at the runtime. Its parameter is **models** that lists each key associated with each respective model. The following figure is an example:

```

{
  "id": "readingSelector",
  "module": "ifef/model/ModelSelector",
  "properties": ["selector", "store"],
  "parameters": {
    "models": [
      { "key": "Load_Index", "model": "@{loadIndexModel}" },
      { "key": "Health_Index", "model": "@{healthIndexModel}" },
      { "key": "Impact_Level", "model": "@{impactLevelModel}" }
    ]
  }
}

```

Figure 46. The Models parameter with associated keys

ModelSelector widget provides two properties available for binding:

selector: the dojo object consists of a key and filter criteria.

store: the dojo store that provides access to data that is applied according to the selector. In the example, the selection in the measurement list triggers the

change of reading list.

```
{ "source": "@{measurementList}", "sourceProp": "selector", "target": "@{readingSelector}", "targetProp": "selector"},  
{ "source": "@{readingSelector}", "sourceProp": "store", "target": "@{readingList}", "targetProp": "store"},
```

Figure 47. Provides access applied according to the selector

Reusable widgets

Widgets are provided that can be modified to the needs of the user.

The parameters for each of the widgets are set by the user, the properties set how the widget displays in the user interface.

Filter widget:

The filter widget is used to organize the filter criteria for the result of IFE REST service.

The widget is made up of a check box and an additional dialog box.

ifef.widget.filter.Filter

label:

String, shows as the label for the check box, e.g. Name.

enabled:

Boolean, specifies whether the criteria is enabled or not.

searchFields:

An array, describes the search fields in the more dialog filter:

- name: String, the name of the field.
- type: String, the type of field. Supports three string types: string, number, date.
- label: String, label of the field to show in the more dialog box
- regExp: String, regular expression that validates the user input value. Available only when the type is string and 'options' is not defined.

options:

An array, predefined values that the user can select:

- label: the label shows from a drop-down list.
- value: predefined values

The properties

The properties are:

criteria:

An object that describes filter criteria. The property is changed when the user either clicks a check box or the **More Filter** dialog box is changed.

Example

The example shows the use of a filter widget.

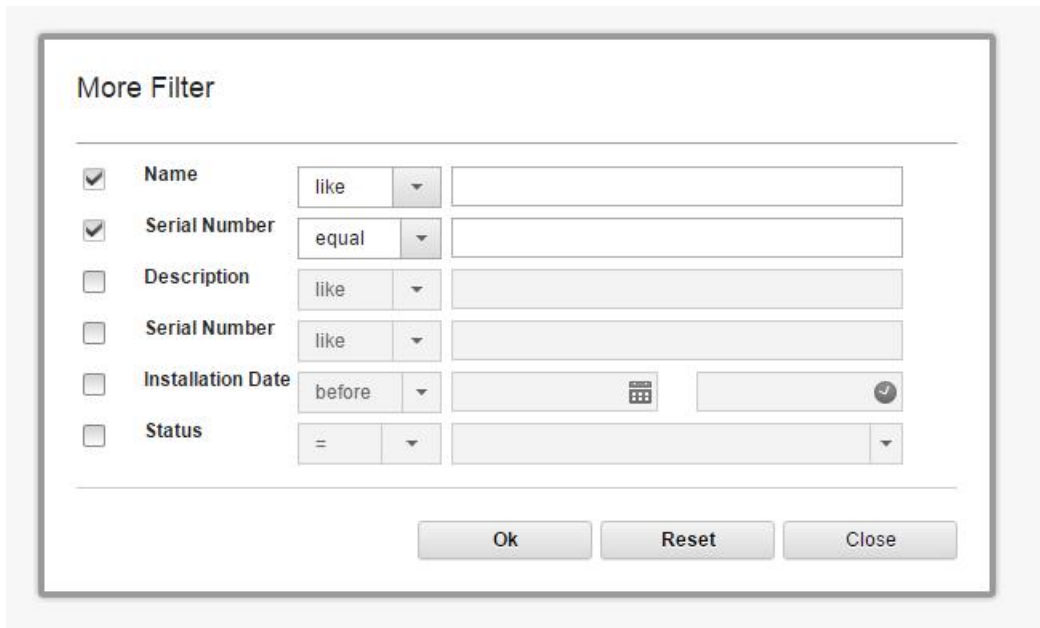


Figure 48. Example filter widget

```

{
  "id": "filter1",
  "module": "ifef/widget/filter/Filter",
  "container": "group1",
  "properties": ["criteria"],
  "parameters": {
    "label": "Filter 1",
    "enabled": true,
    "logistic": "and",
    "searchFields": [
      { "name": "name", "type": "string",
        "label": "Name", "regExp": "([A-Za-z0-9][A-Za-z0-9-_]*)" },
      { "name": "createTime", "type": "date", "label": "Create Time" },
      { "name": "updateTime", "type": "dateTime", "label": "Update Time" },
      { "name": "isActive", "type": "boolean", "label": "Active" },
      {
        "name": "status", "label": "Status", "type": "string", "multiple": true,
        "options": [
          { "label": "Critical", "value": "1" },
          { "label": "Warning", "value": "2" },
          { "label": "Ok", "value": "3" },
          { "label": "NoScore", "value": "4" } ]
      }
    ]
  }
}

```

PreviewCard widget:

The PreviewCard is a temporary card that shows the details panel for an asset, with the ability to select a **More Details** dialog box and an **More Actions** menu.

The user can select an asset from the map, then preview the details on a preview card with the information of asset.

ifef.widget.previewcard.PreviewCard

titleProperty:

String, the name of property in the data object. The property value shows as the title of preview card.

position:

A String or Array that specifies the position of the preview card. If it is a string, then the string content is the property name Position in the data object. The property name Position is defined by an array of two numbers in the data object or a domNode. If it is an array with two numbers, the preview card shows in the fixed position. The first number of array specify x value and the second number specify y value of current page.

properties:

An array that specifies the properties that show on the preview card content panel or more details dialog box.

- name: String, the property name.
- label: String, label of property that shows on content panel.
- isKey: Boolean, if true this property shows on content panel, if false the property shows on the **More details** dialog box.
- index: Numeric, if an index is specified, shows the indexed order.
- render: Object, defines the rules to render the property value:
 - numberFormatOptions: Object, shows the options to format a number as a string. Refer to the dojo/number document.
 - dateFormatOptions: Object, shows the options to format a date as a string. Refer to the dojo/date/locale document.
- templates: Array, defines the conditions and template strings to render property values.
 - condition: String, a condition returns either a true or false status. Any property in data can be referenced as a condition, for example: STATUS=1
 - content: String, an html segment template that contains variables.
 - variables: Object, key or value pairs used to replace the variables in the content.

moreActions:

Array, shows the **More actions** menu

- label: String, shows the name of the menu item.
- children: Array, defines the sub-menu items.
- func: Function, invoke when you click the menu item. Available when sub-menu items are not specified.
- options: Object, used by the 'func' function.

The properties

The properties are:

data:

A key or value object, when the data is set the review card is filled by the data object, and shows.

Example

The example shows the use of a preview card.

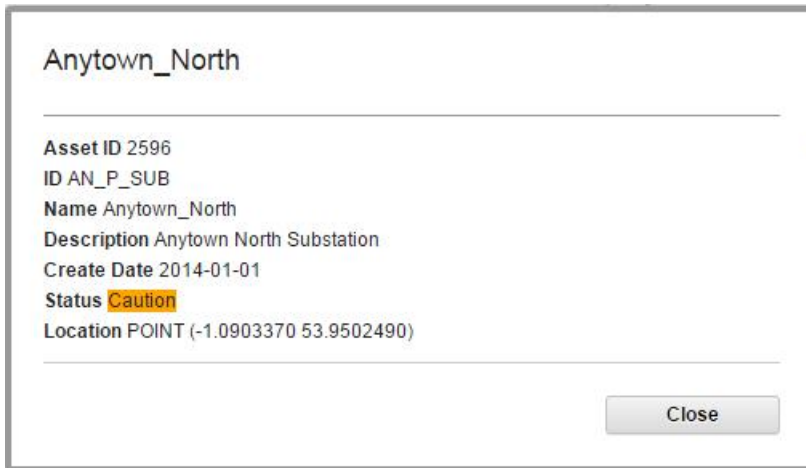


Figure 49. Preview card

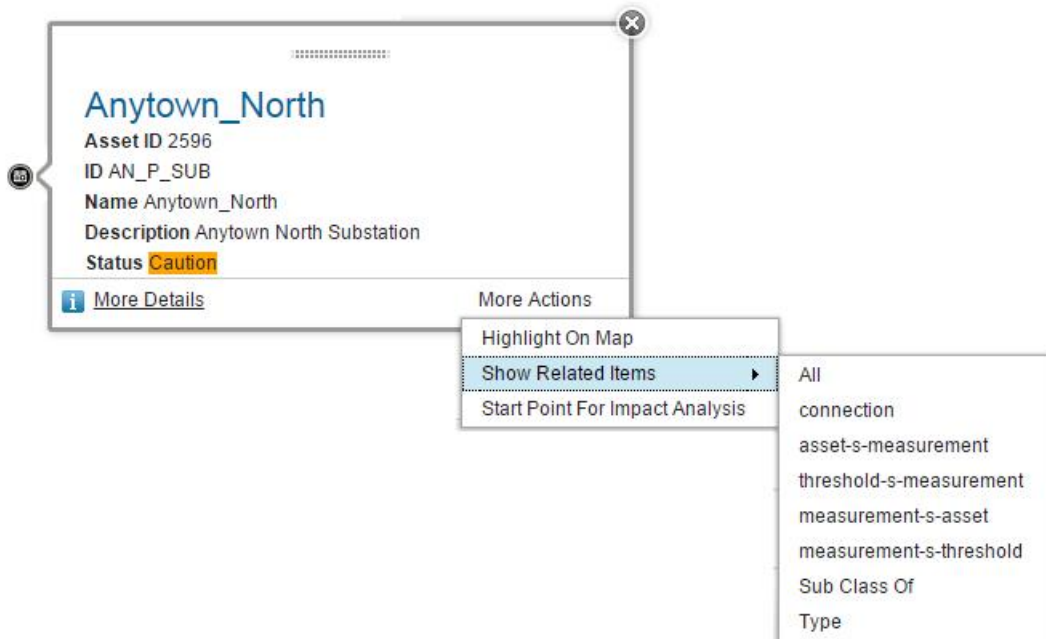


Figure 50. Preview card with more details and more action dialog boxes

```

{
  "id": "overheadlineMapPreviewCard",
  "module": "ifef/widget/previewcard/PreviewCard",
  "properties": ["data"],
  "parameters": {
    "titleProperty": "NAME",
    "position": "_position",
    "properties": [
      {"label": "@{dno_nls.Asset_ID}", "name": "ASSET_ID", "isKey": true},
      {"label": "@{dno_nls.Serial_Number}", "name": "SERIAL_NUMBER", "isKey": true},
      {"label": "@{dno_nls.NAME}", "name": "NAME", "isKey": true},
      {"label": "@{dno_nls.Description}", "name": "DESCRIPTION", "isKey": true},
      {
        "label": "@{dno_nls.STATUS}",
        "name": "STATUS",
        "isKey": true,
      }
    ]
  }
}

```

```

        "render":{
"templates": [
    {
        condition:"STATUS==0",
        content:"<span style='background-color:green'>${Acceptable}</span>",
        variables:{"Acceptable":"@{dno_nls.Acceptable}" }
    },{
        "condition":"STATUS==1",
        "content":"<span style='background-color:orange'>${Caution}</span>",
        "variables":{"Caution":"@{dno_nls.Caution}" }
    ]
    }},
    {"label":"@{dno_nls.Location}","name":"LOCATION"
},
"moreActions":[ { "label":"@{nls.HighlightOnMap}",
"func":"@{highlightOnMap.highlightOnMap}" } ]
}
}

```

Map widget:

The administrator can configure the map controls that show on map for the user.

The administrator can what zoom and positing controls show on a map for the user. The administrator can also configure multiple base layers with the control parameters and attributes for each base layer.

ifef.widget.Map

The parameters for the mapwidget are:

controls:

The administrator can configure which map controls show on a map.

- type: Possible values are "ol.control.Zoom", "ol.control.ZoomSlider", "ol.control.FullScreen", "ol.control.MousePosition", "ol.control.OverviewMap", "ol.control.Rotate", "ol.control.ScaleLine"
- parameters: each parameter has a shows a different type of mapping. See the type and parameters mapping table bellow:

Table 23. Settings for the control parameter

Type	Parameters	Description
ol.control.Zoom	zoomInLabel: Text label used for the zoom in button. Default is +. zoomOutLabel: Text label to use for the zoom out button. Default is - zoomInTipLabel: Text label to use for the button tip. Default is Zoom in. zoomOutTipLabel: Text label to use for the button tip. Default is Zoom out.	A control with 2 buttons, one for zoom in and one for zoom out. This control is one of the default controls of a map. default is show.
ol.control.ZoomSlider:	nul	A slider type of control for zooming in and out.

Table 23. Settings for the control parameter (continued)

Type	Parameters	Description
ol.control.FullScreen:	<p>label: Text label to use for the button. The default is an arrow.</p> <p>tipLabel: Text label for the button tip. Default is Toggle full-screen.</p> <p>ol.control.MousePosition: A control to show the 2D coordinates of the mouse cursor.</p>	<p>A button that when clicked fills the screen with the map. When in full screen mode, a close button shows to exit full screen mode.</p>
ol.control.OverviewMap:	<p>collapsed: Whether the control should start collapsed or expanded. Default to true.</p> <p>collapseLabel: Text label to use for the expanded overviewmap button. Default is «.</p> <p>collapsible: Whether the control can be collapsed or not. Default to true.</p> <p>label: Text label to use for the collapsed overviewmap button. Default is ».</p> <p>layers: Layers for the overview map. tipLabel: Text label to use for the button tip. Default is Overview map</p>	<p>Create a new control with a map acting as an overview map for an other defined map</p>

view

The administrator can configure the view portion of the map.

- **center:** An array of numbers representing an xy coordinate [pointX, pointY]. This is the initial center for the view. The coordinate system for the center is **EPSG:4326**
- **zoom:** Only used if the resolution is not defined. The zoom level is used to calculate the initial resolution for the view. The initial resolution is determined using the **ol.View#constrainResolution** method.
- **extent:** An array of numbers representing an extent: [minx, miny, maxx, maxy]. The extent constrains the center, the center cannot be set outside the extent. The default is undefined. The coordinate system for the extent is **EPSG:4326**.
- **maxZoom:** The maximum zoom level used to determine the resolution constraint. It is used together with minZoom (or maxResolution) and zoomFactor. Th default is 28.

Note: If **minResolution** is provided, it has precedence over **maxZoom**.

- **minZoom:**The minimum zoom level used to determine the resolution constraint. It is used together with maxZoom (or minResolution) and zoomFactor. Default is 0. Note that if maxResolution is also provided, it is given precedence over minZoom.

layers

The administrator can configure multiple base map layers.

- **visible:** boolean type, the default is true.
- **title:** The title of the layer, you can view it on the map control.
- **type:** The possible values are **ol.source.OSM**, **ol.source.MapQuest**, **ol.source.stamen**, **ol.source.xyz**, **ol.source.BingMaps**, **ol.source.TileArcGISRest**, **ol.source.TileJSON**, **ol.source.TileWMS**. Different layer types use different parameters.
- **parameters:** The object type, the parameters it is different according to the type you select. Parameters is the same with openlayer3. See the type and parameters mapping table bellow.

Table 24. Type and parameter mapping

Type	Description
ol.source.OSM	Optional.. A URL template that includes {x}, {y} or {-y}, and {z} placeholders. The default is <code>http://{a-c}.tile.openstreetmap.org/{z}/{x}/{y}.png</code> .
ol.source. MapQuest	Layer. The possible values are: osm, sat, and hyb. A URL template that includes {x}, {y} or {-y}, and {z} placeholders. A value for Layer is mandatory, the URL is optional.
ol.source.stamen	Layer. The possible values are watercolor, terrain-labels. A URL template that includes {x}, {y} or {-y}, and {z} placeholders. A value for Layer is mandatory, the URL is optional.
ol.source. BingMaps	The Bing map key is mandatory, an example key is: Ak-dzM4wZjSqTlzveKz5u0d4IQ4bRz VI309GxmkgSVr1ewS6iPSrOvOKhA-CJlm3 The options for the imagerySet: are: Road, Aerial, AerialWithLabels, collinsBart, and ordnanceSurvey.
ol.source.xyz	A URL template that includes {x}, {y} or {-y}, and {z} placeholders.
ol.source. TileArcGISRest	Call for an ArcGIS REST service URL for a Map Service or Image Service. The url should include either /MapServer or /ImageServer.

Table 24. Type and parameter mapping (continued)

Type	Description
ol.source. TileJSON	The URL link to the address to the TileJSON file.
ol.source. TileWMS	WMS service URL. serverType: The available server types are: 'carmentaserver, geoserver, mapserver, qgis. params: WMS request parameters. A LAYERS parameter is mandatory. STYLES is ' ' by default. VERSION is 1.3.0 by default. WIDTH, HEIGHT, BBOX and CRS (SRS for WMS version < 1.3.0) are set dynamically.
ol.source.WMTS	url: A URL for the service. For KVP encoding, it is a normal URL. For the RESTful request encoding, this is a URL template of the pattern {?-?}, for example: subdomain{a-f}.domain.com. attributions: Attributions. layer: Layer name as advertised in the WMTS capabilities. Mandatory. matrixSet: Matrix set. Mandatory. format: Image format. Default is image/jpeg. projection: Projection. tileGrid: Tile grid. Set the grid pattern for sources accessing WMTS tiled-image servers.

Example

The example shows the use of a map widget.

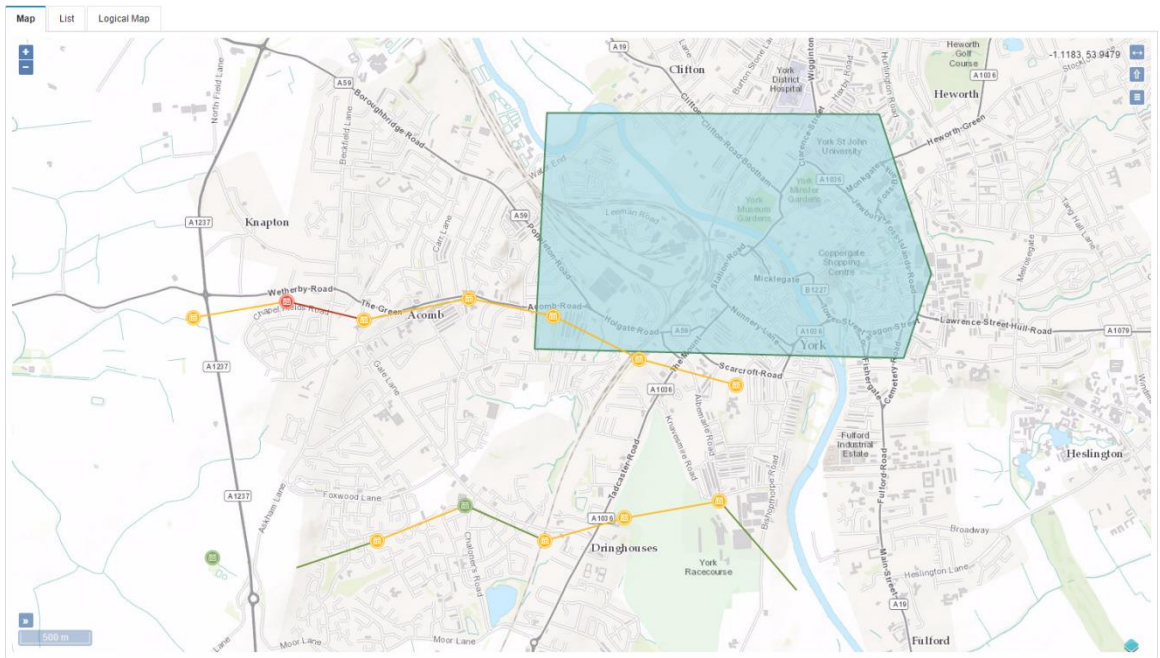


Figure 51. Map widget

```

{
  "id": "map",
  "module": "ifef/widget/Map",
  "container": "mapTab",
  "parameters": {
    "controls": [
      {
        "type": "ol.control.Zoom",
        "parameters": {
          "zoomInTipLabel": "@{nls.Map_Zoom_In_Tip_Label}",
          "zoomOutTipLabel": "@{nls.Map_Zoom_Out_Tip_Label}"
        }
      },
      {
        "type": "ol.control.ZoomSlider",
        "parameters": {}
      },
      {
        "type": "ol.control.ScaleLine",
        "parameters": {
          "minWidth": "64",
          "units": "metric"
        }
      },
      {
        "type": "ol.control.FullScreen",
        "parameters": {
          "label": "\u2194",
          "tipLabel": "@{nls.Map_Full_Screen_Tip_Label}"
        }
      },
      {
        "type": "ol.control.Rotate",
        "parameters": {
          "autoHide": false,
          "tipLabel": "@{nls.Map_Rotate_Tip_Label}"
        }
      },
      {
        "type": "ol.control.OverviewMap",

```

```

        "parameters":{
            "tipLabel":"@{nls.Map_Overview_Tip_Label}"
        }
    },
    {
        "type":"ol.control.MousePosition",
        "parameters":{
            "projection": "EPSG:4326"
        }
    }
],
"layers":[
    {
        "type":"ol.source.OSM",
        "visible":false,
        "title":"@{nls.Map_Layer_Title_OSM}",
        "parameters":{}
    },
    {
        "type":"ol.source.MapQuest",
        "visible":false,
        "title":"@{nls.Map_Layer_Title_MapQuest_Sat}",
        "parameters":{
            "layer": "sat"
        }
    },
    {
        "type":"ol.source.Stamen",
        "visible":false,
        "title":"@{nls.Map_Layer_Title_Stamen}",
        "parameters":{
            "layer": "watercolor"
        }
    },
    {
        "type":"ol.source.XYZ",
        "visible":true,
        "title":"@{nls.Map_Layer_Title_XYZ_ArcgisOnline}",
        "parameters":{
            "url":"http://server.arcgisonline.com/ArcGIS/rest/
Vr1ewS6iPSr0vOKhA-CJlm3",
            "imagerySet": "Road"
        }
    },
    {
        "type":"ol.source.TileArcGISRest",
        "visible":false,
        "title":"@{nls.Map_Layer_Title_TileArcGISRest}",
        "parameters":{
            "url":"https://services.arcgisonline.com/arcgis/rest/
services/ESRI_Imagery_World_2D/MapServer"
        }
    },
    {
        "type":"ol.source.TileJSON",
        "visible":false,
        "title":"@{nls.Map_Layer_Title_TileJSON}",
        "parameters":{
            "url":"http://api.tiles.mapbox.com/v3/

```

```

mapbox.geography-class.jsonp"
    },
    {
      "type": "ol.source.TileWMS",
      "visible": false,
      "title": "@{nls.Map_Layer_Title_TileWMS}",
      "parameters": {
        "url": "http://demo.boundlessgeo.com/geoserver/wms",
        "params": {"LAYERS": "ne:ne"}
      }
    },
    {
      "type": "ol.source.WMTS",
      "visible": false,
      "title": "@{nls.Map_Layer_Title_WMTS}",
      "parameters": {
        "url": "http://services.arcgisonline.com/arcgis/rest/
          services/Demographics/USA_Population_Density
          /MapServer/WMTS/",
        "layer": "0",
        "matrixSet": "EPSG:3857",
        "format": "image/png",
        "projection": "EPSG:3857",
        "style": "default",
        "wrapX": "true",
        "tileGrid": {
          "type": "ol.tilegrid.WMTS",
          "parameters": {
            "origin": [-20037508.342789244,
              20037508.342789244],
            "resolutions": [156543.03392804097,
              78271.51696402048,
              39135.75848201024,
              19567.87924100512,
              9783.93962050256,
              4891.96981025128,
              2445.98490512564,
              1222.99245256282,
              611.49622628141,
              305.748113140705,
              152.8740565703525,
              76.43702828517625,
              38.21851414258813,
              19.109257071294063],
            "matrixIds": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
              11, 12, 13]
          }
        }
      }
    }
  ],
  "view": {
    "zoom": 5,
    "maxZoom": "4",
    "center": [104.31, 28.72],
    "extent": [-180, -80, 180, 80]
    "minZoom": "8"
  }
}

```

DataLayer widget:

The Datalayer widget is used to show a data layer on the map.

The administrator can configure additional data layers. The data can come from a REST service or another data file in standard format. The formats that are currently supported are: GeoJSON, TopoJSON, EsriJSON, IGC, Polyline, WKT, GML, GPX, KML, OSMXML.

You can specify the data type in the Datalayer widget with the `formatType` parameter.

You can specify the data link in the ifef/model/Model widget by the `target` parameter.

You can configure the style for each data layer and set a condition for each style. Each data layer can be set for a custom range, fill, stroke and shape. You can also set text styles for each layer. The user can set the order of the layers.

Container

The container for the `DataLayer` must be a map.

Properties

Store

You can retrieve stored data.

criteria

An object that describes the filter criteria, this property is changed when the user click selects a check box or changes the more filter. If you bind this property with a model and the model is bound to a filter, you can use this properties to control how a layer is displayed or not.

Selected

Indicates that a user has selected an asset in the data layer. An asset can also be selected by the settings the `selected` properties value.

Parameters for `ifef.widget.DataLayer`

map:

The map ID that the data layer is dependent on. For example: `@{map}`.

index:

The order of the data layers. The greater the value or the layer the higher the layer.

formatType:

If data comes from a REST service, then this parameter setting is optional. The formats that are supported are: GeoJSON, TopoJSON, EsriJSON, IGC, Polyline, WKT, GML, GPX, KML, and OSMXML. You can make a selection based on your service type.

geometryName:

The parameter is required when the data is from the REST service and geometry field Name is not LOCATION. Default is "LOCATION".

Styles:

The user can configure multiple styles.

- Condition - you can set the condition that uses the style.
- Style:
 - fill: Set the fill style for vector features.

type: "ol.style.Fill"

parameters: same as the openlayer "ol.style.Fill" class API.

color: Colors can be defined as strings as rgb (r,g,b) or rgba (r,g,b,a) formats, or in hex #rrggbb or #rgb format. The color names, 'red', 'blue' or 'green', can be used with the Canvas renderer. Default null; if null, the Canvas/renderer default black is used.

- stroke: Set the stroke style for vector features.

type: "ol.style.Stroke"

parameters: same as the openlayer "ol.style.Stroke" class API.

Table 25. Stroke parameters

Parameter	Description
color:	Colors can be defined as strings as rgb (r,g,b) or rgba (r,g,b,a) formats, or in hex #rrggbb or #rgb format. The color names, 'red', 'blue' or 'green', can be used with the Canvas renderer. Default null; if null, the Canvas/renderer default black is used.
lineCap	Line cap style: butt, round, or square. Default is round.
lineJoin:	Line join style: bevel, round, or miter. Default is round.
lineDash:	Line dash pattern. Default is undefined (no dash).
miterLimit:	Miter limit. Default is 10.
width:	Line width.

- image: - A circle, icon, or a RegularShape. See the openlayer API (<http://openlayers.org/en/v3.14.2/apidoc/>) for more information.

type: "ol.style.Circle", "ol.style.Icon" , "ol.style.RegularShape",

parameters: same as the openlayer's "ol.style.Circle" , "ol.style.Icon", "ol.style.RegularShape" API

if the type is ol.style.Circle:

Table 26. Style is ol.style.Circle

Parameter	Description
fill:	Set fill style for vector features. (Object) Refer to openlayer API for more information.
stroke:	Set stroke style for vector features. (Object) Refer to the openlayer API for more information.
radius:	Radius of the circle.

if the type is ol.style.Icon:

Table 27. Style is ol.style.Icon

Parameter	Description
fill:	Set fill style for vector features. (Object) Refer to openlayer API for more information
stroke:	Set stroke style for vector features. (Object)

Table 27. Style is *ol.style.Icon* (continued)

Parameter	Description
anchor:	Anchor. Default value is [0.5, 0.5] (icon center).
anchorOrigin:	Origin of the anchor: bottom-left, bottom-right, top-left or top-right. Default is top-left.
offset:	Offset, which, together with the size and the offset origin, define the sub-rectangle to use from the original icon image. Default value is [0, 0].
offsetOrigin:	Origin of the offset: bottom-left, bottom-right, top-left or top-right. Default is top-left.
opacity:	Sets the opacity of the icon. Default is 1.
scale:	Scale
rotation:	Sets the rotation in radians (positive rotation clockwise). Default is 0.
src:	The URL of the image source.
size:	Icon size in pixel. Can be used together with offset to define the sub-rectangle to use from the origin (sprite) icon image.

If the type is *ol.style.IRegularShape*: Then the resulting shape is a regular polygon when radius is provided, or a star when radius1 and radius2 are provided.

Table 28. Style is *ol.style.IRegularShape*

Parameter	Description
fill:	Set fill style for vector features.(Object)
stroke:	Set stroke style for vector features. (Object)
points:	Number of points for stars and regular polygons. In the case of a polygon, the number of points is the number of sides.
radius:	The circumradius of a regular polygon.
radius1:	The inner radius of a star.
radius2:	The outer radius of a star.
angle:	The angle from the vertical of a shape in radians. A value of 0 will have one of the shape's point facing up. Default value is 0.
rotation:	The angle of rotation in radians (positive rotation clockwise). Default is 0.

- text: - Sets the text style for vector features.
if the type is "ol.style.Text"

Table 29. Style is *ol.style.Text*

Parameter	Description
font:	The font used as a CSS font value. .

Table 29. Style is ol.style.Text (continued)

Parameter	Description
offsetX:	The horizontal text offset in pixels. A positive value shifts the text to the right. Default is 0.
offsetY:	The vertical text offset in pixels. A positive value shifts the text down. Default is 0.
scale:	Scale
rotation:	The angle of rotation in radians (positive rotation clockwise). Default is 0.
text:	The text content.
textAlign:	The text alignment. The values are: 'left', 'right', 'center', 'end' and 'start'. The default is 'start'..
textBaseline:	Text base line. Possible values: 'bottom', 'top', 'middle', 'alphabetic', 'hanging', 'ideographic'. Default is 'alphabetic'.
fill:	Set fill style for vector features. (Object).
stroke:	The stroke style for vector features.

Example

```
{
  "id": "overheadlineLayer",
  "module": "ifef/widget/map/DataLayer",
  "container": "map",
  "properties": ["store", "creteria", "selected"],
  "parameters": {
    "index": 2,
    "map": "@{map}",
    "styles": [
      {
        "condition": "STATUS==0",
        "style": {
          "stroke": {
            "type": "ol.style.Stroke",
            "parameters": {
              "color": "#699037",
              "width": 2
            }
          }
        }
      },
      {
        "condition": "STATUS==1",
        "style": {
          "stroke": {
            "type": "ol.style.Stroke",
            "parameters": {
              "color": "#FDBA1A",
              "width": 2
            }
          }
        }
      },
      {
        "condition": "STATUS==2",
        "style": {
          "stroke": {
            "type": "ol.style.Stroke",

```

```

        "parameters": {
            "color": "#C32E14",
            "width": 2
        }
    }
}]]
},
{"id": "overheadlineModel",
 "module": "ifef/model/Model",
 "properties": ["store"],
 "parameters":
 {
     "storeModule": "ifef/model/Store",
     "target": "/ibm/ife/sample/dno/api/overheadline",
     "idProperty": "ASSET_ID"
 }
},
{"id": "bind1",
 "module": "ifef/behavior/Bind",
 "parameters": {
     "bindings": [
         {
             "source": "@{overheadlineModel}",
             "sourceProp": "store",
             "target": "@{overheadlineLayer}",
             "targetProp": "store"
         }
     ]
 }
},

```

Logical map widget:

The logical map shows the relationship between assets and between an asset and a type of measurement.

The user can query the logical map via a keyword of the instance. The logical map can be extended to show related items, and the user can analyze the impact between instances.

In the illustration, the node in the logical map represents the instance. The line represents the relationship.

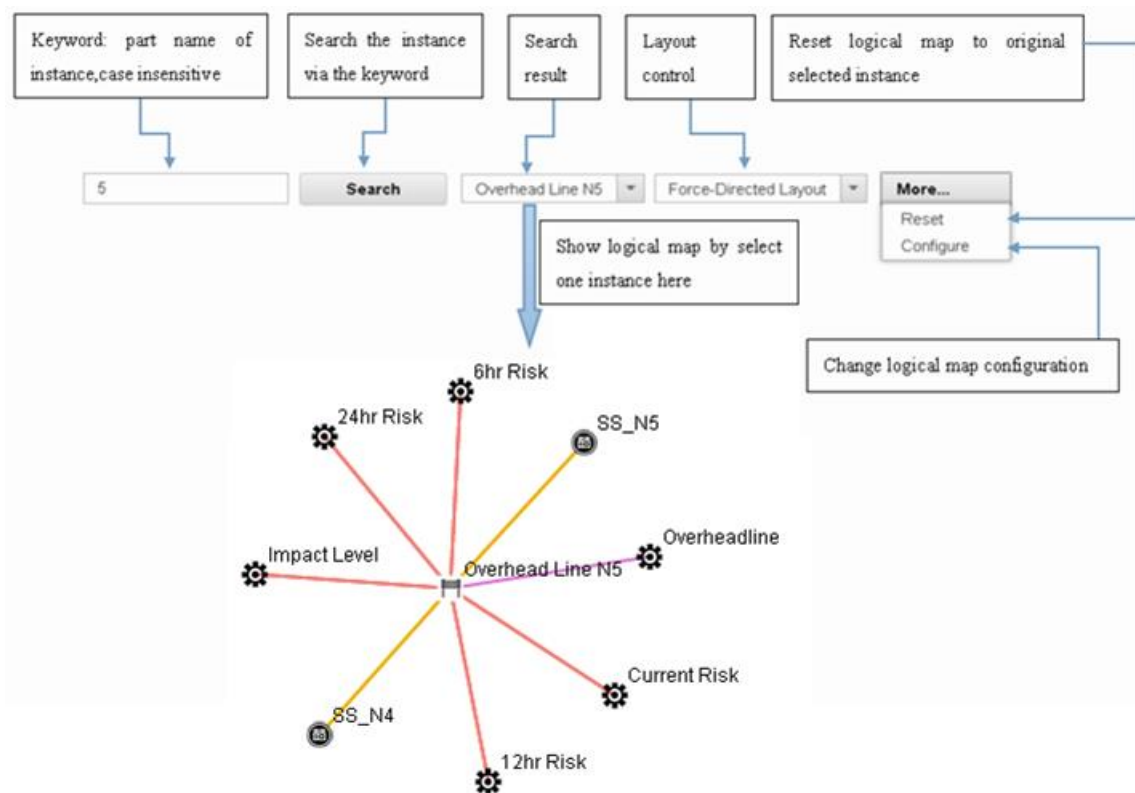


Figure 52. Logical map widget

The framework of a logical map is in three parts:

1. The logical map and its services. The data is queried from the model server via services and the model server interface.
2. The crawler and model server, The crawler generates the OWL file based on the configuration.
3. The user-defined part. This includes the data model and its service, and the configuration for the crawler to generate the OWL file.

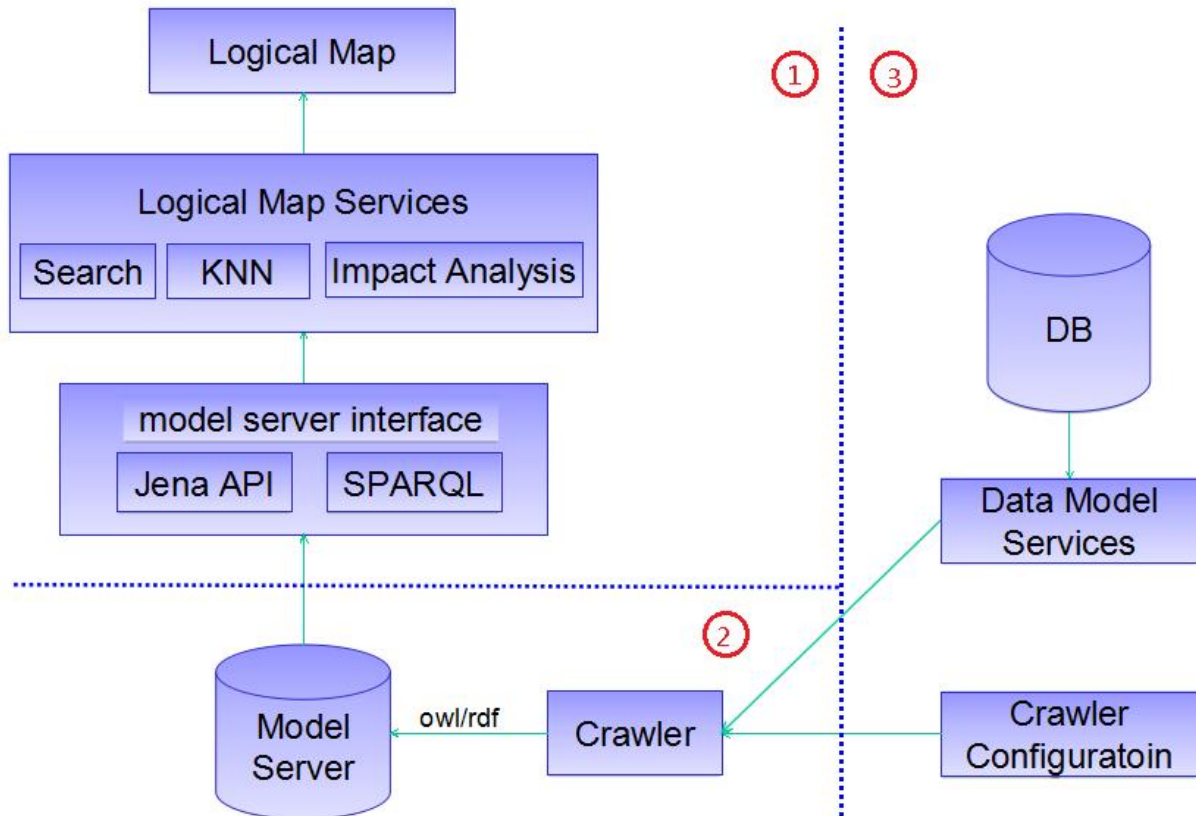


Figure 53. The framework of a logical map

Here is sample data that is generated by the crawler. In this example the OWL defines 2 kinds of instance types:

- substation and asset.
- Current_Risk a measurement type.

The OWL also defines two relationships:

- Asset.hasMeasurement references the measurement belonging to the asset.
- Measurement.associatedToAsset references the asset belonging to the measurement.

The two relationships are an inverse of each other, therefore in the rdf file, only one relationship needs to be defined. After execution of inference.sh from the Jena installation path the other relationship is automatically generated.

```
<rdf:Description rdf:about="http://ontology#substation">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#class"/>
  <rdfs:label xml:lang="en">Substation</rdfs:label>
  <rdfs:subClassOf rdf:resource="http://ontology#asset"/>
</rdf:Description>

<rdf:Description rdf:about="http://ontology#Asset.hasMeasurement">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:label xml:lang="en">Asset.hasMeasurement</rdfs:label>
  <owl:inverseOf rdf:resource="http://ontology#Measurement.associatedToAsset">
  <rdfs:subClassOf rdf:resource="http://ontology#Relationship"/>
</rdf:Description>

<rdf:Description rdf:about="http://ontology#Current_Risk">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
```

```

    <rdfs:label xml:lang="en">Current_Risk</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://ontology#Measurement"/>
    <rdfs:comment>Measurement</rdfs:comment>
  </rdf:Description>

  <rdf:Description rdf:about="http://ontology#Measurement.associationToAsset">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:label xml:lang="en">Measurement.associationToAsset</rdfs:label>
    <owl:inverseOf rdf:resource="http://ontology#Asset.hasMeasurement">
    <rdfs:subClassOf rdf:resource="http://ontology#Relationship"/>
  </rdf:Description>

```

The example rdf file defines that Current_Risk_944 is a measurement Measurement.associatedToAsset of substation_2596. After execution of the inference.sh then substation_2596 has the relationship Asset.hasMeasurement and references Current_Risk_944.

The rdf file defines two instances:

- substation_2596
- Current_Risk_944

These two instance have the relationship Measurement.associatedToAsset. The Current_Risk_944 is the measurement of the asset substation_2596.

Note: The rdf:resource should refer to an existing id defined in rdf:ID, e.g. rdf:resource of Current_Risk_944 refers to the rdf:ID of substation_2596.

```

<otl:substation rdf:ID="substation_2596">
  <otl:Object.name>Anytown_North</otl:Object.name>
  <rdfs:label xml:lang="en">Anytown_North</rdfs:label>
  <otl:Object.item.Url>ibm/ife/sample/dno/api/substation/2596</otl:Object.item.Url>
</otl:substation>

<otl:Current_Risk rdf:ID="Current_Risk_944">
  <otl:Object.name>Current Risk</otl:Object.name>
  <rdfs:label xml:lang="en">Current Risk</rdfs:label>
  <otl:Object.description>Current Risk</otl:Object.description>
  <otl:Object.item.Url>ibm/ife/sample/dno/api/measurement/944</otl:Object.item.Url>
  <otl:Measurement.associcatedToAsset rdf:resource="#substation_2596"/>
</otl:Current_Risk>

```

The example rdf file has the crawler configuration necessary to generate the OWL and RDF files as follows:

```

{
  "type": "Class",
  "id": "substation",
  "value": "Substation",
  "subClassOf": "Asset",
  "owl_template": " <rdf:Description rdf:about=\"@{namespace.fullurl}@{substation.id}\">\n
    <rdf:type rdf:resource=\"http://www.w3.org/2002/07/owl#@{substation.type}\"/>\n
    <rdfs:label xml:lang=\"en\">@{substation.value}</rdfs:label>\n
    <rdfs:subClassOf rdf:resource=\"@{namespace.fullurl}@{substation.subClassOf}\"/>\n
  </rdf:Description>\n",
  "itemsUrl": "/ibm/ife/sample/dno/api/substation",
  "rdf_template": "<@{namespace.shorturl}:@{substation.id} rdf:ID=\"@{substation.id}_@{itemsUrl.asset_id}\">\n
    <@{namespace.shorturl}:@{Object_name.value}>@{itemsUrl.name}</@{namespace.shorturl}:@{Object_name.value}>\n
    <rdfs:label xml:lang=\"en\">@{itemsUrl.name}</rdfs:label>\n
    <@{namespace.shorturl}:@{Object_itemUrl.value}>@{itemsUrl.url}/@{itemsUrl.asset_id}</@{namespace.shorturl}:@{Object_itemUrl.value}>\n
  </@{namespace.shorturl}:@{substation.id}>\n"
},

```

Figure 54. Crawler configuration ID:substation

```

{
  "type": "Class",
  "id": "Current_Risk",
  "value": "Current_Risk",
  "subClassOf": "Measurement",
  "owl_template": "<rdf:Description rdf:about=\"@{namespace.fullurl}@{Current_Risk.id}\">\n
  <rdf:type rdf:resource=\"http://www.w3.org/2002/07/owl#{@(Current_Risk.type)}\"/>\n
  <rdfs:label xml:lang=\"en\">{@(Current_Risk.value)}</rdfs:label>\n
  <rdfs:subClassOf rdf:resource=\"@{namespace.fullurl}@{Current_Risk.subClassOf}\"/>\n
  <rdfs:comment>Measurement</rdfs:comment>\n
  </rdf:Description>\n",
  "itemsUrl": "/ibm/ife/sample/dno/api/measurement",
  "rdf_template": "<@{namespace.shorturl}:@{Current_Risk.id} rdf:ID=\"@{Current_Risk.id}_@{itemsUrl.MEASUREMENT_ID}\">\n
  <@{namespace.shorturl}:@{Object_name.value}>@{itemsUrl.name}</@{namespace.shorturl}:@{Object_name.value}>\n
  <rdfs:label xml:lang=\"en\">{@(itemsUrl.name)}</rdfs:label>\n
  <@{namespace.shorturl}:@{Object_description.value}>@{itemsUrl.description}</@{namespace.shorturl}:@{Object_description.value}>\n
  <@{namespace.shorturl}:@{Object_itemUrl.value}>@{itemsUrl.url}/@{itemsUrl.MEASUREMENT_ID}</@{namespace.shorturl}:@{Object_itemUrl.value}>\n
  <@{namespace.shorturl}:Measurement.associatedToAsset rdf:resource=\"#@{itemsUrl.asset_type}_@{itemsUrl.asset_id}\" />\n
  </@{namespace.shorturl}:@{Current_Risk.id}>\n"
},

```

Figure 55. Crawler configuration ID:Current_Risk

```

{
  "type": "ObjectProperty",
  "id": "Measurement_associatedToAsset",
  "value": "Measurement.associatedToAsset",
  "subClassOf": "Relationship",
  "owl_template": "<rdf:Description rdf:about=\"@{namespace.fullurl}@{Measurement_associatedToAsset.value}\">\n
  <rdf:type rdf:resource=\"http://www.w3.org/2002/07/owl#{@(Measurement_associatedToAsset.type)}\"/>\n
  <rdfs:label xml:lang=\"en\">{@(Measurement_associatedToAsset.value)}</rdfs:label>\n
  <owl:inverseOf rdf:resource=\"@{namespace.fullurl}@{Asset_hasMeasurement.value}\"/>\n
  <rdfs:subClassOf rdf:resource=\"@{namespace.fullurl}@{Measurement_associatedToAsset.subClassOf}\"/>\n
  </rdf:Description>\n"
},

```

Figure 56. Crawler configuration ID:Measurement_associatedToAsset

Note: Explanatory notes:

- The red rectangle ID:"substation" should be the same value as itemsUrl.asset_type, otherwise the rdf:resource will refer to a none existing rdf:ID. The result in the logical map will not be correct.
- The values for id, value, subClassOf, owl_template, and rdf_template can all be edited.
- The value for itemsUrl is the service URL and is mandatory. It represents the result returned from the service.
- The label is used to search and is mandatory.
- The owl_template is the content of this type in the owl file.
- The rdf_template is the content of a instance of this type in the rdf file.
- The variable @{...} is changed to a real value.

The Service.properties file at the location: /opt/IBM/energy/crawler defines some of the properties used in the crawler.

Table 30. Type and parameter mapping

Property	Description
serviceBaseURL=http://ip:port	The service URL.
serviceUser=sysadmin	Service username.
servicePassword=passw0rd	Service password.
CLIENT_READ_TIMEOUT=0	Read_timeout.
CLIENT_CONNECT_TIMEOUT=0	Connect_timeout.
owlfile=d:/newfile.owl	New OWL file.

Table 30. Type and parameter mapping (continued)

Property	Description
rdffile=d:/newrdffile.rdf	New rdf file.

ifef.widget.LogicalMap Property

The property for the logical map is:

layoutData:

The data that is used to draw in the logical map.

ifef.widget.LogicalMap Parameters

The parameters for the logical map are:

keyFields:

Used to identify a record, json object, coming from a call of the model.

namespace:

The namespace of the data as an OWL/rdf file that is generated by the crawler. The namespace is defined in the configuration file of the crawler.

knnDepth:

The depth for the KNN service, K-NearestNeighbor.

impactAnalysisDepth:

The depth of the impact analysis service. It can be temporarily changed by the configuration dialog of the logical map.

instCfg:

The configuration for the instance. It can be temporarily changed by the configuration dialog of the logical map. The user can add or delete items to add or remove one kind of instance. The instType of newly added item here must be generated in OWL files and already imported to model server.

For example: Add one instance type:

```
{ "instType": "undergroundline", "canBeOnMap": true, "checked": false, "label": "Underground Line", "instStyle": "/ibm/ife/logicalmap/test/logicalmap/icon/undergroundline.png" }
```

The instance type is defined in the configuration file of crawler:

```
{
  "type": "Class",
  "id": "substation",
  "value": "Substation",
  "subClassOf": "Asset",
  "owl_template": " <rdf:Description rdf:about=\"@{namespace.fullurl}@{substation.id}\">\n
    <rdf:type rdf:resource=\"http://www.w3.org/2002/07/owl#@{substation.type}\"/>\n
    <rdfs:label xml:lang=\"en\">@{substation.value}</rdfs:label>\n
    <rdfs:subClassOf rdf:resource=\"@{namespace.fullurl}@{substation.subClassOf}\"/>\n
  </rdf:Description>\n",
  "itemsUrl": "/ibm/ife/sample/dno/api/substation",
  "rdf_template": "<@{namespace.shorturl}@{substation.id} rdf:ID=\"@{substation.id}_@{itemsUrl.asset_id}\">\n
    <@{namespace.shorturl}@{Object_name.value}>@{itemsUrl.name}</@{namespace.shorturl}@{Object_name.value}>\n
    <rdfs:label xml:lang=\"en\">@{itemsUrl.name}</rdfs:label>\n
    <@{namespace.shorturl}@{Object_itemUrl.value}>@{itemsUrl.url}/@{itemsUrl.asset_id}</@{namespace.shorturl}@{Object_itemUrl.value}>\n
  </@{namespace.shorturl}@{substation.id}>\n"
},
```

Figure 57. Configuration file of the crawler

- canBeOnMap - the instance can be displayed on map. If true, then there must be a "Highlight On Map" action for this instance.
- checked - Determines if this instance type is used to filter the search result.
- instStyle - The icon used for this instance type
- label - The text display in configuration dialog.

relationshipCfg:

The configuration for the relationship. It can be temporarily changed by the configuration dialog of logical map. The user can add or delete item here to add or remove one kind of relationship. The relationship must be generated in the OWL file and already be imported to the model server.

For example: Add one relationship.

```
{"relationship":"http://ontology#Asset.NewRelationship", "checked":true, "label":"New Relation
```

- relationship - the relationship defined in the OWL file.
- checked - determines if this relationship is used to filter the search result.
- relationshipStyle - the color for this relationship. If the value is not in range of ColorPalette, then black(#000000) is used.
- label - the text displayed in the configuration dialog.

An sample configuration

```
{
    "id": "logicalMap",
    "module": "lmap/widget/logicalmap/LogicalMap",
    "container": "logicalMapTab",
    "properties": ["layoutData"],
    "parameters": {
        "keyFields":
        ["ASSET_TYPE",
        "ASSET_ID"],
        "namespace": "http://ontology#",
        "knnDepth": 1,
        "impactAnalysisDepth": 6,
        "instCfg": [{"instType": "substation",
        "canBeOnMap": true, "checked": true,
        "label": "@{nls.CfgDlgInstSubstation}",
        "instStyle": "/ibm/ife/logicalmap/test/logicalmap/icon/substation.png"},
        {"instType": "overheadline", "canBeOnMap": true, "checked": true,
        "label": "@{nls.CfgDlgInstOverheadline}",
        "instStyle": "/ibm/ife/logicalmap/test/logicalmap/icon/overheadline.png"},
        {"instType": "wind_farm", "canBeOnMap": true, "checked": false,
        "label": "@{nls.CfgDlgInstWindFarm}",
        "instStyle": "/ibm/ife/logicalmap/test/logicalmap/icon/wind_farm.png"},
        {"instType": "Measurement", "canBeOnMap": false, "checked": false,
        "label": "@{nls.CfgDlgInstMeasurement}",
        "instStyle": "/ibm/ife/logicalmap/test/logicalmap/icon/measurement.png"},
        {"instType": "Default", "instStyle": "/ibm/ife/logicalmap/test/logicalmap/icon/
    }],
    "relationshipCfg": [{"relationship": "http://ontology#Asset.Connects", "checked": true, "
    "relationshipStyle": "#7fff00"}, {"relationship": "http://ontoc
    "checked": true, "label": "@{nls.CfgDlgRealConnected}", "relationshipStyle": "#ffa500"},
        {"relationship": "http://ontology#Asset.Contains",
        "checked": false, "label": "@{nls.CfgDlgRealContains}", "relationshipStyle": "#6495ed"},
        {"relationship": "http://ontology#Asset.Contained",
        "checked": false, "label": "@{nls.CfgDlgRealContained}", "relationshipStyle": "#7b68ee"},
        {"relationship": "http://ontology#Asset.hasMeasurement",
        "checked": true, "label": "@{nls.CfgDlgRealHasMeasurement}", "relationshipStyle": "#f08080"},
        {"relationship": "http://ontology#Measurement.associatedToAsset",
        "checked": true, "label": "@{nls.CfgDlgRealAssociatedToAsset}",
        "relationshipStyle": "#ff4500"},
        {"relationship": "http://ontology#Asset.hasPrimaryMeasurement",
        "checked": true, "label": "@{nls.CfgDlgRealHasPrimaryMeasurement}", "relationshipStyle":
```

```

        {"relationship":"http://ontology#Measurement.Is_Primary_Measurement_Of",
"checked":true, "label":"@{nls.CfgDlgRealIsPrimaryMeasurementOf}", "relationshipStyle":",
        {"relationship":"http://www.w3.org/2000/01/rdf-schema#subClassOf",
"checked":true, "label":"@{nls.CfgDlgRealSubClassOf}",
"relationshipStyle":"#ffe4c4"},
        {"relationship":"http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
"checked":true, "label":"@{nls.CfgDlgRealType}",
"relationshipStyle":"#da70d6"}
    ]
}
}

```

Related configuration for the logical map

There are three other configurations to support the logical map:

1. Logical map layouts - used to draw logical maps with different layouts.

- Property -
rawLayoutData: the data used to draw the logical map.

- Parameter -
The id of logical map.

Configuration sample:

```

{
  "id": "logicalMapLayouts",
  "module": "lmap/behavior/LogicalMapLayouts",
  "properties": ["rawLayoutData"],
  "parameters": { "logicalMap": "@{logicalMap}" }
}

```

2. Logical map previewcard helper - used as the helper class for more actions of logical map previewcard.

- Parameters:
 - logicalMap: the ID of logical map.
 - highlightOnMap: the common highlight on the map support.

Configuration sample:

```

{
  "id": "logicMapPreviewCardHelper",
  "module": "lmap/widget/logicalmap/LogicMapPreviewCardHelper",
  "parameters": {
    "logicalMap": "@{logicalMap}",
    "highlightOnmap": "@{highlightOnMap.highlightOnMap}"
  }
}

```

3. Logical map previewcard - Refer to “PreviewCard widget” on page 133 for information about the parameters.

The property “NAME” is required to display the relationship or instance information without an item URL, for example: Instance type:substation, in the previewcard. For example: {"label":"@{dno_nls.NAME}", "name":"NAME", "isKey": true}.

A sample configuration is given:

```

" id": "logicalMapPreviewCard",
  "module": "ifef/widget/previewcard/PreviewCard",
  "properties": ["data"],
  "parameters": {
    "titleProperty": "NAME",
    "position": "_position",
    "properties": [
      {"label":"@{dno_nls.Measurement_ID}", "name":"MEASUREMENT_ID", "isKey": true},
      {"label":"@{dno_nls.Asset_ID}", "name":"ASSET_ID", "isKey": true},
      {"label":"@{dno_nls.FARM_ID}", "name":"FARM_ID", "isKey": true},
      {"label":"@{dno_nls.Measurement_Type}", "name":"MEASUREMENT_TYPE", "isKey": true},
      {"label":"@{dno_nls.ID}", "name":"ID", "isKey": true}, {"label":"@{dno_nls."

```

```

        {"label": "@{dno_nls.Serial_Number}", "name": "SERIAL_NUMBER", "isKey": true},
        {"label": "@{dno_nls.UNIT}", "name": "UNIT", "isKey": true},
        {"label": "@{dno_nls.Description}", "name": "DESCRIPTION", "isKey": true},
        {"label": "@{dno_nls.Create_Date}", "name": "CREATE_DATE"},
        {"label": "@{dno_nls.Installation_Date}", "name": "INSTALLATION_DATE"},
        {"label": "@{dno_nls.STATUS}", "name": "STATUS"},
        "name": "STATUS",
        "isKey": true,
        "render": {
            "templates": [
                {
                    "condition": "STATUS==0",
                    "content": "<span
style='background-color:green'>${Acceptable}</span>",
                    "variables": {
                        "Acceptable": "@{dno_nls.Acceptable}"
                    }
                },
                {
                    "condition": "STATUS==1",
                    "content": "<span
style='background-color:orange'>${Caution}</span>",
                    "variables": {
                        "Caution": "@{dno_nls.Caution}"
                    }
                },
                {
                    "condition": "STATUS==2",
                    "content": "<span
style='background-color:red'>${Critical}</span>",
                    "variables": {
                        "Critical": "@{dno_nls.Critical}"
                    }
                }
            ]
        },
        {"label": "@{dno_nls.Location}", "name": "LOCATION"},
    ]"
    "moreActions": [
    ]
    }
}

```

List Widget:

The List widget is the wrapper for the idx gridx widget. It provides the capability for a developer to create and configure list and column customizations.

ifef.widget.List Properties

store:

The data store that acts as the bridge to the REST service.

selected:

When a property is selected from a list, it is populated with the corresponding JSON object.

criteria:

A JSON object that describes the filter criteria. This property will be populated only if the `keyFields` parameter is specified (see below) and is updated when the selected property is changed.

ifef.widget.List Parameters

list level:

- **pageSize:** The initial size of the page, the number of the items that will be displayed on the initial page. For example: "pageSize": 10 will show 10 items on the initial page.
- **paginationBar:** the pagination for the list. The quantity of items that show on each page. A value of 150 will display the next 150 items on that page and also on subsequent pages. The last value 0 displays all the items in the data store.
- **baseSort:** The sort sequence for the columns, ascend or descend. Specify descending as true or false to determine the sort order of the column.

Example:

```
"baseSort": [  
  {  
    "attribute": "ASSET_ID",  
    "descending": true  
  },  
  {  
    "attribute": "NAME",  
    "descending": true  
  }  
]
```

- **keyFields:** The fields used to generate the criteria and selectors based on the selected row in the list.

column level:

- **field:** The column field. The field that matches the field in the store that is queried from the database.
- **name:** The name for the column that will be displayed in the list. For example "name": "ASSET_STATUS"

Note:

The user can use the strings in the nls file to support globalization , for example: "name": "@{dno_nls.STATUS}".

- **sortable:** Determines if the column can be sorted or not, true or false. For example: "sortable": true, mean that the column can be sorted.
- **width:** the width of the column in pixels. For example: "width": "80px".
- **formatter:** the format of the cell data for time or date.

– type: specify the two data type that you want to format , 'number' or 'date'.

number
When number, you must also specify the parameter "places", this is the number of fixed significant digits without specifying the dot as a placement. For example:

```
"formatter":  
{  
  "type": "number",  
  "places":2  
}
```

date

When date, you must also specify the parameter "formatLength". The available values are: long, short, medium and full. For example:

```

"formatter":
{
  "type": "date",
  "formatLength": "short"
}

```

- **style:** the style for the cells. The value for this style can be a function that returns a style string, or simply be a style string for this column. For example: "style" : "background-color:#EE3D96;color:red" or "style" : "@{statusFormatter.style}".
- **decorator:** the function to decorate the raw data.

Configuration example

```

{
  "id": "measurementList",
  "module": "ifef/widget/list/List",
  "container": "measurementPane",
  "properties": ["store", "selected", "criteria", "selector"],
  "parameters": {
    "selectorFields" : ["MEASUREMENT_TYPE"],
    "keyFields" : ["MEASUREMENT_ID"],
    "pageSize": 10,
    "paginationBar": [10,50,100,0],
    "baseSort": [
      {
        "attribute": "STATUS",
        "descending": false
      },
      {
        "attribute": "NAME",
        "descending": true
      }
    ],
    "column": [
      {
        "name": "@{dno_nls.STATUS}",
        "field": "STATUS",
        "sortable" : true,
        "style" : "@{statusFormatter.style}",
        "decorator" : "@{statusFormatter.decorator}"
      },
      {
        "name": "@{dno_nls.Measurement_ID}",
        "field": "MEASUREMENT_ID",
        "sortable" : true
      },
      {
        "name": "@{dno_nls.NAME}",
        "field": "NAME",
        "sortable" : true
      },
      {
        "name": "@{dno_nls.UNIT}",
        "field": "UNIT",
        "sortable" : false
      }
    ]
  }
}

```

List Container Widget:

The user can place one list directly in one page. However, where there are multiple lists that can change based on the selection made by the user, it is better to put all the lists in one List Container.

In the List Container widget you can specify the container for the lists. This is configured in a widget containment JSON file.

ifef.widget.ListContainer Parameters

selectLabel:

The label used for the drop down list.

style:

The cascading style sheet (CSS) used to define the display style.

Configuration sample

```
{
  "id": "listContainer",
  "module": "ifef/widget/list/ListContainer",
  "container": "listTab",
  "parameters": {
    "selectLabel": "@{dno_nls.Select_Item}",
    "style": "width:
100%; height: 100%;"
  }
},
{
  "id": "overheadlineList",
  "module": "ifef/widget/list/List",
  "container": "listContainer",
  "properties": ["store"],
  "parameters": {
    "keyFields": ["ASSET_TYPE", "ASSET_ID"],
    "title": "@{overheadline_filter.label}",
    "pageSize": 100,
    "paginationBar": [100,150,200,1000,0],
    "baseSort": [
      {
        "attribute": "STATUS",
        "descending": true
      }
    ],
    "column": [
      {
        "name": "@{dno_nls.STATUS}",
        "field": "STATUS"
      },
      {
        "name": "@{dno_nls.NAME}",
        "field": "NAME",
        "sortable": true
      }
    ]
  }
}
}
```

Line Chart Widget:

The LineChart widget is used to show changes to a series of values over time.

The Line Chart can be enlarged or made smaller by zooming in or out, and the time period can be changed by dragging the chart to show different time periods. The line chart is used to show changes of one measurement reading over time for a specified asset.

ifef.widget.LineChart Parameters

title:

String, the title of the chart.

chartWidth:

Number, specifies the width of the chart.

chartHeight:

Number, specifies the height of the chart.

series:

Array, describes the number of lines and name for the specified series.

- **displayName:** String, the display name for the specified series.
- **property:** String, The field name for each of the series of the Y axis data. The field name must be a number.

X_property:

String, The field name for each time period used for the X axis. The field name must be a timestamp or period.

X_title:

The display name for the X axis.

Y_title:

The display name for the Y axis.

The properties for the Line Chart widget

store:

The data for line chart should be bound to the property store of the LineChart instance. This property changes when the user selects different measurement criteria.

Configuration sample

The configuration sample shows two series of data for the displayName VALUE1 and VALUE2.

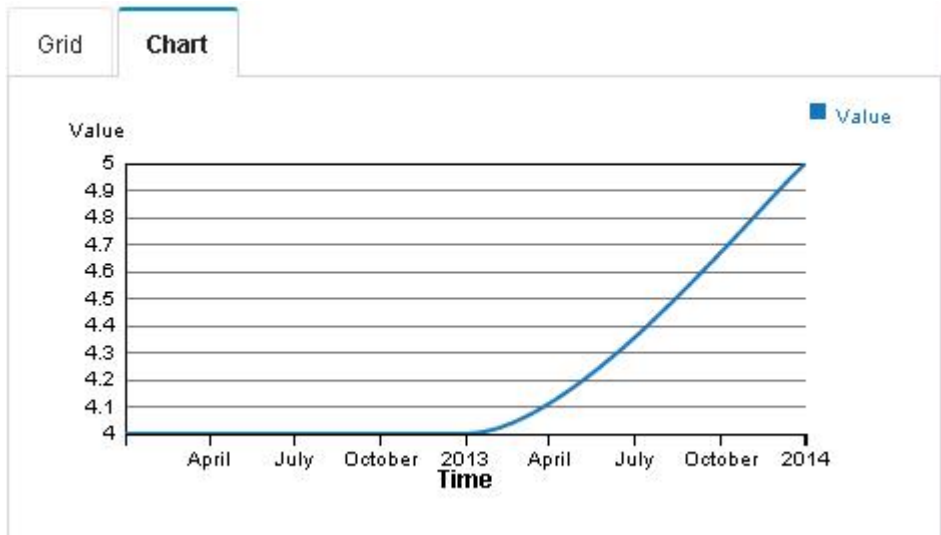


Figure 58. Example line chart

```

{
  "id": "readingLineChart",
  "module": "ifef/widget/chart/LineChart",
  "container": "readingLineChartTab",
  "properties": ["store"],
  "parameters": {
    "title": "@{dno_nls.Reading_Line_Chart}",
    "chartWidth": 450,
    "chartHeight": 200,
    "series": [{
      "displayName": "@{dno_nls.VALUE1}",
      "property": "VALUE1"
    }, {
      "displayName": "@{dno_nls.VALUE2}",
      "property": "VALUE2"
    }
  ],
  "X_property": "TIME",
  "X_title": "@{dno_nls.TIME}",
  "Y_title": "@{dno_nls.VALUE}"
}

```

The data for the Line Chart should be provided as a JSON Array that must have a field that contains the timestamp value. The other fields which contain number value can be shown as a series of lines.

```

[
  {
    "id": 1,
    "TIME": "2013-01-01 08:58:00",
    "VALUE": 4
  },
  {
    "id": 2,
    "TIME": "2014-01-01 08:59:00",
    "VALUE": 4
  },
  {
    "id": 3,
    "TIME": "2015-01-01 09:00:00",
    "VALUE": 5
  }
]

```


Bar Chart widget:

The BarChart widget is used to show a group of related values that can be grouped as a hierarchical relationship.

As a time-series value, a bar chart widget can be grouped to show year, Month, Day, Hour, or Minute.

As a population series a bar chart can be grouped to show population by Country, State, City, Town, or Region.

You can click on the bar chart to drill up and drill down the hierarchical levels.

The example bar chart is used to show measurements readings for maximum, minimum and average values for a specified asset. The bar chart is grouped by Year, Month, Day, Hour and Minute.

ifef.widget.LineChart Parameters

title:

String, the title of the chart.

chartWidth:

Number, specifies the width of the chart.

chartHeight:

Number, specifies the height of the chart.

legends:

Array, specifies the number of bars for each of the groups.

- **displayName:** String, the display name for the specified series of bars.
- **property:** String, The field name for each series of bars of Y axis data. The field name must be a number.

X_property:

String, The field name for each time period used for the X axis. The field name must be a number.

X_title:

The display name for the X axis.

defaultLevelIndex:

Number, specifies the level of the hierarchy that shows when the bar chart opens.

levels:

Array, describes the number of levels of the bar chart can be drilled down or up, and also the number of hierarchy levels of the data in the chart.

- **index:** String, the index number of the current level.
- **X_title:** String, The display name of the X axis.
- **modelSelector:** Widget instance, specifies which model selector instance is used for the current level. Model Selector is a type of REST store map that provides the specified store for the widget by a key. The following figure is an example of a model selector instance.

```

"id": "readingSelector",
"module": "ifef/model/ModelSelector",
"properties": ["selector", "store"],
"parameters": {
  "models": [
    {"key": "Load_Index", "model": "@{loadIndexModel}"},
    {"key": "Health_Index", "model": "@{healthIndexModel}"},
    {"key": "Impact_Level", "model": "@{impactLevelModel}"},
  ]
}

```

Figure 59. Model Selector instance for the current level of hierarchy

- params: Array, the fields that describe the hierarchy relationship of each level.
 - displayName: String, the display name for this field that shows as the drill up title.
 - property: String, specifies the field name of the data that is used to drill down or up and filter the data for Y axis.
 - value: Number, shows the default value for this field when the current level is the default level.

ifef.widget.BarChart Properties

store:

The data for the bar chart that should bound to the property “store” of the bar instance. This property changes when the user selects different measurement criteria.

Configuration sample

```

{
  "id": "readingBarChart",
  "module": "ifef/widget/chart/BarChart",
  "container": "statisticsBarChartTab",
  "properties": ["store", "chartListColumns", "chartListData"],
  "parameters": {
    {
      "title": "@{dno_nls.Reading_Bar_Chart}",
      "chartWidth":
        450,          "chartHeight":
        200,          "legends":
        [{
          "displayName": "@{nls.Chart_Label_MAX}",
          "property": "MAX"
        }],
        {
          "displayName": "@{nls.Chart_Label_MIN}",
          "property": "MIN"
        }],
        {
          "displayName": "@{nls.Chart_Label_AVG}",
          "property": "AVG"
        }
      ]],
      "X_property": "TIME",
      "Y_title": "@{dno_nls.VALUE}",
      "defaultLevelIndex": 0,
      "levels": [{
        "index": "0",
        "X_title": "@{nls.Chart_Label_YEAR}",
        "modelSelector": "@{readingByYearSelector}",
      }
    ]
  }
}

```

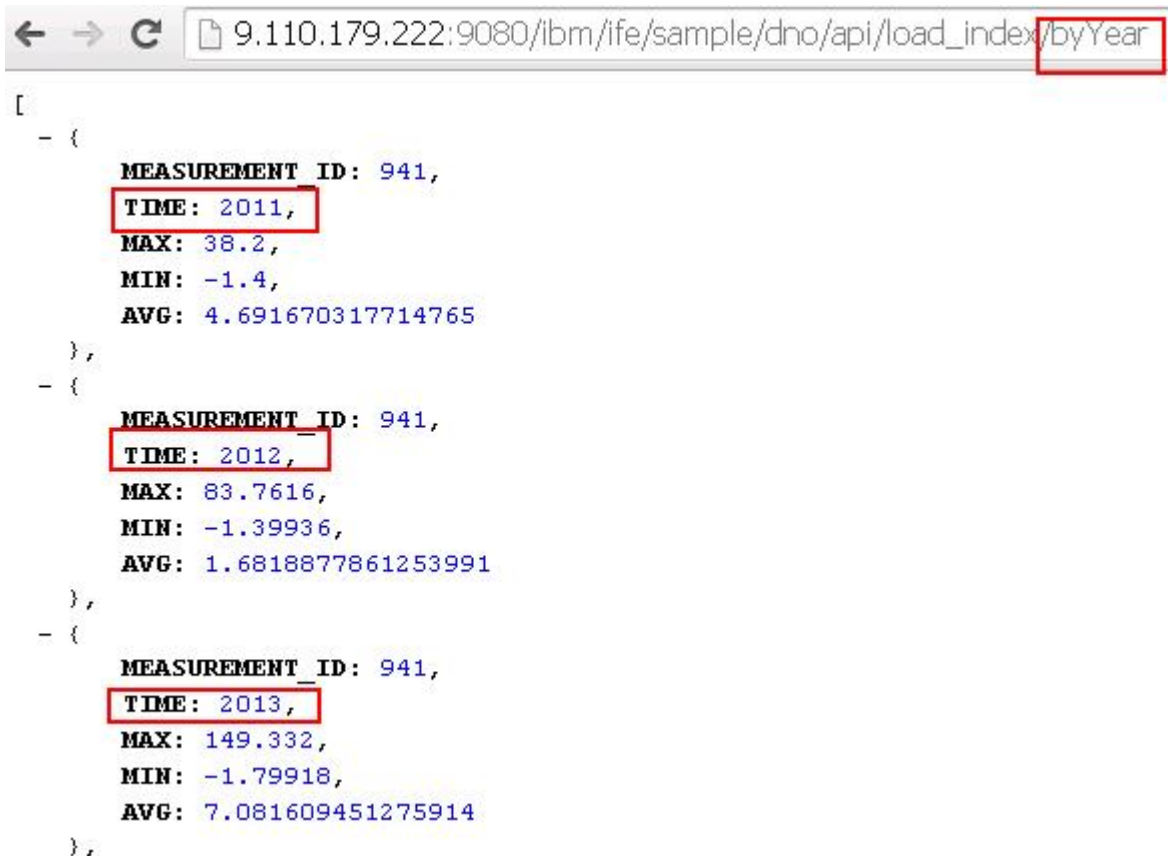
```

        "params": []
    },{
        "index": "1",
        "X_title": "@{nls.Chart_Label_MONTH}",
        "modelSelector": "@{readingByMonthSelector}",
        "params": [{
            "displayName": "@{nls.Chart_Label_YEAR}",
            "property": "YEAR",
            "value": "2013"
        }]
    },{
        "index": "2",
        "X_title": "@{nls.Chart_Label_DAY}",
        "modelSelector": "@{readingByDaySelector}",
        "params": [{
            "displayName": "@{nls.Chart_Label_YEAR}",
            "property": "YEAR",
            "value": "2013"
        }],{
            "displayName": "@{nls.Chart_Label_MONTH}",
            "property": "MONTH",
            "value": "1"
        }
    ]
    },{
        "index": "3",
        "X_title": "@{nls.Chart_Label_HOUR}",
        "modelSelector": "@{readingByHourSelector}",
        "params": [{
            "displayName": "@{nls.Chart_Label_YEAR}",
            "property": "YEAR",
            "value": "2013"
        }],{
            "displayName": "@{nls.Chart_Label_MONTH}",
            "property": "MONTH",
            "value": "1"
        },{
            "displayName": "@{nls.Chart_Label_DAY}",
            "property": "DAY",
            "value": "1"
        }
    ]
    },{
        "index": "4",
        "X_title": "@{nls.Chart_Label_MINUTE}",
        "modelSelector": "@{readingByMinuteSelector}",
        "params": [{
            "displayName": "@{nls.Chart_Label_YEAR}",
            "property": "YEAR",
            "value": "2013"
        }],{
            "displayName": "@{nls.Chart_Label_MONTH}",
            "property": "MONTH",
            "value": "1"
        },{
            "displayName": "@{nls.Chart_Label_DAY}",
            "property": "DAY",
            "value": "1"
        },{
            "displayName": "@{nls.Chart_Label_HOUR}",
            "property": "HOUR",
            "value": "1"
        }
    ]
    ]
}
}
}

```

Data sample

The data samples show a time-series value for year as the top level and month and day as subsequent levels.



```
[
- {
  MEASUREMENT_ID: 941,
  TIME: 2011,
  MAX: 38.2,
  MIN: -1.4,
  AVG: 4.691670317714765
},
- {
  MEASUREMENT_ID: 941,
  TIME: 2012,
  MAX: 83.7616,
  MIN: -1.39936,
  AVG: 1.6818877861253991
},
- {
  MEASUREMENT_ID: 941,
  TIME: 2013,
  MAX: 149.332,
  MIN: -1.79918,
  AVG: 7.081609451275914
},
]
```

Figure 60. Hierarchy level: By Year

```
- → 9.110.179.222:9080/ibm/ife/sample/dho/api/load_index/byMonth?
- {
  MEASUREMENT ID: 941,
  YEAR: 2013,
  TIME: 1,
  MAX: 7.79643,
  MIN: -1.79918,
  AVG: 1.617091073681189
},
- {
  MEASUREMENT ID: 941,
  YEAR: 2014,
  TIME: 1,
  MAX: 3.19853,
  MIN: -1.79918,
  AVG: -0.2287927434120238
},
- {
  MEASUREMENT ID: 941,
  YEAR: 2011,
  TIME: 2,
  MAX: 3.2,
  MIN: -1.4,
  AVG: 0.34890291865036355
},
- {
  MEASUREMENT ID: 941,
  YEAR: 2012,
  TIME: 2,
  MAX: 4.7978,
  MIN: -1.19945,
  AVG: 1.8333586516282894
}
```

Figure 61. Hierarchy level: By Month

9.110.179.222:9080/ibm/ife/sample/dno/api/load_index/byDay

```
- {
  MEASUREMENT_ID: 941,
  YEAR: 2011,
  MONTH: 1,
  TIME: 1,
  MAX: 3.2,
  MIN: -0.8,
  AVG: 2.39387308533915
},
- {
  MEASUREMENT_ID: 941,
  YEAR: 2012,
  MONTH: 2,
  TIME: 1,
  MAX: 4.19808,
  MIN: 2.19899,
  AVG: 2.865803075196376
},
- {
  MEASUREMENT_ID: 941,
  YEAR: 2013,
  MONTH: 2,
  TIME: 1,
  MAX: 0.199908,
  MIN: -1.39936,
  AVG: -0.8856438865710623
},
- {
  MEASUREMENT_ID: 941,
  YEAR: 2014,
  MONTH: 2,
  TIME: 1.
```

Figure 62. Hierarchy level: By Day

Header Button Widget:

The Header Button widget provides the ability to show a pop-up dialog box that contains other widgets.

The Header Button is placed on the header bar.

ifef.widget.HeaderButton Parameters

label:

String, the label of the button.

icon:

The customized icon for the button to be shown for the label.

content:

The widget instance, specifies the widget that show in the pop-up dialog box when opened.

Properties**alertcount:**

The alertcount is bound to the button, and shows as a number to the right of the label.

Configuration sample

In the configuration example, the icon can be customized, and the content changed to show the widgets that need to be included.



Figure 63. The header button showing the notifications button.

```

{
  "id": "notificationButton",
  "module": "ifef/widget/header/HeaderButton",
  "container": "header",
  "properties": ["alertcount"],
  "parameters": {
    "label": "Notification",
    "style": "float: right;",
    "buttonType": "info",
    "icon":
      "http://findicons.com/icon/download/175313/notification_warning/16/png",
    "displayMode": "iconAndLabel",
    "content": "@{notificationContent}"
  }
}

```

Ready-to-use application template

An application template is a set of preconfigured configuration widgets that provide the basis of an application for the user to build upon. The template gives the user a start in building the application.

The REST service framework

This section introduces the mechanism and tasks to help the user to develop customized services. The REST service framework programming model provides an extension mechanism that lets the user to develop their own service quickly. These customized services can be used by the custom user interface (UI extension) for existing data.

The extension mechanism is a declarative programming model that lets the user to quickly create custom services. The services are shown in REST style and support general large-scale data handling. For example: pagination, filter by expression, and sorting.

The Sample Application

The aim of the sample application is to help teach new developers how to incrementally develop the user interface based on UI framework provided by IFE.

There are two parts to the sample application:

- filter panel.
- corresponding map view.

The filter panel is used to control the data that is displayed in the map view. By enabling or disabling the filters, the map view changes accordingly.

Creating a UI Web project

About this task

You create a Dynamic Web Project called `ife_demo_web`.

Procedure

1. Start Eclipse.
2. Right-click the project explorer and select **Dynamic Web Project**.
3. In the **Project Name** field, type `ife_demo_web`.
4. In the **Target runtime** field, select **WebSphere Application Server V8.5 Liberty Profile**.
5. In the **Dynamic web module version** field, select **3.0**.
6. In the **Configuration** field, select **Default configuration for WebSphere Application Server V8.5 Liberty Profile**.
7. Click **Next**.
8. In the **Context root** field, type `ife_demo_web`.
9. In the **Content directory** field, type `WebContent` and enable the option: **Generate web.xml deployment descriptor**.
10. Click **Finish**.

Creating the main configuration and bootstrap files

You need to create the bootstrap file and place it into the `WebContent` directory of the project and create the configuration files to include different parts of the configuration items.

About this task

Different from traditional UI development, the UI Framework can parse JSON format configuration files dynamically at the runtime. The JSON format configuration contains the widget creation, containment, and the dependency relationship. The bootstrap file acts as the startup for the runtime.

Procedure

1. Create the bootstrap file `index.html` and save it to the `WebContent` directory for the project.

2. Create the configuration files to include the different configuration items, and save each one to the WebContent directory of the project.
 - config.json the entry configuration file.
 - layout.json the page layout configuration.
 - map.json the map view configuration.
 - model.json the model configuration.
 - bind.json the widget interaction configuration.

The example shows the structure of the directory.

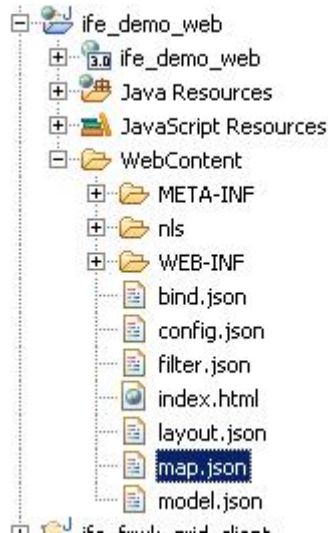


Figure 64. The structure of the directory

3. Create the UIMessage.js message file in the nls folder for the supported localization languages. The example message file contains the localized language settings for French, Chinese, traditional Chinese, Japanese, and Brazilian Portuguese.

```
define({
  root : {
    Assets : "Assets",
    STATUS : "Status",
    NAME : "Name",
    UNIT : "Unit",
    ...
  },
  "fr" : true,
  "zh" : true,
  "zh-tw" : true,
  "ja" : true,
  "pt-br" : true
});
```

For each enabled locale, there is corresponding subdirectory. In this example, the message files under fr, zh, zh-tw, ja, and pt-br or nls is created. The result is:



Figure 65. The nls directory structure

Deploying the web application to the Websphere Liberty server

About this task

You need to deploy the `ife_demo_web` project to the Websphere Liberty server.

Procedure

1. Open the Server view, and right-click **Websphere Application Server V8.5 Liberty** from the local host and select **Add and Remove**.
2. Select `ife_demo_web` from the **Available** field and click **Add**.
3. Click **Finish**.

Launching the page

After you have deployed the sample application, you can launch the web page.

Procedure

1. Open your browser.
2. Type the URL as follows: `https://<your ip>:<port>/ife_demo_web/index.html`
Where `<your ip>` and `<port>` are adjusted to your machine settings.

Chapter 8. Predictive models

Use predictive models to generate the information you need to make informed operational, maintenance, repair, or component replacement decisions.

This section describes the steps that are needed to build predictive models in the predictive maintenance area by using IBM Predictive Maintenance and Quality (PMQ). It also covers some sample use cases in the manufacturing field. Later, it highlights the steps involved, starting from the business/data understanding to deploying the predictive models built for a given use case.

The following models form the basis of the predictive models in IBM Predictive Maintenance and Quality:

- The Maintenance predictive model
- The Sensor Health predictive model
- The Top Failure Reason predictive model
- The Integrated Health predictive model

Sample predictive models are provided. For more information, see “IBM SPSS artifacts” on page 288.

The training and scoring process

The steps for training and scoring the predictive models are as follows:

1. The modeling node estimates the model by studying records for which the outcome is known and creates a model nugget. This is referred to as training the model.
2. The model nugget can be added to any stream with the expected fields to score records. By scoring the records for which you already know the outcome (such as existing customers), you can evaluate how well it performs.
3. After you are satisfied that the model performs acceptably well, you can score new data (such as health score of an asset or life time of an asset) to predict how they will perform.

Optimized recommended actions

When an asset or a process is scored and identified as having a high probability of failure, recommendations can be generated.

Define recommended actions by using rules in IBM Analytical Decision Management. Use IBM Analytical Decision Management to understand the drivers that are used to define the rules, and to determine what happens based on the scores received. For example, if a score breaches a threshold, what is the resulting action? You can automate alerts for recommended actions by integrating with other systems or by defining a routing rule to send emails. Depending on the manufacturing execution systems (MES) that you use, the recommendation may be acted on automatically. You can also predict the success rate of the corrective action that is based on previous actions.

When IBM Predictive Maintenance and Quality generates recommendations, for example, to inspect an asset, you can configure the system so that the

recommendation results in a work order that is created by IBM Maximo. The work order is populated with the information needed to complete the task, for example, a device identifier and a location.

Prioritize application template

Use the prioritize application template when you have a good understanding of the predictive analytics scores and the interaction between the predictive scores. You can use the template `OptimizedAssetMaintenance.xml` to prioritize your business objective that is based on, for example, profit maximization, or downtime minimization.

The Maintenance predictive model

The Maintenance predictive model helps you optimize your Preventive Maintenance System.

In the past, a scheduler would optimize a plant's Preventive Maintenance System (PMS) by carefully altering the days that were allotted for maintenance in the OEM's default schedule. The IBM Predictive Maintenance and Quality Maintenance predictive model helps you to optimize your maintenance schedule using predictive analysis.

Often, in a new setup of PMQ sensors in the plant, even if sensor data has not gained optimal maturity for effective predictions, there may be enough data in the plant's maintenance system (Maximo/ SAP-PM etc.) to initiate a Predictive Maintenance regime. IBM PMQ's Maintenance Analytics can work on such maintenance work orders alone and does not depend on any sensor data. Therefore, the Maintenance model may help to expedite the ROI of any Predictive Analytics system before any useful sensor data is obtained.

For some resources or instances, sensor analytics alone might not provide the most accurate predictions. In this case, you can combine the results of both Maintenance Analytics and Sensor Analytics (via Integration Analytics module) to produce more optimal end results.

Data understanding

The performance indicator table `RESOURCE_KPI` contains aggregated values for each day. You can use it to prepare for model training and scoring.

The following figure shows the counts of various profiles in the dataset for a given resource and their percentages of the dataset.

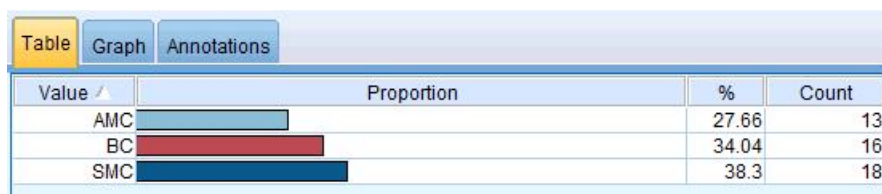


Figure 66. Percentage and counts of each profile

Additionally, the `MASTER_PROFILE_VARIABLE` and the `MASTER_MEASUREMENT_TYPE` tables help to define the appropriate codes, names, and other generic or static data.

The following figure shows a data audit node.

Field	Sample Graph	Measurement	Min	Max	Mean	Std. Dev	Skewness	Unique	Valid
KPI_DATE		Continuous	2010-01-01	2014-10-28	--	--	--	--	3287
ACTUAL_VALUE		Continuous	-82.650	70423.000	197.788	1631.452	34.824	--	2329
MEASURE_COUNT		Continuous	1	109	8.436	21.000	3.335	--	3287
PROFILE_VARIABL...		Continuous	1002	1106	1042.998	18.013	1.104	--	3287
RESOURCE_ID		Continuous	1146	1766	1174.556	122.805	4.486	--	3287
EVENT_CODE_ID		Continuous	1	1822	19.439	178.989	9.831	--	3287
LOCATION_ID		Continuous	4	1301	75.814	296.669	3.890	--	3287
PROCESS_ID		Continuous	7	73	10.654	15.097	3.890	--	3287
PRODUCTION_BA...		Continuous	11	434	34.421	96.755	3.890	--	3287
TENANT_ID		Continuous	1	1	1	0	--	--	3287

Figure 67. Data audit node

The data audit node provides summary statistics, histograms, and distribution graphs that can help you to better understand the data. The report also displays the storage icon (data type) before the field name.

Pre-modeling the data

All pre-modeling required by Maintenance Analytics is done during the MAINTENANCE.str modeling stream.

For information about both the modeling and pre-modeling data preparation, see “Modeling the data.”

Modeling the data

Modeling for the Maintenance model occurs during the MAINTENANCE.str stream.

See the following table for information about MAINTENANCE.str.

Table 31. The MAINTENANCE.str stream

Stream name	Purpose	Input	Target	Output
MAINTENANCE.str	Predicts forecasted days to maintenance interval of equipment based on Maximo work orders, and then converts these predictions into continuous health scores.	The Maximo (or other Plant Maintenance Systems') work orders converted into profiles for the actual, planned, and scheduled maintenance dates for Breakdown and Planned Maintenance.	<ol style="list-style-type: none"> 1. Custom Target as obtained using pre-data preparation within the stream itself. 2. IsFail 	<ol style="list-style-type: none"> 1. Forecasted days until the next maintenance for each resource and each historic and current day 2. Health score of the equipment for each day

There are some limitations that affect the Maintenance model:

- Limitations exist in the Breakdown + Planned maintenance work orders that are extracted from Maximo. As a result, these work orders are not optimal for forecasting directly. Breakdown + Planned maintenance work orders represent intermittent events, for which the default SPSS Time Series Modeling node cannot be used directly.
- Both types of the maintenance series contain censored data at either limit. For example, for the Breakdown series, we cannot identify from the given work orders what the optimal maintenance day would be to prevent a breakdown or irreversible wear. Similarly, for the Planned Maintenance work orders, we cannot identify the day on which a breakdown or irreversible wear would occur if we choose not to perform machine maintenance on the day identified by the Breakdown work orders.
- The series that we want to predict, that is, an ideal maintenance period, either does not exist or is divided into two series of planned and unplanned maintenance. Direct application of time series models, even with transfer function or multi-variate ARIMA models, might not help to solve the problem.

To overcome these limitations, IBM PMQ uses a custom application of Croston's forecasting methods for intermittent demand (patent pending). Using this method, the two work orders' dates series are converted to the difference of days and then combined into a single series (using censoring adjustments). This single series can be subsequently forecasted using available time series nodes in SPSS. In the current application, a simple method of global user-defined multiplicative factors is used. However, other more sophisticated, optimal, and customized methods can also be used.

The resulting value of the number of days until the next forecast can then be used to predict the machine's failure. Health scores can then be obtained using the raw propensity scores or the adj raw propensity/ or the confidence of the obtained predictions. These Health scores can be used directly or with standardization at each resource level. The present implementation uses standardization to get a uniform scale/ level of health scores for each resource.

Post modeling data manipulation

Post modeling for the Maintenance model occurs during the MAINTENANCE_DAILY.str and MAINTENANCE_EVENTS.str streams.

See the following table for more information.

Table 32. The MAINTENANCE_DAILY.str and MAINTENANCE_EVENTS.str streams

Stream name	Purpose	Input	Output
MAINTENANCE_DAILY.str	This is a post modeling data preparation stream for the purpose of preparing data for BI plots. This stream converts the predictions from the MAINTENANCE_TRENDS table into a format required by the Maintenance Overview Dashboard. The results are entered into the Maintenance Daily table in the DB.	The input data source contains all the records present in the Maintenance Trends table in the DB across all the days.	Only the current day's data with some transformations into the Maintenance Daily Table

Table 32. The MAINTENANCE_DAILY.str and MAINTENANCE_EVENTS.str streams (continued)

Stream name	Purpose	Input	Output
MAINTENANCE_EVENTS.str	This is a post modeling data preparation stream for the purpose of preparing data for BI plots. This stream converts the data from the MAINTENANCE DAILY table into a format required by the by IIB flows. The results are used to populate the IBM PMQ Events in the Event Observation table in the DB.	The input data source contains all the records present in the Maintenance Daily table in the DB.	A csv file (uploaded on the IIB integration in folder on the Analytics server) with the Maintenance Daily data in a format that can be used by IIB flows to populate the Events Table.

To improve the performance at the BI end and ensure a fast refresh and optimal user experience, all static computations and data manipulations (calculations and data manipulations not affected by user selection of prompts/ filters on the dashboards) were transferred to the SPSS batch jobs. These batch jobs can be run at an off-peak hour.

The later part of Maintenance.str and Maintenance_daily.str run the batch jobs and prepare the Maintenance Trends and Maintenance Daily tables.

The data from maintenance daily can be transferred back as events in a IBM PMQ acceptable event format. External applications can then access the events. Dashboards can also consume structure events efficiently, as the Overview dashboard does currently. The Maintenance_Events.str stream helps achieve this objective.

Evaluation of the model

This example illustrates how the Maintenance predictive model can be used effectively.

The following figure shows a time series plot with both predicted values and actual values. In this case, the predictions were accurate.

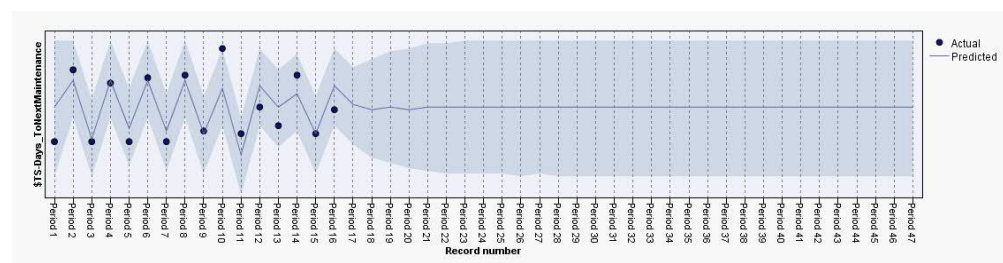


Figure 68. Time series plot

The Analysis node in the output tab helps with the evaluation of a particular model output. In this example, the predicted IsFAIL is compared on with the existing or actual values and arrived at a best-fitting training model. See the following table.

Table 33. Comparing \$L-IsFAIL with IsFAIL

Category	Value
Minimum Error	0.0
Maximum Error	1.0
Mean Error	0.032
Mean Absolute Error	0.032
Standard Deviation	0.177
Linear Correlation	
Occurrences	495

Deployment of the model

The Maintenance predictive model uses parameters from SPSS.

The model is developed using parameters which should also be used during deployment. Some parameters are configured in the downstream applications. If the parameter values are passed when the stream is executed, these values are used. Otherwise, default values are used.

The following figure shows parameters used for deployment.

Name	Long name	Storage	Value
RESOURCE_ID		Integer	1147
PROFILE_PLAN_AMC	PROFILE_VARIABLE_CD_PlannedMaintenance_ActualStart	String	AMC
PROFILE_PLAN_SMC	PROFILE_VARIABLE_CD_PlannedMaintenance_ScheduledStart	String	SMC
PROFILE_BREAKDOWN_BC	PROFILE_VARIABLE_CD_BreakdownMaintenance_Reported	String	BC
R_CENSURING	RightCensuring(Value>1)_PlannedMaintenanceLifeEnhancement	Real	1.2
L_CENSURING	LeftCensuring(Value<1)_BreakdownMaintenanceLifeReduction	Real	0.9
MAX_FUTURE_DAYS	Maximum_Future_Days_For_Which_Prediction_Is_Required	Integer	31

Figure 69. Parameters used for deployment

You can find all of these parameters using SPSS. However, only the RESOURCE_ID is exposed from the IIB end out of the box. This is because the stream has multiple execution branches that use scripts to sequence the parameters. You can see the scripts being referenced in the Execution tab.

Recommendations from ADM

The Maintenance predictive model provides scores and data that allow you to adjust maintenance dates optimally.

The deployed model, once invoked, helps to produce probability and propensity scores. However, probability and propensity scores may not be very useful to an end business user. Therefore, the results are consumed by IBM SPSS Decision Management, which then provides a more useful, text-based result.

The following figure shows the probability and propensity scores.

<input type="checkbox"/> Prepone_Maintenance_Dev_LT_-100	2005
DEVIATION_PERCENT < -100	
<input type="checkbox"/> Maintenance_as_planned_bet_0_10	3001
DEVIATION_PERCENT BETWEEN 0.0 and 10.0	
<input type="checkbox"/> Maintenance_as_planned_bet_-10_0	3002
DEVIATION_PERCENT BETWEEN -10.0 and 0.0	
<input type="checkbox"/> No Forecast Available	4001
FORECASTED_DAYS IS NULL	

Figure 70. Probability and propensity scores

Based on the scores and the data received from the modeler stream, we can determine whether specific maintenance tasks should be rescheduled.

The Sensor Health predictive model

The Sensor Health predictive model analyzes an asset's sensor readings to help determine the likelihood that the asset will fail. If the likelihood of failure is high, you can schedule an urgent inspection of the machine.

The Sensor Health model continuously monitors the health of a machine or an asset and predicts potential machine faults in real time. The model uses historical sensor data profile values stored in the KPI tables and keeps a running status to determine the current health of an asset. The Sensor Health model can also be used to predict the future health of an asset.

Tip: If there are too many failures (for example, more than 30% of the days, or multiple times in a day), then instead of using the KPI tables for training, a user could use raw events from the event table for training with appropriate filtering or treatment of noise, if any.

Data understanding

The Sensor Health predictive model uses the RESOURCE_KPI and MASTER_PROFILE_VARIABLE tables.

The performance indicator table RESOURCE_KPI is used to hold aggregated values for each day. The table can be further used to prepare for model training and scoring. The MASTER_PROFILE_VARIABLE is used to help identify the specific profiles and select only the profiles that require further analysis.

The following diagram shows an example of the source data stream for the Sensor Health predictive model.

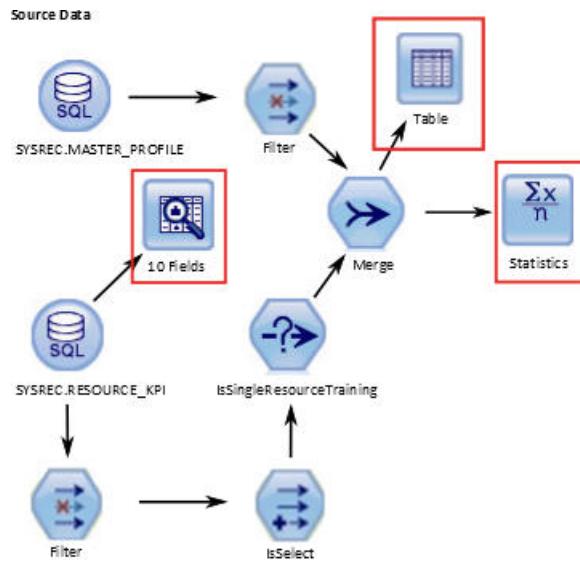


Figure 71. Example source data stream

In the diagram, the highlighted red boxes indicate possible ways that we can interpret the data. For example, the Statistics node addresses the summary statistics for individual fields and the correlations between fields. The Data Audit node provides a comprehensive first look at the data and is presented in an easy-to-read matrix. This matrix can be sorted and used to generate full-size graphs and a variety of data preparation nodes.

Data preparation

Data preparation for the Sensor Health predictive model occurs during execution of the SENSOR_HEALTH_DATA_PREP stream.

The SENSOR_HEALTH_DATA_PREP.str data preparation stream extracts the data from Predictive Maintenance and Quality tables and prepares the data to be used in the modeling. The eligible data is exported to a csv file for the modeling. The input data source contains the measurement type actual reading information of machines. The output is a list of machines for which there is enough data and that are eligible for training to identify the patterns.

To prepare for analysis of the health score based on the measurement types, only the machine's measurement type attributes are considered. Each measurement type has a value. The number of times that the value exceeds the upper and lower limits is accounted for. To train the model to identify failure patterns, enough failure data must be available. Machines that do not have sufficient failure data are not eligible for further modeling. Machine names are logged in the file Training_Eligibility_SensorAnalytics_Report.csv. In this file, resources are indicated either with 1 (eligible) or 0 (not eligible).

The following diagrams show an example of a data preparation stream for the Sensor Health predictive model.

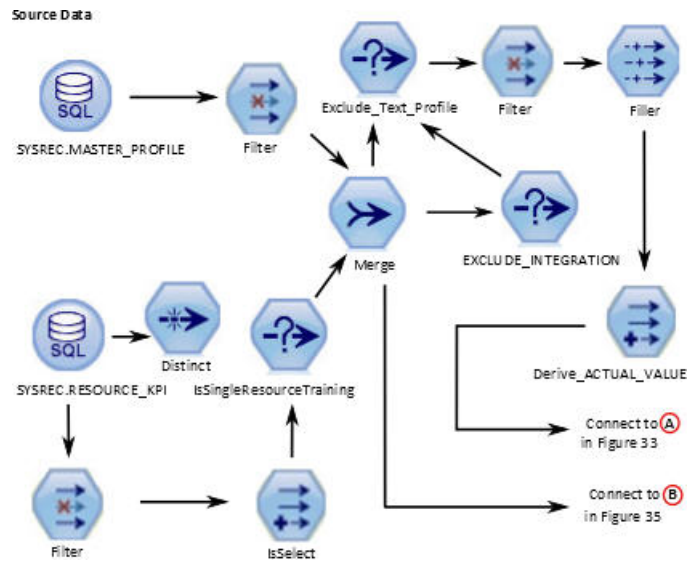


Figure 72. Example data preparation stream for the Sensor Health predictive model - Part 1

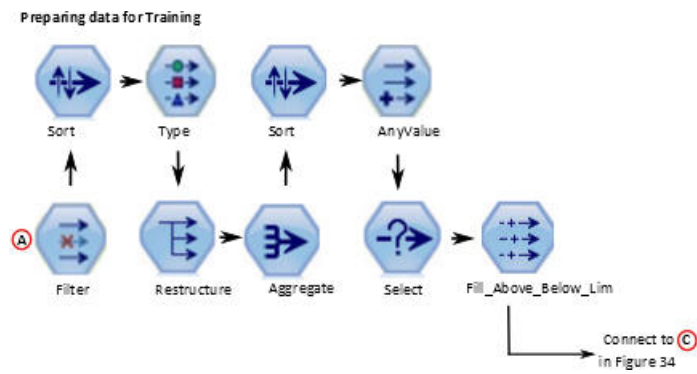


Figure 73. Example data preparation stream for the Sensor Health predictive model - Part 2

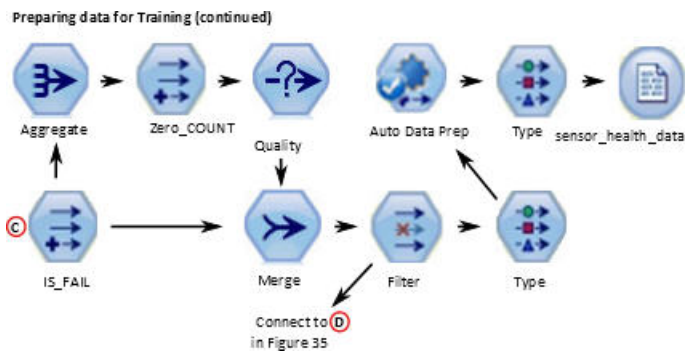


Figure 74. Example data preparation stream for the Sensor Health predictive model - Part 3

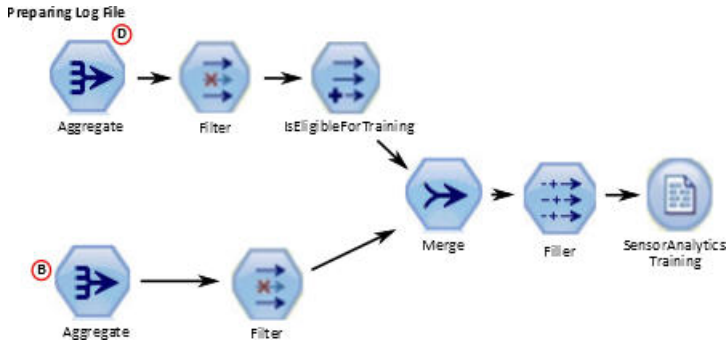


Figure 75. Example data preparation stream for the Sensor Health predictive model - Part 4

Data modeling

The Sensor Health predictive model uses the SENSOR_HEALTH_COMBINED.str stream.

See the following table.

Table 34. The SENSOR_HEALTH_COMBINED.str stream

Stream name	Purpose	Input	Target	Output
SENSOR_HEALTH_COMBINED.str	Predicts failure of equipment based on the measurement types received thru the sensor details, train the models and also refresh them for the scoring service	The machine levels measurement type data received thru the sensor reading systems	IS_FAIL	Health score of the equipment

The following figures show an example of a modeling stream for the Sensor Health predictive model.

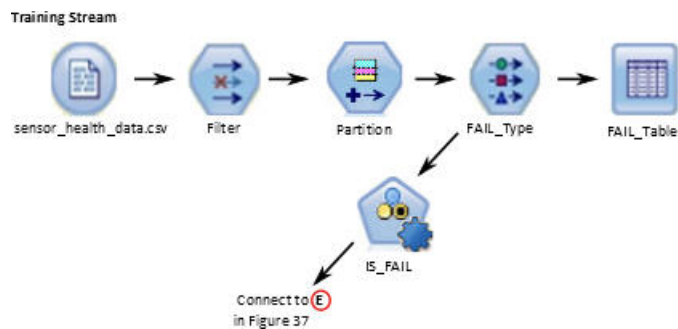


Figure 76. Example modeling stream for the Sensor Health predictive model - Part 1

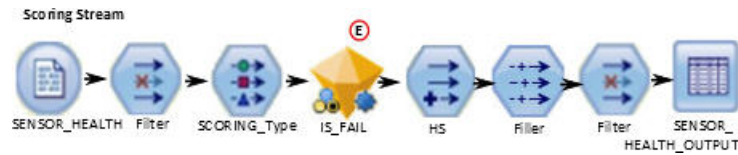


Figure 77. Example modeling stream for the Sensor Health predictive model - Part 2

Depending upon the input data, you might need to consider a different approach for the health score modeling. In addition, the concept of Splits at a resource id level (in Type Node) is introduced because, for each resource, the trained model would be unique.

The value of the health score of an asset is between 0 and 1. The higher the value of the health scores, the better the health of the asset. If the input data model and structure is modified, the health score model must be retrained on the new data.

The health score model is based on the IBM SPSS Modeler auto classification model's confidence. Alternatively, raw and adjusted raw propensity scores can be used to generate such scores. In the model node, there are options to modify the costs, revenue, and weights. This setting depends on the requirements and the available data. Similarly, the data in this case is not balanced. Depending on the data and requirements, balancing might give better results.

Note: Not all of the models support propensity score outputs, especially when splits are enabled.

Evaluation of the model

The model should be verified against the business success criteria established at the beginning of the project.

At this point, most of the data mining activities are complete. However, there is a need to verify the model across the business success criteria that were established at the beginning of the project. We asked the following questions:

- Did the Health scores that were generated from sensor readings provide any helpful insights?
- What new insights or surprises were discovered?
- Were there any issues that were caused by inadequate data preparation or misinterpretation of the data? If there was an issue, we returned to the appropriate stage and rectified the problem.

Deployment

The Sensor Health predictive model uses a combined stream that performs several functions.

The model is developed using parameters which should also be used during deployment. Some parameters are configured in the downstream applications. If the parameter values are passed when the stream is executed, these values are used. Otherwise, default values are used. See the following figure.

Name	Long name	Storage	Value
IS_1_RES_TRAIN	Resource Training required	Integer	0
RESOURCE_ID	Resource identifier	Integer	595

Figure 78. Parameters used for Deployment

If there is a provision to train one resource at a time, the resource id is passed along with the flag value.

This combined stream performs the following functions:

- helps to train the models
- refreshes data for the scoring service
- uses auto-modeling to identify the best suitable model
- produces health score output that measures the probability of machine failure

The stream has multiple execution branches that use scripts to sequence the parameters. Note that the scripts being referenced appear in the Execution tab. See the following figure.

The screenshot shows the 'Deployment' tab of a software interface. At the top, there are several tabs: Options, Messages, Parameters, Deployment (highlighted), Execution, Globals, Search, Comments, and Annotations. Below the tabs, the 'Deployment type' is set to 'Model Refresh'. A description below reads: 'Model Refresh: enables creation of streams and scenarios supporting scoring, building and refresh features'. Under 'Deployment Settings', there are three sections: 'Scoring node' with a dropdown menu showing 'SENSOR_HEALTH_SCORE' and a 'Scoring Parameters...' button; 'Modeling node' with a dropdown menu showing 'IS_FAIL' and a 'Model Build Parameters...' button; and 'Model nugget' with a dropdown menu showing 'IS_FAIL'.

Figure 79. Refreshing the data for the scoring service

The stream is auto-generated when a training instance occurs and, for the real time scoring, in the SENSOR_HEALTH_SCORE service which is invoked by the IIB flows.

Recommendations

The Sensor Health predictive model provides recommendations for each asset.

Sensor analytics recommendations are produced using the real-time mode of invocation. In the invocation mode, the stream is developed using the ADM and a SENSOR_RECOMMENDATION service is configured for scoring services. The service is invoked to receive a recommendation for each asset. See the following figure.

Urgent Inspection	HS101
HS >= 0.7	
Need Inspection	HS102
HS BETWEEN 0.4 and 0.7	
Remainder	HS103

Figure 80. Recommendation settings

Depending on the Health score calculated from the Modeler, a recommendation of an Urgent Inspection (HS101) may be produced. For each HS101 code, a trigger is sent to Maximo to create the work order.

The Top Failure Reason predictive model

The Top Failure Reason predictive model helps you identify the top failure predictors for a given asset in order of their importance. You can then further analyze the identified reasons or parameters to assist in a guided trail from cause or root cause analysis to its respective pattern discovery.

This model is used to analyze and discover the top percentile and number of parameters which are influential in predicting a machine's failure (or optimal health) and their relative importance.

Understanding the data

The Top Failure Reason predictive model uses the event and master tables from the IBM PMQ database to get the sensor data available for each resource at a given point in time. It also gathers the defective and failure information.

The performance indicator table RESOURCE_KPI contains aggregated values for each day. You can use it to prepare for model training and scoring. The MASTER_PROFILE_VARIABLE and the MASTER_MEASUREMENT tables are used to help identify the specific profiles which are considered as parameters and will be considered for further analysis.

Preparing the data

Preparation for the Top Failure Reason predictive model includes merging data, selecting a sample subset, deriving new attributes, and removing the unwanted fields.

Depending on the data and identified goals, in this phase of data preparation the following tasks are performed:

- Merging data sets and records of the master and the events data
- Selecting a sample subset of data, identifying only the specified resource and profiles
- Deriving new attributes for each of the selected profiles based on the parameters
- Removing unwanted fields that are not required for further analysis

The measurements used as parameters are based on the data understanding. They are kept as parameters so they can be modified later, based on the data set. In the IIB layer, only the resource id is available.

Modeling the data

The prepared data is now considered for the modeling exercise. The target is set as the IS_FAIL variable and uses the Logistic regression model to obtain a percentile or probability value.

See the following figure.

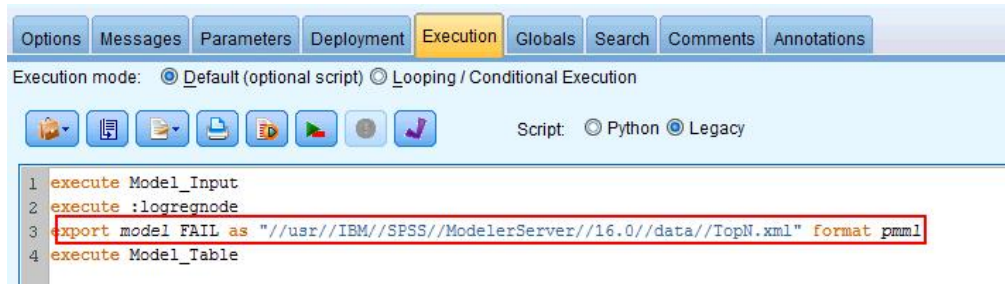


Figure 81. The Execution tab for the modeling stream

The stream has multiple execution branches that use scripts to sequence the parameters. You can see the scripts being referenced in the Execution tab. The important point here is to have exported the model FAIL in the pmml format. This is consumed in the TopN_XML stream to obtain the appropriate predictive importance of each profile.

Evaluation

The Top Failure Reason predictive model should be verified against the business success criteria established at the beginning of the project.

The Cumulative Gains chart shows the advantage of using a predictive model instead of a default, random model. The random model (depicted by a red line in the following figure) shows an equal ratio of percent gain (that is, the selection of entities of interest) to the percent of the total number of entities processed. Therefore, the red line has a 45 degree slope and the gains percentage is equal to the population percentile.

Cumulative Gains charts always start at 0 percent and end at 100 percent. In the following Cumulative Gains chart, the percentage gain rises from zero percent to one hundred percent as the percentage of failures rises from zero percent to forty percent. Continuing after the forty percent failure rate, there are no gains until one hundred percent of the assets have failed.

A good predictive model has a steeper slope than the random model. When using a predictive model, the goal is to categorize and predict more entities of interest than would be done randomly. The model shown in the following figure can predict all the entities of interest by incorporating only 40 percent of the total population.

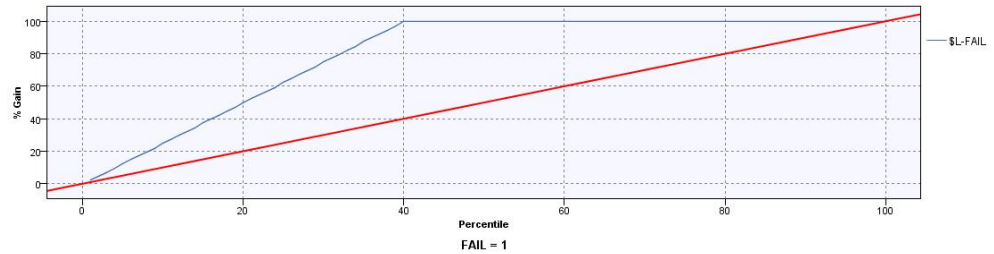


Figure 82. A Cumulative Gains chart

For example, a training and validation set contains only 2 percent of defective parts. Using a random selection model, we would have to select 100 percent of the parts to identify the intended 2 percent of the faults. However, using the model in the preceding figure, we only need to select the top 40 percent of the parts that are most likely defective. This will ensure that all 2 percent of the intended defective parts (equivalent to 100 percent of gain) are covered in our target set.

Deployment

The output of the model calculates the cumulative sum of all the predictive importance values. These values are exported to a csv file. The IIB flow loads the csv file into the profiles table that will be consumed in the Cognos charts.

Note: You can alter the values at each resource level by exposing them to IIB and making a mechanism for picking the correct parameters for each resource. Otherwise, for ad-hoc purposes, the parameters may be changed and triggered manually for each required resource. Also, the output table contents must be deleted manually for this operation, in case the data for the same resource exists from an earlier run.

The Feature-based predictive model

The Feature-based predictive model produces a resource's predicted health score and optimal maintenance period, and recommends inspection or changes to maintenance schedules.

Sample features supported in IBM Predictive Maintenance and Quality

The following list describes sample features that Predictive Maintenance and Quality supports.

- Based on the raw sensor reading of a single sensor.
For example, <Load>-<RAW> = Raw Sensor reading for Load
- Based on computations that involve a single sensor reading.
For example, <FlowMeter>-<OilFlow> = $\text{delta}(\log(\langle\text{FlowMeter}\rangle-\langle\text{RAW}\rangle+2))$
Subtypes under this class of features include mathematical operators, system-defined mathematical functions, logical operators, comparison operators, If Then-Else conditions, and time functions.
- Based on computations that involve more than one sensor, and/or sensor+ time stamp types.
For example, <FlowMeter>-<OilFlowWeighted> = $\text{Power}(\text{delta}(\log(\langle\text{FlowMeter}\rangle-\langle\text{RAW}\rangle+2)) / \text{delta}(\text{Mean}(\langle\text{RPM}\rangle-\langle\text{RPM-FEATURE-1}\rangle+2))$
- If Else condition based on one or more sensors and a time stamp

For example, `<current>-<overload>= If (month(timestamp) In (2,3,4) and <Current>-<Raw> > <Static>- <SummerOverload>) OR (month(timestamp) Not In (2,3,4) and <Current>-<Raw> > <Static>- <WinterOverload>) THEN 1 Else 0`

- Based on sensor reading and nameplate parameters.

For example, `<Static>-<OverloadThreshold> = 0.8 * <Static>-<RatedKVA> <Load>-<OverloadFactor> = Log (0.9 * <Load>-<RAW>/<Static>-<RatedKVA>)`

- Based on nameplate parameters.

For example, `<Load>-<RatedVA> = <Static>-<RatedKVA>*1000`

- Logical comparison and condition-based features.

For example, `<Load>-<WeightedOverload> = If <Load>-<OverloadFactor> > <Static>-<OverloadThreshold> then <Measurement-2><Feature-3>*0.75 else 0`

Features are modeled as profile calculations. The following calculations are supported:

Comparison operators

Various comparison operators are supported.

Logical operators

Various logical operators are supported.

Date calculations

Various date calculations are supported, including Day of Month and Year.

Date math calculations

Various date mathematical operations are supported.

If Else calculations

You can define If Else conditions.

Math calculations

Simple mathematical calculations are supported, such as addition and subtraction.

Advanced math calculations

Mathematical functions are supported, such as cos, sin, log, and exp.

Measurement text does not contain count

Users can check whether an observation text matches a string.

Substation cooling stage analysis and substation overload analysis

These calculations apply to substation transformers.

Derived preservative kind, derived species type, and pole analysis

These calculations apply to poles.

Data model changes for Feature-Based Analytics

IBM SPSS requires both raw sensor readings and the computed features that are derived by using sensor readings in consistent way, from a consistent data source (table), and on same grain as event.

For Feature-Based Analytics, a new profile table, `EVENT_PROFILE`, is supported in Predictive Maintenance and Quality, which allows users to store profiles at the same grain of the event. All features that are computed for the Feature-Based Analytics model are stored in the `EVENT_PROFILE` table.

Features that are aggregated at day and life time grain are stored respectively in the RESOURCE_KPI and RESOURCE_PROFILE tables.

SPSS also supports resource sub type level modeling. As a result, there is a new table in Predictive Maintenance and Quality, called MASTER_PROFILE_MAPPING, that holds mappings between resource sub types and profile mappings. Predictive Maintenance and Quality master data loading flows are used to load data in the MASTER_PROFILE_MAPPING table.

Orchestration rules

The data preparation for Feature-Based Analytics is done through Predictive Maintenance and Quality standard event processing flows. Events for Feature-Based Analytics must have the event type FEATURE. When feature events are processed by the orchestration engine, the following orchestration steps are executed by the specified adapter.

1. Service Adapter

This step is executed only when Dissolved Gas Analysis measurements are reported. This step is bypassed for all other measurement types. This step performs computations for gas measurement data.

2. Profile Adapter

This step computes various features on raw reading data, and stores the computed results in KPI and profile tables.

3. Scoring Adapter

This step invokes the SPSS Feature-Based Analytics model through the Representational State Transfer (REST) interface.

4. Scoring Adapter

This step invokes the SPSS Integration Analytics model through the REST interface.

5. Service Adapter

This step performs post-processing of Integration Analytics results.

6. Service Adapter

This step creates a work order in IBM Maximo if the Integration Analytics model provides Urgent Inspection as the recommendation. Work orders are created in Maximo through the Service Adapter.

Training

Feature-Based Analytics training is timer-based, and it is configured through generic batch orchestration. Generic batch orchestration provides capabilities to run the scheduler or invoke any SPSS batch job by taking inputs from a configurable XML file, instead of developing separate message flows for each use case implementation.

Feature-Based Analytics training is scheduled to trigger every 90 days. If you must schedule training earlier, modify the orchestration definition XML file. Modify the values in the <scheduler> element as shown in the the following figure.

```

<!-- Orchestration for FBA Training -->
<orchestration>
  <Identifier>FHSTrigger</Identifier>
  <scheduler>
    <scheduled_time>02:00:00</scheduled_time>
    <queue_name>PMQ.FHSTIMER.IN</queue_name>
    <duration_in_days>90</duration_in_days>
  </scheduler>
  <!-- Webservice configuration for FBA Training-->
  <webservice>
    <url>http://localhost:9080/process/services/ProcessManagement</url>
    <jobLocationURI>spsscr:///?id=5691007b1cf528e700000149562efe098639</jobLocationURI>
    <notificationEnabled>true</notificationEnabled>
  </webservice>
</orchestration>

```

Figure 83. Location in the orchestration definition file where to modify the training schedule

Input data for training

Feature-based health score and days to maintenance training requires a combination of input data.

The following list describes the input data that the feature-based model requires.

- raw sensor events
- static or nameplate parameters
- day level KPI data
- lifetime parameters
- domain-specific features, which are computations on a combination of one or more raw events, static parameters, or lifetime parameters

Sensor data and features are processed and loaded into the Predictive Maintenance and Quality data store through the Predictive Maintenance and Quality event load flow, as specified in the orchestration definition file. Feature calculations are configured in IBM Integration Bus (IIB), or implemented in SPSS batch jobs.

Minimum data requirements

The minimum data that feature-based analytics requires is three failure and three non-failure events per resource, or per resource sub type for sub type level models, with optimal data quality for other parameters.

The input data can be at event grain, day grain, or lifetime grain. For day grain data, any KPI date with even a single failure event in that day is considered a failure date. It is recommended to use noise treatments, as the training event uses data too. For event grain data, users can flag events before failure as fail events for realistic modeling.

Resources that do not satisfy the minimum data requirements are not trained, and they are logged in the Training Eligibility report as 0 against their eligibility to be trained. Any attempt to score these resources returns a health score of -1 or a predicted days to maintenance of -9999. These results are identified and dropped by IBM Integration Bus (IIB), and they are not available in databases for downstream applications.

Note: The data requirements are only an implemented bare minimum, and they do not guarantee an optimally trained model.

For resources that satisfy the bare minimum data requirements, IBM SPSS can sometimes give errors or fail some of the ensemble models. The errors or failures occur when the data quality is not good, or for other reasons. Under such conditions, the data volume and quality must be improved. As a last resort, you can remove the failing models from the expert modeler.

Resource sub type level modeling

Sub type level modeling is sometimes needed to supplement resource level modeling.

In many cases, equipment failures occur after a long duration, and they are replaced or refurbished after the failure. Because IBM Predictive Maintenance and Quality 2.0 Sensor Analytics is trained per resource, there are no trainable resources for these failures.

Users can have a custom segmentation model that will segment resources, even within a type, based on similar characteristics, and use that segment information within the model. Predictive Maintenance and Quality 2.5 supports segmentation at a resource sub type level.

During scoring, the health score and forecasted days to maintenance come from the most specific model that was trained. For example, for a resource, if the resource level model was not trained, then the resource sub type model is used for scoring. If the resource level model was trained, then that model is used for health scores and forecasted days to maintenance.

Training job

The `IBMPMQ_SENSOR_FEATURE_BASED_ANALYTICS` job calls the data preparation stream, followed by training and refreshing the health score and forecasted days to maintenance models.

The following diagram shows the `IBMPMQ_SENSOR_FEATURE_BASED_ANALYTICS` job.

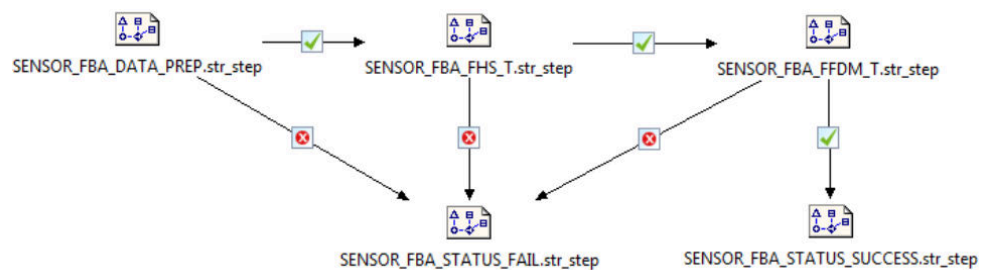


Figure 84. The `IBMPMQ_SENSOR_FEATURE_BASED_ANALYTICS` job

At the end of the job, the file `Training_FBA_in.csv` is returned to the `\root\integrationin` directory. The file contains two fields, System Timestamp and Status, which are read by the IBM Integration Bus (IIS).

If there is failure at any step in the job, the file contains `FAIL` as the status. When all steps are completed successfully, the file contains `SUCCESS` as the status.

Data preparation

Data preparation for the Feature-based predictive model occurs during execution of the `SENSOR_FBA_DATA_PREP.str` stream.

The following table describes the `SENSOR_FBA_DATA_PREP.str` stream.

Table 35. The `SENSOR_FBA_DATA_PREP.str` stream

Stream name	Purpose	Input	Output
<code>SENSOR_FBA_DATA_PREP.str</code>	A Data preparation stream extracts the data from IBM Predictive Maintenance and Quality tables, and prepares the data to be used in the modeling. The eligible data is exported to a csv file for the modeling.	The input data source contains raw sensor event data, day level and lifetime KPIs, and static or nameplate parameters with appropriate linking to the master data tables. It also references the IBM Maximo, or other plant maintenance systems, work orders that were converted into profiles for the actual, planned, and scheduled maintenance dates for breakdown and planned maintenance.	A list of machines for which there is enough data and that are eligible for training to identify the patterns. Transformed data input to modeling for the eligible resources.

This stream prepares the data for analysis of the health score and forecasted days to maintenance, based on raw sensor data, day and lifetime KPI data, and static or nameplate parameters. In addition, to train the model to identify failure patterns, enough failure data must be available.

Machines that do not have sufficient failure data are not eligible for further modeling. Machine IDs are logged in the `Training_Eligibility_SensorFBA_Report.csv` file. In this file, resources are indicated either with 1 (eligible) or 0 (not eligible).

Based on the specific domain and equipment data available, if data or features are added, deleted, or modified, then the data preparation stream must be modified to incorporate the changes. In addition, the framework processes both numeric and categorical features in the final modeling input. There is also a provision to enable or disable certain features out of the input. This provision is configured at a resource sub type level in the `MASTER_PROFILE_MAPPING` table.

Data modeling

The Feature-based predictive model uses different streams for health score modeling and for forecasted days to maintenance modeling.

Health score modeling

The following table describes the `SENSOR_FBA_FHS_T.str` stream that is used for health score modeling.

Table 36. The *SENSOR_FBA_FHS_T.str* stream

Stream name	Purpose	Input	Target	Output
SENSOR_FBA_FHS_T.str	Predicts the failure of equipment based on the measurement types that are received through the features, trains the models, and refreshes them for the scoring service	Transformed data (raw event data, day level and lifetime KPIs, and static or nameplate parameters) from the data preparation stream	IS_FAIL	Health score of the equipment

Depending on the input data, you might have to consider a different approach for the feature-based health score modeling. In addition, the concept of Splits is introduced at a tenant ID, location ID, resource sub type, and resource ID level in the Type Node, because the trained model must be unique for each combination of tenant, location, and resource.

The value of the health score of an asset is 0 - 1. The higher the value of the health scores, the better the health of the asset. If the input data model and structure is modified, the health score model must be retrained on the new data.

The health score model is based on the IBM SPSS Modeler auto classification model's confidence. Alternatively, raw and adjusted raw propensity scores can be used to generate such scores. In the model node, there are various modeling technique-specific settings. These settings can be configured based on the requirements and the available data. Similarly, the data in this case is not balanced. Depending on the data and requirements, balancing might give better results.

Note: Not all of the models support propensity score outputs, especially when splits are enabled.

Forecasted days to maintenance modeling

The following table describes the *SENSOR_FBA_FFDM_T.str* stream that is used for forecasted days to maintenance modeling.

Table 37. The *SENSOR_FBA_FFDM_T.str* stream

Stream name	Purpose	Input	Target	Output
SENSOR_FBA_FFDM_T.str	Predicts forecasted days to maintenance based on the measurement types that are received through the features, trains the models, and refreshes them for the scoring service	Transformed data (raw events, day level and lifetime KPIs, and static or nameplate parameters) from the data preparation stream	DTM (Days to maintenance based on the maintenance or failure records)	Forecasted days to the next maintenance for the equipment

Depending on the input data, you might have to consider a different approach for the feature-based forecasted days to maintenance modeling. In addition, the concept of Splits is introduced at a tenant ID, location ID, resource sub type, and resource ID level in the Type Node, because the trained model must be unique for each combination of tenant, location, and resource.

The value of the forecasted days to maintenance determines the optimal maintenance period. The lower the value, the earlier the maintenance is recommended.

The forecasted days to maintenance are based on the SPSS Modeler auto numeric model. In the model node, there are various modeling technique-specific settings. These settings can be configured based on the requirements and available data. Similarly, the data in this case is not balanced. Depending on the data and requirements, balancing might give better results.

Deployment

In its data preparation stream, the Feature-based predictive model exposes certain parameters that can be used to modify the input data that is created for modeling.

Some parameters are configured in the downstream applications. If the parameter values are passed when the stream is executed, these values are used. Otherwise, default values are used. See the following figure.

Name	Long name	Storage	Value	Type
IS_1_RES_TRAIN	Train for a Single Resource	Integer	0	(no values)
RESOURCE_ID	Which Resource to train for...	Integer	461	(no values)
PROFILE_PLAN_AMC	PROFILE_VARIABLE_CD...	String	AMC	(no values)
PROFILE_PLAN_SMC	PROFILE_VARIABLE_CD...	String	SMC	(no values)
PROFILE_BREAKDOWN_BC	PROFILE_VARIABLE_CD...	String	BC	(no values)
R_CENSURING	RightCensuring (Value>1) ...	Real	1.0	(no values)
L_CENSURING	LeftCensuring (Value<1) B...	Real	0.999	(no values)

Figure 85. Parameters used for Deployment

In IBM SPSS, you can find all of these parameters, but IBM Integration Bus (IIB) exposes only the RESOURCE_ID and the IS_1_RES_TRAIN flag in the IBM Predictive Maintenance and Quality release.

If there is a requirement to train one resource at a time, the resource id is passed along with the flag value.

The deployment type for both the health score and forecasted days to maintenance models is selected as **Model Refresh**, which supports scoring, building, and refresh features.

The following figure shows the deployment type for the health score model.

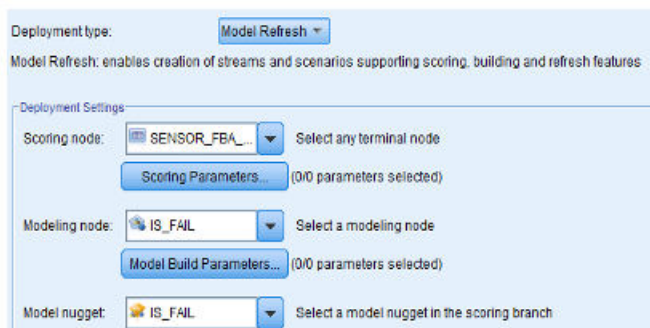


Figure 86. Refreshing the data for the scoring service

The following figure shows the deployment type for the forecasted days to maintenance model.

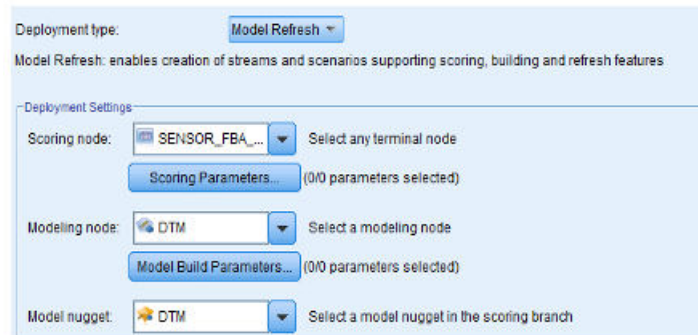


Figure 87. Refreshing the data for the scoring service

Recommendations

The Feature-based predictive model provides recommendations for each asset.

Feature-based analytics recommendations are produced by using the real-time mode of invocation. In the invocation mode, the stream is developed by using the ADM, which internally calls the health score and forecasted days to maintenance models, and a feature-based analytics service is configured for scoring services.

The service returns the health score (FHS), Forecasted Days to Maintenance (FFDM), deviation (DEV_FFDM), deviation percentage (DEV_FFDM_PCT) when compared to schedule maintenance (when available), and a recommended action (RECOMMENDATION) for each asset.

Depending on the health score, forecasted days to maintenance, and deviation percentage, the final recommendation is produced.

The following diagram shows an example of feature-based analytics recommendations.

Rule name	Allocate to	Insert rule	Sort	Remove
1 No Forecast Available	4001			
2 Urgent Inspection	FBA101			
3 Need Monitoring	FBA102			
4 Within Limits	FBA103			
5 No Schedule Available	4002			
6 Maintenance as Scheduled (0 to 10)	3001			
7 Maintenance as Scheduled (-10 to 0)	3002			
8 Prognose Maintenance (-25 to -10)	2001			
9 Prognose Maintenance (-30 to -25)	2002			
10 Prognose Maintenance (-75 to -50)	2003			

No Forecast Available

FHS IS NULL

OR FHS = -1

FFDM IS NULL

OR FFDM = 9999

Urgent Inspection

FHS BETWEEN 0.0 and 0.4

OR FFDM BETWEEN -9998.0 and 3.0

Figure 88. Recommendation settings for the Feature-based predictive model

The Integrated predictive model

The Integrated predictive model produces a predicted health score and days to maintenance for each asset or process at a site. The health score is used to determine an asset's performance.

The health score determines how likely an asset is to fail. This health score model can continuously monitor machine or asset health and predict potential machine faults in real time. It uses historical defect and maintenance data, along with

results of the other Analytics models, to determine the integrated health score and forecasted days to maintenance of an asset. The integrated model can also be used to predict the future health of an asset.

Input data for training

Integrated health score and days to maintenance training requires a combination of input data.

The following list describes the input data that the integrated model requires.

- sensor health score
- maintenance health score and predicted days to maintenance
- feature-based maintenance health score and predicted days to maintenance
- user-planned or user-scheduled days to maintenance, if available

Minimum data requirements

The minimum data that integrated analytics requires is three failure and three non-failure events per resource, or per resource sub type for sub type level models, with optimal data quality for other parameters.

The input data can be at event grain, day grain, or lifetime grain. For day grain data, any KPI date with even a single failure event in that day is considered a failure date. It is recommended to use noise treatments, as the training event uses data too. For event grain data, users can flag events before failure as fail events for realistic modeling.

Resources that do not satisfy the minimum data requirements are not trained, and they are logged in the Training Eligibility report as 0 against their eligibility to be trained. Any attempt to score these resources returns a health score of -1 or a predicted days to maintenance of -9999. These results are identified and dropped by IBM Integration Bus (IIB), and they are not available in databases for downstream applications.

Note: The data requirements are only an implemented bare minimum, and they do not guarantee an optimally trained model.

For resources that satisfy the bare minimum data requirements, IBM SPSS can sometimes give errors or fail some of the ensemble models. The errors or failures occur when the data quality is not good, or for other reasons. Under such conditions, the data volume and quality must be improved. As a last resort, you can remove the failing models from the expert modeler.

Resource sub type level modeling

Sub type level modeling is sometimes needed to supplement resource level modeling.

In many cases, equipment failures occur after a long duration, and they are replaced or refurbished after the failure. Because IBM Predictive Maintenance and Quality 2.0 Integrated Analytics is trained per resource, there are no trainable resources for these failures.

Users can have a custom segmentation model that will segment resources, even within a type, based on similar characteristics, and use that segment information within the model. Predictive Maintenance and Quality 2.5 supports segmentation at a resource sub type level.

During scoring, the health score and forecasted days to maintenance come from the most specific model that was trained. For example, for a resource, if the resource level model was not trained, then the resource sub type model is used for scoring. If the resource level model was trained, then that model is used for health scores and forecasted days to maintenance.

Training job

The `IBMPMQ_INTEGRATED_FEATURE_BASED_ANALYTICS` job calls the data preparation stream, followed by training and refreshing the health score and forecasted days to maintenance models.

The following diagram shows the `IBMPMQ_INTEGRATED_FEATURE_BASED_ANALYTICS` job.

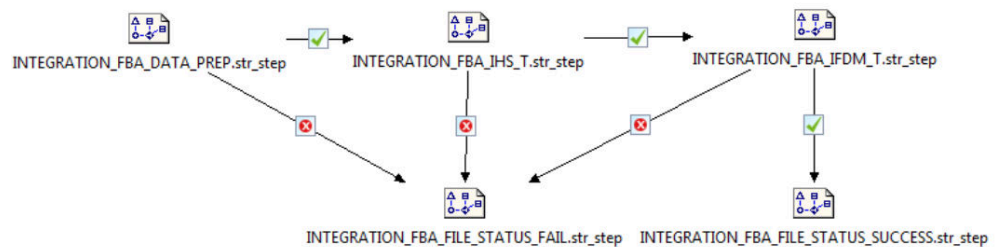


Figure 89. The `IBMPMQ_INTEGRATED_FEATURE_BASED_ANALYTICS` job

At the end of the job, the file `Training_IHS_in.csv` is returned to the `\root\integrationin` directory. The file contains two fields, System Timestamp and Status, which are read by the IBM Integration Bus (IIS).

If there is failure at any step in the job, the file contains FAIL as the status. When all steps are completed successfully, the file contains SUCCESS as the status.

Data preparation

Data preparation for the Integrated predictive model occurs during execution of the `INTEGRATION_FBA_DATA_PREP.str` stream.

The Integrated Analytics model produces a final predicted health score and forecasted days to maintenance for each asset at a site. The health score is used to determine an asset's performance.

The health score determines how likely an asset is to fail (inversely proportional), and the forecasted days to maintenance determines optimal days to the next maintenance. Together, these models can continuously monitor machine or asset health, and predict potential machine faults and/or optimal maintenance schedules in real time.

The `INTEGRATION_FBA_DATA_PREP.str` stream is described in the following table.

Table 38. The `INTEGRATION_FBA_DATA_PREP.str` stream

Stream name	Purpose	Input	Output
<code>INTEGRATION_FBA_DATA_PREP.str</code>	A Data preparation stream extracts the data from IBM Predictive Maintenance and Quality tables, and prepares the data to be used in the modeling. The eligible data is exported to a csv file for the modeling.	The input data source contains sensor, maintenance, and feature-based analytics health scores and forecasted days to maintenance information of machines, in addition to the scheduled maintenance details.	A list of machines for which there is enough data and that are eligible for training to identify the patterns. Transformed data input to modeling for the eligible resources.

This stream prepares the data for analysis of the health score and forecasted days to maintenance, based on the output of the other analytics models, namely Sensor (HS), Sensor FBA (FHS and FFDM), Maintenance (MHS and FDM), and scheduled days to maintenance (SDM) when available. In addition, to train the model to identify failure patterns, enough failure data must be available.

Machines that do not have sufficient failure data are not eligible for further modeling. Machine IDs are logged in the `Training_Eligibility_IntegrationFBA_Report.csv` file. In this file, resources are indicated either with 1 (eligible) or 0 (not eligible).

Orchestration rules

Integrated Analytics takes input from Sensor, Feature-Based, and Maintenance Analytics.

In IBM Predictive Maintenance and Quality, the Integration Analytics model is called after Feature-Based Analytics, through the same orchestration rule configured for the event type FEATURE.

The Integration Analytics model is invoked through the Scoring adapter, which invokes the model through the Representational State Transfer (REST) interface. The Scoring adapter is exposed by Analytics Solutions Foundation.

IBM SPSS exposes a single service to invoke the scoring model and the ADM service. The service returns the integrated health score (IHS), Integrated Forecasted Days to Maintenance (IFDM), deviation (DEV_IFDM), deviation percentage (DEV_IFDM_PCT) when compared to schedule maintenance (when available), and a recommended action (RECOMMENDATION) for each asset. In Predictive Maintenance and Quality, the recommendation is processed as an event.

SPSS results are stored in the `EVENT_PROFILE` and `RESOURCE_PROFILE` tables.

Predictive modeling

The Integrated predictive model uses different streams for health score modeling and for forecasted days to maintenance modeling.

Health score modeling

The following table describes the `INTEGRATION_FBA_IHS_T.str` stream that is used for health score modeling.

Table 39. The INTEGRATION_FBA_IHS_T.str stream

Stream name	Purpose	Input	Target	Output
INTEGRATION_FBA_IHS_T.str	Predicts the failure of equipment based on Sensor, Feature-based, and Maintenance Analytics model-based health scores, in addition to scheduled and forecasted maintenance details. Trains the models, and refreshes them for the scoring service.	The input data source contains the health scores from Sensor (HS), Feature-based (FHS), and Maintenance (MHS) models, in addition to the scheduled (SDM) and forecasted maintenance (FDM from Maintenance model and FFDM from the Feature-based model) results	IS_FAIL	Integrated health score of the equipment

Depending on the input data, you might have to consider a different approach for the integrated health score modeling. In addition, the concept of Splits is introduced at a tenant ID, location ID, resource sub type, and resource ID level in the Type Node, because the trained model must be unique for each combination of tenant, location, and resource.

The value of the health score of an asset is 0 - 1. The higher the value of the health scores, the better the health of the asset. If the input data model and structure is modified, the health score model must be retrained on the new data.

The health score model is based on the IBM SPSS Modeler auto classification model's confidence. Alternatively, raw and adjusted raw propensity scores can be used to generate such scores. In the model node, there are various modeling technique-specific settings. These settings can be configured based on the requirements and the available data. Similarly, the data in this case is not balanced. Depending on the data and requirements, balancing might give better results.

Note: Not all of the models support propensity score outputs, especially when splits are enabled.

Forecasted days to maintenance modeling

The following table describes the INTEGRATION_FBA_IFDM_T.str stream that is used for forecasted days to maintenance modeling.

Table 40. The INTEGRATION_FBA_IFDM_T.str stream

Stream name	Purpose	Input	Target	Output
INTEGRATION_FBA_IFDM_T.STR	Predicts forecasted days to next maintenance based on Sensor, Feature-based, and Maintenance Analytics model-based health scores, in addition to scheduled and forecasted maintenance details. Trains the models, and refreshes them for the scoring service.	The input data source contains the health scores from Sensor (HS), Feature-based (FHS), and Maintenance (MHS) models, in addition to the scheduled (SDM) and forecasted maintenance (FDM from Maintenance model and FFDM from the Feature-based model) results	DTM (Days to maintenance based on the maintenance or failure records)	Integrated forecasted days to the next maintenance for the equipment

Depending on the input data, you might have to consider a different approach for the integrated forecasted days to maintenance modeling. In addition, the concept of Splits is introduced at a tenant ID, location ID, resource sub type, and resource ID level in the Type Node, because the trained model must be unique for each combination of tenant, location, and resource.

The value of the forecasted days to maintenance determines the optimal maintenance period. The lower the value, the earlier the maintenance is recommended.

The forecasted days to maintenance are based on the SPSS Modeler auto numeric model. In the model node, there are various modeling technique-specific settings. These settings can be configured based on the requirements and available data. Similarly, the data in this case is not balanced. Depending on the data and requirements, balancing might give better results.

Deployment

The Integrated predictive model uses a data preparation stream that determines that the data for modeling is developed by using parameters that must also be used during deployment.

Some parameters are configured in the downstream applications. If the parameter values are passed when the stream is executed, these values are used. Otherwise, default values are used. See the following figure.

Name	Long name	Storage	Value	Type
IS_1_RES_TRAIN	Train for a Single Resource	Integer	0	(no values)
RESOURCE_ID	Which Resource to train for...	Integer	461	(no values)
PROFILE_PLAN_AMC	PROFILE_VARIABLE_CD...	String	AMC	(no values)
PROFILE_PLAN_SMC	PROFILE_VARIABLE_CD...	String	SMC	(no values)
PROFILE_BREAKDOWN_BC	PROFILE_VARIABLE_CD...	String	BC	(no values)
R_CENSURING	RightCensuring (Value>1) ...	Real	1.0	(no values)
L_CENSURING	LeftCensuring (Value<1) B...	Real	0.999	(no values)

Figure 90. Parameters used for Deployment

In IBM SPSS, you can find all of these parameters, but IBM Integration Bus (IIB) exposes only the RESOURCE_ID and the IS_1_RES_TRAIN flag in the IBM Predictive Maintenance and Quality release.

If there is a requirement to train one resource at a time, the resource id is passed along with the flag value.

The deployment type for both the health score and forecasted days to maintenance models is selected as **Model Refresh**, which supports scoring, building, and refresh features.

The following figure shows the deployment type for the health score model.

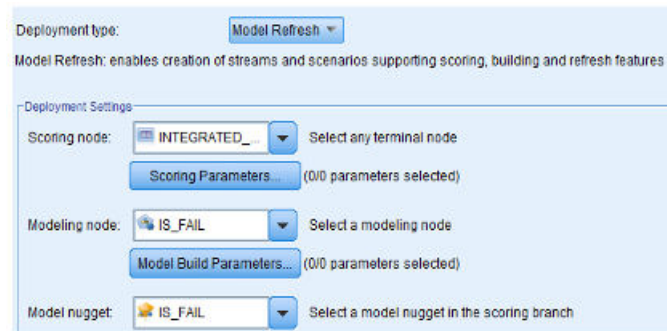


Figure 91. Refreshing the data for the scoring service

The following figure shows the deployment type for the forecasted days to maintenance model.

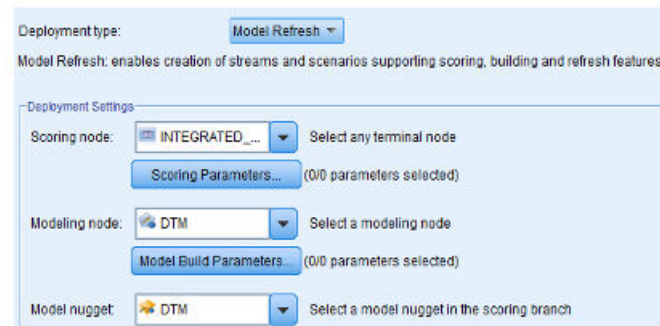


Figure 92. Refreshing the data for the scoring service

Recommendations

The Integrated predictive model provides final recommendations for each asset, based on the results of other analytics models on Sensor, Maintenance, and Feature-based data.

Integrated analytics recommendations are produced by using the real-time mode of invocation. In the invocation mode, the stream is developed by using the ADM, which internally calls the health score and forecasted days to maintenance models, and an INTEGRATED_FBA service is configured for scoring services.

The service returns the integrated health score (IHS), Integrated Forecasted Days to Maintenance (IFDM), deviation (DEV_IFDM), deviation percentage

(DEV_IFDM_PCT) when compared to schedule maintenance (when available), and a recommended action (RECOMMENDATION) for each asset.

Depending on the health score, forecasted days to maintenance, and deviation percentage, the final recommendation is produced.

The following diagram shows an example of Integrated analytics recommendations.

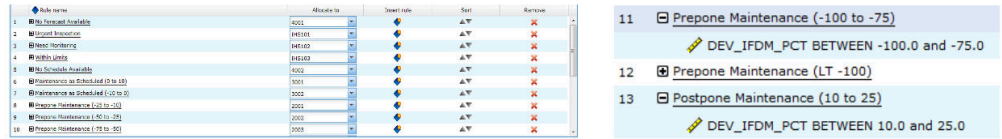


Figure 93. Recommendation settings for the Integrated predictive model

Chapter 9. Recommendations

When an asset or a process is scored and identified as having a high probability of failure, recommendations can be generated.

Define recommended actions by using rules in IBM Analytical Decision Management. Use IBM Analytical Decision Management to understand the drivers that are used to define the rules, and to determine what happens based on the scores received. For example, if a score breaches a threshold, what is the resulting action? You can automate alerts for recommended actions by integrating with other systems or by defining a routing rule to send emails. Depending on the manufacturing execution systems (MES) you use, the recommendation can be acted on automatically. You can also predict the success rate of the corrective action that is based on previous actions.

For information about using IBM Analytical Decision Management, see IBM Analytical Decision Management Knowledge Center (<http://www.ibm.com/support/knowledgecenter/SS6A3P>).

When IBM Predictive Maintenance and Quality generates recommendations, for example, to inspect an asset, you can configure the system so that the recommendation results in a work order that is created by IBM Maximo. The work order is populated with the information needed to complete the task, for example, a device identifier and a location.

Two IBM Analytical Decision Management templates are provided with IBM Predictive Maintenance and Quality:

- Prioritize application template
- Combine application template

Prioritize application template

Use the prioritize application template when you have a good understanding of the predictive analytics scores, and the interaction between the predictive scores. You can use this template to prioritize your business objective that is based on, for example, profit maximization, or downtime minimization.

The template is stored in the following location: `/opt/IBM/SPSS/Deployment/5.0/Server/components/decision-management/Templates/PredictiveMaintenanceQuality.xml`

This template contains the following information that can be customized:

- Input source data: contains the health score and expected device life time data from the IBM SPSS Modeler stream output. Additionally, it contains the calculations such as Mean, Minimum, Maximum values for a particular resource for a specific time stamp.
- Defined rules: resource recommendations are given based on the rules defined. The recommended actions are classified as **Urgent inspection**, **Need monitoring**, or **Within limits**.
- Prioritization: you can define the optimization objective for the business, for example, “profit maximization” or “downtime or loss minimization”.

Combine application template

Use the combine application template to use existing rules along side new predictive scores. This is useful if there are many rules that you do not want to replace with new predictive scores straight away. You can define a precedence structure for these rules to enable them to co-exist.

The template is stored in the following location: `/opt/IBM/SPSS/Deployment/5.0/Server/components/decision-management/Templates/PredictiveMaintenance.xml`

This template contains the following information that can be customized:

- Input source data: contains the health score and expected device life time data from the IBM SPSS Modeler stream output. Additionally, it contains calculations such as Mean, Minimum, Maximum values for a particular resource for a particular time stamp.
- Defined rules: logic-based business rules with appropriate risk points.
- Combine: specify the precedence order when the actions from the business rules and the model do not match.

Preventing scoring for incoming events

You can prevent scoring from being performed by IBM SPSS for incoming events. If you require IBM Maximo work order creation, you must not prevent scoring. By default, scoring is enabled (`SPSSTRIGGER` is set to `TRUE`).

Procedure

1. In the IBM WebSphere MQ Explorer, expand the **Brokers** node, the **MB8Broker** node, the **PMQ1** node, the **PMQEventLoad** node, right-click the **StdEventLoad** item, and click **Properties**.
2. Click **User Defined Properties**.
3. Set the `SPSSTRIGGER` property to `FALSE`. To re-enable scoring, set the `SPSSTRIGGER` property to `TRUE`.

Disabling work order creation

If IBM Maximo is not integrated with your IBM Predictive Maintenance and Quality installation, or if you want to disable work order creation, do the following steps:

Procedure

1. In the IBM WebSphere MQ Explorer, go to **Brokers > MB8Broker > PMQ1**. Right-click the **PMQIntegration** node, and click **Properties**.
2. Click **User Defined Properties**.
3. Set the `MaximoTRIGGER` value to `FALSE`. To re-enable work order creation, set the `MaximoTRIGGER` property to `TRUE`. By default, the `MaximoTRIGGER` property is set to `FALSE`.

Chapter 10. Reports and dashboards

You can customize and extend the reports and dashboards that are supplied with IBM Predictive Maintenance and Quality. You can also design your own reports and dashboards and add them to the menu.

You can use IBM Cognos Report Studio to create scorecards and reports. Before you run the reports, familiarize yourself with the behavior of reports in Report Studio. For example, a star next to a prompt indicates that it is required. For information about how to use Report Studio, see *IBM Cognos Report Studio User Guide*. You can obtain this user guide from IBM Knowledge Center (<http://www.ibm.com/support/knowledgecenter/>).

Tip: Images that are needed for custom and extended reports must be placed in the `/opt/ibm/cognos/analytics/webcontent/Images` folder on the Business Intelligence node. The image URL property value in Report Studio for images placed in this directory is the name of the file.

You can modify the data model for these reports by using IBM Cognos Framework Manager. For more information, see Appendix D, “IBM Cognos Framework Manager model description,” on page 267.

The following table describes the reports available from the Site Overview Dashboard.

Table 41. Reports available from the Site Overview Dashboard

Reports	Description
Overview	Provides a high-level summary of the health of all of your assets at all sites, it shows the key performance indicators (KPIs) with the greatest impact. You can change the detail that is displayed by selecting items from the list boxes. For example, you can change the date and the equipment type.
Top 10 Contributors	Identifies the equipment, locations, and operators responsible for the most failures.
KPI Trending	You can select multiple key performance indicators (KPIs) to be plotted side-by-side in a line chart. You can identify correlations between the KPIs and see whether there is any lagging behavior. For example, if there is a spike in one KPI, how long does it take to impact the other KPIs?
Actual vs Plan	You can monitor how closely the metrics track against the plan. Variances are highlighted.
Equipment Listing	The health score for a site is derived from the lower-level scores from each piece of equipment in the site. This report shows you all the pieces of equipment on the site and the health scores and relevant KPIs for that equipment.
Equipment Outliers	Lists the equipment (or assets) that are performing outside of allowable limits. The measures that are shown differ depending on the equipment, but examples are operating temperature, lateral strain, hydraulic pressure, average value, last value, and control limits.

Table 41. Reports available from the Site Overview Dashboard (continued)

Reports	Description
List Of Recommended Actions	A summary of all recommended actions for each piece of equipment, for the health score measurement.

The following table describes the reports that are available from the Equipment dashboard.

Table 42. Reports available from the Equipment dashboard

Reports	Description
Equipment Profile	A detailed report that shows everything that is known about a piece of equipment: how it is performing today and how it performed in the past.
Equipment Control Chart	Shows the upper and lower control limits and the average limits for selected measures.
Equipment Run Chart	Shows the measures for a particular piece of equipment.
Equipment Outliers	Shows detailed measures for a piece of equipment that shows anomalies.
Event Type History	Lists the events for a device.

The following table describes the reports that are available from the Product Quality Dashboard.

Table 43. Reports available from the Product Quality Dashboard

Reports	Description
Defect Analysis	Shows product defects and inspection rates.
Inspection Rate Analysis	Examines the relationship between inspections and defects over time to find the optimal rate of inspection.
Material Usage By Process	Provides an overview of material usage in the production processes.

Site Overview Dashboard

The Site Overview Dashboard provides a high-level summary of the health of all of your assets at all sites. It shows the key performance indicators (KPIs) with the greatest impact. It contains the Site Summary Report, the Health Score Trend Bar Chart, the Health Score Contributors Pie Chart, and the Incident and Recommendation Analysis Bar Chart.

You can use the following prompt filters in the dashboard. The filters are applied to all reports and charts on the dashboard:

- From Date
- To Date
- Location
- Resource sub type

Site Summary Report

The following table describes the measures for the Site Summary Report.

Table 44. Site summary measures

Measures	Description
Health Score	The evaluation of the health of a resource that is based on predictive models.
Resource Count	The number of resources.
Incident Count	Counts the number of failures that are recorded by resources.
Alarm Count	Counts the number of alarms that are generated by resources.
Recommendation Count	A recommendation is generated by the predictive model when a resource approaches a failure. This measure counts the number of recommendations generated.
MTTR (Mean time to repair)	Average time, for example, in hours between the occurrence of an incident and its resolution, which is calculated by using the following calculation: $\text{Repair Time} / \text{Repair Count}$
MTBF (Mean time between failures)	The average time between equipment failures over a period. For example, the average time a device functions before it fails. It is the reliability rating that indicates the expected failure rate of equipment. It is calculated by using the following calculation: $\text{Operating Hours Delta} / \text{Incident Count}$

The measure source is the resource_kpi table. The measures are displayed by location.

Health Score Trend Bar Chart

The Health Score Trend Bar Chart uses the Health score measure. The measure source is the resource_kpi table. The measure is displayed by date.

Health Score Contributors Pie Chart

The Health Score Contributors Pie Chart uses the Health score measure. The measure source is the resource_kpi table. The measure is displayed by resource.

Incident and Recommendation Analysis Bar Chart

You can use this report to analyze incidents and recommendations.

You can access the Drill Through - Incident and Recommendation Event List drill through report from the Incident and Recommendation Analysis Bar Chart:

Note: The drill through reports are stored in the **Drill Through Reports** folder. The reports in this folder are intended to be run from the main report with which they are associated. Do not run the drill through reports on their own.

The following table describes the measures for the **Incident and recommendation analysis** bar chart. The measure source is the resource_kpi table. The measures are displayed by date.

Table 45. Incident analysis bar chart

Measures	Description
Recommendation Count	A recommendation is generated by the predictive model when a resource approaches a failure. This measure counts the number of recommendations that are generated.
Incident Count	Counts the number of failures that are recorded by resources.

Recommendation and Incident Count Accessibility List

This chart provides the same information as the Incident and Recommendation Analysis Bar Chart in an accessible format.

The Recommendation and Incident Count Accessibility List contains the following drill through reports:

Drill Through - Incident Event List

This report shows the incident event list in tabular form.

Drill Through - Recommendation Event List

This report shows the recommendation event list in tabular form.

Note: The drill through reports are stored in the **Drill Through Reports** folder. The reports in this folder are intended to be run from the main report with which they are associated. Do not run the drill through reports on their own.

Top 10 Contributors dashboard

The Top 10 Contributors dashboard identifies the top 10 equipment, locations, and operators responsible for the most failures.

The following table indicates which dimension is used to display the Actual Value measure in each report.

Table 46. Dimensions that display the Actual Value measure in Top 10 Contributors reports

Report	Dimension
Top 10 Contributors by Resource	Resource
Top 10 Contributors by Location	Location
Top 10 Contributors by Organization	Group Dimension

The Actual Value measure is the aggregated value of an actual event measurement. Depending on the aggregation type of the Profile Variable, it can be calculated by using the following calculation: $\text{sum (Actual Value) / sum (Measure Count)}$ or $\text{sum (Actual Value)}$

The measure source is the resource_kpi table. The measures are displayed by the Location hierarchy.

The following prompt filters are applied to this report:

- Date
- Location
- Resource sub type

- Profile variable

KPI trending report

Users can select multiple key performance indicators (KPIs) to be plotted side-by-side in a line chart. You can identify correlations between the KPIs and see whether there is any lagging behavior. For example, if there is a spike in one KPI, how long does it take to impact the other KPIs?

The KPI trending report uses the Actual Value measure, which is the aggregated value of an actual event measurement. Depending on the aggregation type of the Profile Variable, it can be calculated by using the following calculation: $\text{sum}(\text{Actual Value}) / \text{sum}(\text{Measure Count})$ or $\text{sum}(\text{Actual Value})$. The measure is displayed by the Calendar hierarchy, and the measure source is the `resource_kpi` table.

The following prompt filters are applied to this report:

- Date
- Location
- Resource sub type
- Profile variable

Actual vs plan report

This report monitors how closely the metrics are tracking against the plan. Variances are highlighted when a metric is off-track.

The following table describes the measures, and the measure source for the **Actual vs plan** report.

Table 47. Measures and the measure source in the Actual vs plan report

Measures	Measure description	Measure source
Plan Last Value	The last recorded planned value for the resource. "Planned" is determined by the Value Type	resource_profile table
Actual Last Value	The last recorded actual value for the resource. "Actual" is determined by the Value Type	Resource Profile
Variance	Plan value - Actual Value	Report calculation

The following prompt filters are applied to this report:

- Date
- Location
- Resource sub type
- Profile variable

Equipment listing report

The health score for a site is derived from the lower-level scores from each piece of equipment in the site. Use this report to view all the pieces of equipment on the site and the health scores and relevant key performance indicators (KPIs) for that equipment.

The following table describes the measures for the Equipment listing report. The measure source is the resource_kpi table. The measures are displayed by the Resource hierarchy.

Table 48. Measures in the Equipment listing report

Measures	Description
Health Score	The evaluation of the health of a resource based on predictive models.
Work Order Count	This counts the number of work orders issued. A Work Order is a separate event type from resource measurements.
Incident Count	This counts the number of failures recorded by resources.
Recommendation Count	A recommendation is issued by the predictive model when a resource approaches a failure. This measure counts the number of recommendations that have been issued.
MTBF (Mean time between failures)	The average time between equipment failures over a given period, for example, the average time a device will function before failing. It is the reliability rating indicating the expected failure rate of equipment. It is calculated by using the following calculation: Operating Hours Delta / Incident Count.
MTTR (Mean time to repair)	Average time (for example, in hours) between the occurrence of an incident and its resolution, calculated by using the following calculation: Repair Time / Repair Count

The following prompt filters are applied to this report:

- Date
- Location
- Resource sub type

Outliers report

This reports lists the equipment or assets that are performing outside of allowable limits.

The following table provides the measure details for the Outliers report.

Table 49. Measures in the Outliers report

Measures	Measure description	Measure source
Life to Date Average	The daily average measurement for the resource.	Resource Profile
Upper Control Limit	This is calculated by using the following calculation: [Life-to-Date Average] + [Sigma Level] * [Life-to-Date Standard Deviation]	Report calculation

Table 49. Measures in the Outliers report (continued)

Measures	Measure description	Measure source
Lower Control Limit	This is calculated by using the following calculation: [Life-to-Date Average] - [Sigma Level] * [Life to Date Standard Deviation]	Report calculation
Last Value	The most recent recorded measurement for this resource.	Resource Profile

The following prompt filters are applied to this report:

- Date
- Location
- Resource sub type
- Sigma level

Recommended actions report

This report summarizes all recommended actions for each piece of equipment.

The Recommended actions report uses the Health Score measure, which is the evaluation of the health of a resource that is based on predictive models. The measure is displayed by the Event Observation Resource hierarchy, and the measure source is the event table.

The following prompt filters are applied to this report:

- Date
- Location
- Resource sub type

Equipment dashboard

The Equipment dashboard gives you access to the Equipment profile report, the Equipment control chart, the Equipment run chart, the Equipment outliers chart, and Event type history chart.

Equipment profile report

The Equipment profile report is a detailed report that shows everything that is known about a piece of equipment: how it is performing today and how it performed in the past.

The following table provides the measure description for the **Equipment profile** report. The measure source is the resource_profile table. The measures are displayed by the Profile Variable hierarchy.

Table 50. Measures in the Equipment profile report

Measures	Measure description
Period Minimum	The lowest actual reading that is recorded for the resource measurement this period.
Period Maximum	The highest actual reading that is recorded for the resource measurement this period.

Table 50. Measures in the Equipment profile report (continued)

Measures	Measure description
Period Average	The daily average measurement for the resource.
Last Value	The most recent recorded measurement for this resource.
Period Total	The total actual reading that is recorded for the resource measurement this period.

The following prompt filters are applied to this report:

- Resource sub type
- Resource name
- Resource code
- Location
- Event code

Equipment control chart

The Equipment control chart shows the upper and lower control limits and the average limits for selected measures.

The following table provides the measure details for the **Equipment control chart** report.

Table 51. Measures in the Equipment control chart

Measures	Measure description	Measure source
Life to Date Average	This is an average measurement that is calculated over the life of the resource.	resource_profile table
Upper Control Limit	This is calculated by using the following calculation: [Life-to-Date Average] + [Sigma Level] * [Life-to-Date Standard Deviation]	Report calculation
Lower Control Limit	This is calculated by using the following calculation: [Life-to-Date Average] - [Sigma Level] * [Life-to-Date Standard Deviation]	Report calculation
Measurement	The actual value recorded on an event.	event table

The following prompt filters are applied to this report:

- Resource sub type
- Resource name
- Resource code
- Location
- Event code
- Calendar date
- Start time
- End time
- Measurement type

- Profile variable
- Sigma level

Equipment run chart

The equipment run chart shows the measures for a particular piece of equipment.

The Equipment run chart uses the Measurement measure, which is the actual value that is recorded on an event. The measure source is the event table, and the measure is displayed by the event time hierarchy.

The following prompt filters are applied to this report:

- Resource sub type
- Resource name
- Resource code
- Location
- Event code
- Calendar date
- Start time
- End time
- Measurement type

Equipment outliers

The equipment outliers report shows detailed measures for a piece of equipment that shows anomalies.

The following table describes the measures for the Equipment outliers report. The measures are displayed by the Profile Variable hierarchy.

Table 52. Measures in the Equipment outliers report

Measures	Measure description	Measure source
Life to Date Average	This is an average measurement that is calculated over the life of the resource.	resource_profile
Upper Control Limit	This is calculated by using the following calculation: [Life-to-Date Average] + [Sigma Level] * [Life-to-Date Standard Deviation]	Report calculation
Lower Control Limit	This is calculated by using the following calculation: [Life-to-Date Average] - [Sigma Level] * [Life-to-Date Standard Deviation]	Report calculation
Last Value	The most-recent recorded measurement for this resource.	resource_profile

The following prompt filters are applied to this report:

- Resource sub type
- Resource name
- Resource code

- Location
- Event code

Event type history report

The event type history report lists the events for a device.

The Event type history report uses the Measurement measure, which is the actual value that is recorded on an event. The measure source is the event table, and the measure is displayed by the Event Time, Measurement Type and Event Observation.

The following prompt filters are applied to this report:

- Resource sub type
- Resource name
- Resource code
- Location
- Event code
- Calendar date
- Event type

Product quality dashboard

The Product quality dashboard highlights areas of the production process that are affected by defects, and enables you to see if any relationships exist between the rate of inspections, and the rate of defects.

Defect analysis dashboard

The Defect analysis dashboard provides an overview of product defects and inspection rates. The dashboard is made up of a number of reports that analyze defects by event code, location, and production batch.

Defect summary

This report analyzes product defects and inspection rates.

The following table describes the measures for the **Defect summary** report. The measure source is the process_kpi table. The measures are displayed by the Product hierarchy.

Table 53. Measures in the Defect summary report

Measures	Measure Description
Defect Count	The number of defects that are reported.
Quantity Produced	The quantity that is produced.
Defect Rate	Defect Count divided by Quantity Produced.
Quantity Planned	The quantity that is expected to be produced.
Defect Target	The acceptable number of defects.
Test Failure Rate	Test Failures divided by Number of Tests.
Target Defect Rate	Defect Target divided by Quantity Planned.
Inspection Time	The amount of time that is spent inspecting the product.

Table 53. Measures in the Defect summary report (continued)

Measures	Measure Description
Assembly Time	The amount of time that is spent producing the product.
Inspection Time Rate	Inspection Time divided by the Assembly Time.
Inspection Count	The number of inspections performed.
Inspection Rate	Inspection Count divided by Quantity Produced.
Average Assembly Time	.Assembly Time divided by Quantity Produced.

The following prompt filters are applied to this report:

- Process hierarchy
- Calendar from/to date

Defects by event code

This pie chart shows product defects by event code, also known as failure code.

The Defects by event code report uses the Actual Value measure, which is the aggregated value of an actual event measurement. Depending on the aggregation type of the Profile Variable, it can be calculated by using the following calculation: $\text{sum (Actual Value)} / \text{sum (Measure Count)}$ or $\text{sum (Actual Value)}$

The measure is displayed by the Event Code hierarchy, and the measure source is the process_kpi table.

The following prompt filters are applied to this report:

- Process hierarchy
- Calendar from/to date

Defects by location

This pie chart shows product defects by location.

The Defects by location report uses the Actual Value measure, which is the aggregated value of an actual event measurement. Depending on the aggregation type of the Profile Variable, it can be calculated by using the following calculation: $\text{sum (Actual Value)} / \text{sum (Measure Count)}$ or $\text{sum (Actual Value)}$

The measure is displayed by the Location hierarchy, and the measure source is the process_kpi table.

The following prompt filters are applied to this report:

- Process hierarchy
- Calendar from/to date

Defects by production batch

This pie chart shows product defects by production batch.

The Defects by production batch report uses the Actual Value measure, which is the aggregated value of an actual event measurement. Depending on the aggregation type of the Profile Variable, it can be calculated by using the following calculation:

$\text{sum (Actual Value) / sum (Measure Count) or sum (Actual Value)}$

The measure is displayed by the Production batch hierarchy, and the measure source is the process_kpi table.

The following prompt filters are applied to this report:

- Process hierarchy
- Calendar from/to date

Inspection rate analysis

This report examines the relationship between inspections and defects over time in order to find the optimal rate of inspection.

It is made up of the Defect Summary report, the Defect plan vs actual bar chart, and the Defect rate vs inspection rate line chart.

Defect plan vs actual report

The following table provides the measure details for the Defect plan vs actual report. The measure is displayed by the Product hierarchy, and the measure source is the process_kpi table.

Table 54. Measures in the Defect plan vs actual report

Measures	Measure description
Defect Rate	Defect Count divided by Qty Produced.
Target Defect Rate	The rate of the Defect Target divided by Quantity Planned measure.

The following prompt filters are applied to this report:

- Process hierarchy
- Calendar from and to date

Defect rate vs inspection rate line chart

The following table provides the measure details for the Defect rate vs inspection rate line chart. The measure is displayed by the Calendar hierarchy, and the measure source is the process_kpi table.

Table 55. Measures in the Defect rate vs inspection rate line chart

Measures	Measure description
Defect Rate	Defect Count divided by Quantity Produced.
Inspection Rate	Inspection Count divided by Quantity Produced.

The following prompt filters are applied to this report:

- Process hierarchy

- Calendar from/to date

Material usage by process crosstab

This report provides an overview of material usage in the production processes.

This report includes the Defect Summary report.

This report uses the Period Measure Count, which is the number of measurements that are taken in one period. By default, a period is one day. The measure is displayed by the Material by Type, Supplier, and Batches by Product hierarchies, and the measure source is the material_profile table.

The Process hierarchy prompt filter is applied to this report.

Audit Report

The Audit Report shows the count of rows in the major master data tables.

Note: The Asset count is shown in the Audit Report.

The Audit Report contains the following drill through reports:

Drill Through - Resource List

Lists the resources for a resource type.

Example

For example, the Audit Report shows the count for the Asset resource type. Click on this count number to open the Drill Through - Resource List report that lists all of the assets.

Drill Through - Profile Variables

Lists all measures and key performance indicators that are being tracked in daily profiles and historical snapshots.

Drill Through - Process List

Lists all production processes.

Drill Through - Material List

Lists materials that are used in the production process.

Drill Through - Production Batch List

Lists production batches for a product.

Drill Through - Measurement Type List

Lists measurement types. For each measurement type, the report shows unit of measure and aggregation type.

Note: The drill through reports are stored in the **Drill Through Reports** folder. The reports in this folder are intended to be run from the main report with which they are associated. Do not run the drill through reports on their own.

The following table describes the measures in the Audit Report. The measure source is the report calculation.

Table 56. Measures in the Audit Report

Measures	Measure Description	Hierarchies
Count of Resources by Type	A count of rows in the dimension	Resource by Type

Table 56. Measures in the Audit Report (continued)

Measures	Measure Description	Hierarchies
Count of Materials by Type	A count of rows in the dimension	Material by Type
Count of Profile Variables	A count of rows in the dimension	Profile Variable
Count of Measurement Types	A count of measurement types in the dimension	Measurement Type
Count of Processes	A count of rows in the dimension	Process
Count of Production Batches by Product	A count of rows in the dimension	Batches by Product

Material usage by production batch

This report provides an overview of material usage by production batch. By correlating production batches with defects to material usage by production batch, you can begin to trace the impact of defective materials.

The material usage by production batch report uses the Period Measure Count, which is the number of measurements that are taken in one period. By default, a period is a day. The measure is displayed by the following hierarchies:

- Batches by Product
- Supplier
- Material by Type

The measure source is the `material_profile` table.

The following prompt filters are applied to this report:

- Process hierarchy
- Event code

Maintenance Overview Dashboard

The Maintenance Overview Dashboard provides insights by using existing maintenance data and can include sensor data when your organization's data matures. The Maintenance Overview Dashboard also gives you insight into stable life and rapid wear scenarios.

This report shows the sensor health score, maintenance health score, the integrated health score, and the feature based health score by location and resource for the last current day in the record. The sensor health score is a near real-time value calculated from the sensor reading. The maintenance health score is calculated from the maintenance logs. The sensor health score and the maintenance health score are combined to give the integrated health score.

Recommendations are shown for each health score value to help users take the necessary actions. The prompt filters forecasted days to next maintenance, scheduled days to next maintenance, feature based forecasted days to next maintenance, and integrated forecasted days to next maintenance, along with their deviations, are shown in the report, which help users to proceed with or postpone the scheduled maintenance cycle.

You can set the following prompt filters in this chart:

- Location
- Health Score
- Recommendation
- Absolute Deviation % (Maintenance)
- Absolute Deviation % (Feature based)
- Absolute Deviation % (Integrated)
- Forecasted Days to Next Maintenance
- Scheduled Days to Next Maintenance
- Feature Based Forecasted Days to Next Maintenance
- Integrated Forecasted Days to Next Maintenance
- Event Code
- Resource

The following columns are reported in this chart.

Table 57. Columns in the Maintenance Overview Dashboard

Column	Description
Location	Location of the resource.
Resource Sub Type	Sub type of the resource.
Resource	Identifies the resource.
Serial No	Serial number of the resource.
Health Score	The sensor health score, maintenance health score, and the integrated health score are values between 0.00 and 1.00. The higher the score, the better the performance of the resource.
Days To Next Forecasted & Scheduled Maintenance	The forecast number of days to next maintenance and number of days to next scheduled maintenance. The maximum positive deviation, minimum positive deviation, maximum negative deviation, and minimum negative deviation are also indicated.
Forecast - Schedule Deviation	The difference between the forecast days and the scheduled days.
Recommendation	The recommended action as indicated by the health scores.
Profile Variable CD	Profile Variable codes used in the Maintenance dashboard. The possible values are MHS, IHS, SHS, and FHS.

The Maintenance Overview Dashboard is designed on Materialized Query Tables (MQT) to improve the report response time. The MQTs that are created in the IBMPMQ database are:

- MAINTENANCE_PROFILE
- MAINTENANCE_EVENT

When the underlying tables are loaded, these tables must be refreshed. The crontab job is used to refresh these tables. The job runs ON scheduled basis on a daily basis, and it can be configured according to customer needs. The job runs every midnight to refresh these MQTs.

To see the data in the report immediately, refresh the MQTs manually by running the corntab script manually or by running the REFRESH command against these tables. The REFRESH command syntax is

```
REFRESH TABLE SCHMENAME.MQT_TABLENAME
```

For example, REFRESH TABLE PMQSCH.MAINTENANCE_PROFILE.

For more information about the corntab script, see the IBM Predictive Maintenance and Quality *Installation Guide*.

Maintenance Advance Sorting

Click **Advance Sorting** to drill through to the Maintenance Advance Sorting report. This report displays the same measures as the main report in a tabular format. You can sort on a column by clicking the column header. The prompt values from the main report are used in the Maintenance Advance Sorting report. You can change the prompt values in the Maintenance Advance Sorting report and run it with the new values.

Maintenance Health and Failure Detail Report

Click a resource in the **Resource** column to drill through to the Maintenance Health and Failure Detail Report for the resource.

The prompt values from the main report are used in this chart. You can change the following prompt filters in this chart and run it with the new values:

- From date
- To date
- Location
- Resource

You can include or exclude the following events:

- Breakdown maintenance
- Planned maintenance
- Forecasted maintenance
- Scheduled maintenance

Each event that you include appears as a bar on the chart. The bar indicates the date on which the event occurs. The health score, which is a value between zero and one, is indicated on the y axis. The x axis indicates the date of the health score. Health scores that occur before the current date are historical health scores. Health scores that occur after the current date are forecast health scores. The current health score is shown for the current date.

Click **Top N Failure Analysis** to drill through to the Top N Failure Analysis Report. For more information, see “TopN Failure Analysis Report” on page 227.

Note: It is possible that the location of a resource in the Maintenance Health and Failure Detail Report is different from the location of the same resource in the Top N Failure Analysis Report. If this happens, the **Location** field in the TopN Failure Analysis Report will be empty and you must select a location from the list and then run the report.

Statistical process control reports

The statistical process control (SPC) reports monitor the stability of a process. The charts in the reports show the data points in relation to the mean value, and upper and lower control limits.

SPC - Histogram

This bar chart is an overview of the frequency of an event or observation across a set of ranges or bins. The y axis shows the frequency. The x axis shows the bins. The height of the bar in a bin indicates the frequency of the event that falls into the range.

You can set the following prompt filters in this chart:

- From Date
- To Date
- Location
- Resource
- Event Type
- Measurement Type
- Event Code
- Number of Bins: Select **Number of Bins** to set the number of bins to display in the chart. The value that you select from the **User Selected Value** list is the number of bins that appears on the x axis.
- Bin Interval: Select **Bin Interval** to set the range for each bin. Enter the range in the **User Selected Value** field.
- Minimum: The minimum value for the bin range limit. Use this filter to set the lowest data point to be included in the data set.
- Maximum: The maximum value for the bin range limit. Use this filter to set the highest data point to be included in the data set.

The following measures are reported in the SPC Histogram chart.

Table 58. Measures in the SPC Histogram chart

Measure	Description
Frequency	Number of events that fall into a bin. The height of the bar indicates the frequency. Displayed on the y axis.
Bin Range	The bin interval. Displayed on the bins on the x axis.
Frequency of Bin Containing Mean Value	Frequency of the bin that contains the mean value of the events in the chart.
Count of Observations	Total number of events in the chart.
Mean	Mean value of the data in the chart.
Median	Median value of the data in the chart.
Minimum	Minimum value of the data in the chart.
Maximum	Maximum value of the data in the chart.
Range	The difference between the minimum and maximum values.
Standard Deviation	The standard deviation of the data in the chart.
Skewness	Indicates how symmetrical or asymmetrical the data is.
Kurtosis	Indicates if the data is peaked or flat, relative to a normal situation.

Table 58. Measures in the SPC Histogram chart (continued)

Measure	Description
Start Date	The date of the earliest event in the chart.
End Date	The date of the latest event in the chart.

The **Fitted Distribution** line shows the trend in the data.

Click **X Bar R/S Chart** to run the SPC - X Bar R/S Chart.

SPC - X Bar R/S Chart

The SPC - X Bar R/S chart shows variations in a process. You can use this chart to evaluate the stability of a process over a set of day ranges.

The SPC - X Bar chart shows how the average process changes over time. The median control limit is indicated by a dotted line. The solid lines on the chart indicate the upper and lower control limits. Data points that occur outside of the control limits indicate that the process is unstable.

The SPC - R/S chart shows how the average within a sub group changes over time. The SPC - R (range) chart is displayed when you enter a sub group value of 10 or less. The SPC - S (standard deviation) chart is displayed when you enter a sub group value that is greater than 10. The sub group size prompt controls the ranges that display on the x axis of both charts. For example, if you set the sub group prompt to 11 and the charts contain data from Jan 1 to Mar 9 (68 days), the x axis displays six ranges in 11-day increments. The seventh range contains a 2-day increment. The y axis on both charts indicates the control limit value.

The following prompts apply to this chart:

- From date
- To date
- Location
- Resource sub type
- Resource
- Measurement type
- Event code
- Resource code
- Profile variable type
- Sub group

Advanced KPI Trend Chart

This chart compares multiple key performance indicators (KPIs) across multiple resources. You can use this chart to analyze variations in a resource against a set of profiles. The main chart shows monthly data and you can drill down to a daily chart.

You can set the following prompt filters in this chart:

- From Date
- To Date
- Location

- Resource Sub Type
- Resource
- Profiles
- Event Code

Each chart displays data for one profile and all resources that you select in the prompt lists. By default, the chart displays all resources and all profiles but for clarity, select a few related profiles to analyze across a set of resources. Each data point in the chart represents one month of data for the profile. Click a data point or on the month on the x axis to see one month of data by day.

The following measures are reported in this chart.

Table 59. Measures in the Advanced KPI Trend Chart

Measure	Description
Actual Value	The value of the profile or measure for the resource for the month. Shown on the y axis.
Date	The year and month. Shown on the x axis. The month does not display if there is no data for the month.

QEWS quality dashboards

The QEWS quality dashboards show data for the QEWS quality inspection, warranty, and parametric use cases.

The quality dashboards provide an instant overview of parts and products with the help of stoplights. The following four factors are used to calculate stoplights:

- Return Code
- Severity
- Supplemental Alarms (also known as supplemental criteria)
- Business needs

The dashboards have a main report that shows the number of parts in each traffic light rule category, and a column that shows the overall distribution with respect to the product hierarchy level for a selected run date. The dashboards are linked to a detail history report, with a drill through link to show details of various threshold values with respect to each product of the drill through product hierarchy level. From the detail history report, a drill through link is provided to trigger the QEWS - Inspection chart report, the QEWSL - Warranty chart report, and the QEWSV - Parametric chart report.

Quality dashboard - Inspection

The Quality dashboard - Inspection report provides an overview of the state of products at a selected run date.

Prompt details

You can set the following prompt filter in this report:

- Run Date

Drill behavior

Drill up and drill down is enabled on the Product column. There is a drill through link from the Product column to the Quality Dashboard – Inspection Detail History report.

Quality dashboard - Inspection Detail History

The Quality dashboard - Inspection Detail History report provides details about the state of products and the various threshold values for a selected product category at a selected run date.

Prompt details

You can set the **Run Date** prompt filter in this report.

Drill behavior

There is a drill through link from the Product Code column to the QEWS – Inspection Chart report. The report is triggered from the Quality Dashboard - Parametric report.

Conditional formatting

Conditional formatting is applied to conditionally render the stoplight images for Distribution.

QEWS - Inspection Chart

The quality early warning system inspection chart reports the failure rates and values of evidence that the underlying failure rate process is unacceptably high.

You can report on a specific product type or a group of products. The analysis is based on data for a specified time period.

The chart shows performance of parts by vintage, where vintage is the day that the part was shipped. However, the analysis can be done for other vintages, such as the day of part manufacturing, or the day of part testing.

This chart is generated by IBM Predictive Maintenance and Quality daily. If the daily chart was not generated for the date that you select, the report is empty.

You can set the following prompt filters in this chart:

- Product Type
- Product Code
- Run Date

The chart heading contains the following information:

- Product code
- Last run date of the chart
- Period during which the product was shipped (start date and end date)
- Number of parts that were shipped over the period
- Number of parts that failed over the period
- Failure rate per 100 units over the period

Note: This chart is not an IBM Cognos Report Studio report so you cannot modify it in Report Studio.

Failure rate chart

This chart has a dual x axis that shows vintage number and Cumulative N_Tested. Vintage number is the day number that the part was shipped during the period. Cumulative N_Tested is the number of parts that were tested. The y axis shows the failure rate of the product per 100 units. A data point in the chart indicates the failure rate for a vintage number. The Acceptable Level is a horizontal line that indicates the acceptable failure rate.

Evidence chart

This chart has a dual x axis that shows vintage number and Cumulative N_Tested. Vintage number is the day number that the part was shipped during the period. Cumulative N_Tested is the number of parts that were tested. The y-axis shows the level of evidence that the underlying process failure rate is unacceptable, which is computed by using a weighted cumulative sum (CUSUM) formula.

The H Value is a horizontal line on the chart that shows the failure rate threshold value. The CUSUM values that are higher than H Value are displayed as triangles on the chart. The triangles indicate unacceptable process levels in the data. The vertical dotted line indicates the last time that the vintage number had an unacceptable failure rate. The forgiveness marker is the point in time when the process accumulated enough statistical evidence to suggest that its underlying failure rate was acceptable.

Summary list

The summary list header contains the same information as the chart heading. The summary list shows detailed information by vintage. It includes date, failure rate, total quantity that failed, and other data.

Quality dashboard - Warranty

The Quality dashboard - Warranty report provides an overview of the state of products at a selected run date.

Prompt details

You can set the following prompt filters in this report:

- Run Date
- Analysis Date

This prompt determines whether the Warranty start date is considered as the Date Sold, Date Manufactured, or Date Raw Materials Produced.

Drill behavior

Drill up and drill down is enabled on the Product column. There is a drill through link from the Product column to the Quality Dashboard – Warranty Detail History report.

Quality dashboard - Warranty Detail History

The Quality dashboard - Warranty Detail History report provides details about the state of products and the various threshold values for a selected product category at a selected run date.

Prompt details

You can set the following prompt filters in this report:

- Run Date
- Analysis Date

This prompt determines whether the Warranty start date is considered as the Date Sold, Date Manufactured, or Date Raw Materials Produced.

Drill behavior

There is a drill through link from the Product Code column to the QEWSL – Warranty Chart report.

Conditional formatting

Conditional formatting is applied to conditionally render the stoplight images for Replacement and for Wear out.

QEWSL - Warranty Chart

The quality early warning system life time (QEWSL) warranty chart reports the replacement rates for a specific product type and product code over a time period.

This chart is generated by IBM Predictive Maintenance and Quality daily. If the daily chart was not generated for the date that you select, the report is empty.

You can set the following prompt filters in this chart:

- Run Date
- Product Type
- Product Code

The chart heading contains the following information:

- Product code
- Last run date of the chart
- Period during which the product was shipped (start date and end date)
- Number of parts that were shipped over the period
- Number of parts that failed over the period
- Replacements per machine month of service over the period

Note: This chart is not an IBM Cognos Report Studio report so you cannot modify it in Report Studio.

Replacement rate chart

This chart has a dual x axis that shows vintage number and cumulative number of machine months of service. Vintage number is the day number that the part was shipped during the period. Cumulative number of machine months is the total number of machine months of service that is accrued by the population of

machines that have the parts installed. The y axis shows the replacement rate of the product per machine month. A data point on the chart indicates the replacement rate for a vintage. The Acceptable Level is a horizontal line on the chart that shows the acceptable replacement rate.

If the severity of wear out conditions is greater than zero, the chart contains a curve corresponding to monitoring wear out conditions. The levels of wear out index that is based on vintages summarized monthly corresponds to the y-axis on the chart.

Evidence chart

This chart monitors the reliability or characteristics of the lifetime of a part. The chart has a dual x axis that shows vintage number and cumulative number of machine months of service. Vintage number is the day number that the part was shipped as part of a machine. Cumulative machine months is the number of machine months of service. Cumulative machine month is shown on the x-axis. The y-axis shows the level of evidence that the underlying process replacement rate is unacceptable. It is computed by using a weighted cumulative sum (CUSUM) formula.

Threshold H is a horizontal line that shows the replacement rate threshold value. The CUSUM values that are higher than Threshold H are displayed as triangles on the chart. The triangles indicate unacceptable process levels in the data. The vertical dotted line indicates the last time that the vintage number had an unacceptable replacement rate per machine month.

If the severity of wear out conditions is greater than zero, the chart contains a curve corresponding to monitoring wear out conditions. The wear out curve is shown together with the corresponding threshold.

Summary list

The summary list header contains the same information as the chart heading. The summary list shows detailed information by vintage number. It includes date, number of parts that were tested, total quantity, and other data.

Quality dashboard - Parametric

The Quality dashboard - Parametric report provides an overview of the state of products at a selected run date for a variable.

Prompt details

You can set the following prompt filters in this report:

- Run Date
- Variable

Drill behavior

You can drill up and drill down on the Product column. There is a drill through link from the Product column to the Quality Dashboard – Parametric Detail History report.

Quality dashboard - Parametric Detail History

The Quality dashboard - Parametric Detail History report provides details about the state of products and the various threshold values for a selected product category, at a selected run date for a variable.

Prompt details

You can set the following prompt filters in this report:

- Run Date
- Variable

Drill behavior

There is a drill through link from the Product Code column to the QEWSV – Parametric Chart report.

Conditional formatting

Conditional formatting is applied to conditionally render the stoplight images for the Early Warning System (EWS) stoplights.

QEWSV - Parametric chart

The QEWSV - Parametric chart report is used for monitoring variable-type data and CUSUM values that are obtained from the QEWSV batch along with threshold levels.

The report is designed to support five different validation types: Material Validation, Process-Resource Validation, Production Batch Validation, Resource Health Check, and Location Suitability.

Prompt details

You can set the following prompt filters in this report:

- Run Date
- Validation Type
 - Production Batch is the supported default sub use case.
- Material
 - Depending on the Validation type prompt, this prompt is shown or hidden. This prompt is cascaded from Validation Type.
 - When Validation Type is MVARIABLE, this prompt is shown. Otherwise, it is hidden.
- Process
 - Depending on the Validation type prompt, this prompt is shown or hidden. This prompt is cascaded from Validation Type.
 - When Validation Type is PRVARIABLE, this prompt is shown. Otherwise, it is hidden.
- Resource
 - Depending on the Validation type prompt, this prompt is shown or hidden. This prompt is cascaded from Validation Type.
 - When Validation Type is PRVARIABLE or RVARIABLE, this prompt is shown. Otherwise, it is hidden.

- Location
Depending on the Validation type prompt, this prompt is shown or hidden. This prompt is cascaded from Validation Type.
When Validation Type is LVARIABLE, this prompt is shown. Otherwise, it is hidden.
- Product Type
Depending on the Validation type prompt, this prompt is shown or hidden. This prompt is cascaded from Validation Type.
When Validation Type is PBVARIABLE, this prompt is shown. Otherwise, it is hidden.
- Product Code
Depending on the Validation type prompt, this prompt is shown or hidden. This prompt is cascaded from Validation Type.
When Validation Type is PBVARIABLE, this prompt is shown. Otherwise, it is hidden.
- Variable Type
This prompt represents the measurement type.

Drill behavior

The Material, Location, Process, Resource, Product Type, and Product Code prompts are conditionally displayed based on the Validation Type prompt selection.

TopN Failure Analysis Report

This report shows the profiles that contribute to the failure of a resource. Each profile has an importance value that is expressed as a percentage. The total of the importance values displayed on the report is 100%.

The profile is indicated by the x axis. The importance value is indicated on the y axis. Each profile is represented by a bar on the chart. The higher the importance value, the more that the profile contributes to the failure of the resource.

The curved line on the chart indicates the cumulative importance value.

You can set the following prompt filters in this chart:

- Location
- Resource Sub Type
- Resource
- Resource Code
- Event Code

You can also access this report from the Maintenance Health and Failure Detail Report. For more information, see “Maintenance Overview Dashboard” on page 216.

Drill through to the statistical process control reports

Select a profile from the **Analyze Profile Variable** list. Click a link to one of the statistical process control (SPC) reports.

Note: The raw measurement type for the profile is passed to the SPC report.

Chapter 11. Implement a data lake by using the Hadoop Distributed File System

A *data lake* is a large-scale data storage repository and processing engine. You can upload large amounts of raw data into a data lake, without any transformation, and use the data in IBM Predictive Maintenance and Quality for further analysis.

There are two ways to upload raw data into a data lake that is implemented with the Hadoop Distributed File System (HDFS), retrieve the data from the data lake, do profiling work for analytics usage, and save the transformed data to analytics store.

The following table describes the methods:

Table 60. Methods for implementing a data lake in HDFS

Method	Advantages
Use IBM SPSS Modeler and IBM SPSS Analytic Server to upload and get data to and from HDFS.	<ul style="list-style-type: none">• Easier for development.• Requires fewer coding skills.
Use the HDFS command line to load data into HDFS, and then use Spark to load data and do profiling.	<ul style="list-style-type: none">• Is the method that is native to IBM Open Platform.• Offers better performance.

Use IBM SPSS Modeler and IBM Analytics Server

Setting up a connection between IBM SPSS Modeler and IBM Analytics Server

Procedure

1. Log in to the Ambari console.
2. Hover the mouse pointer over SPSS Analytic Server and ensure that Analytic Metastore and Analytic Server are started.
3. Open the *SPSS_installation_location*\config\options.cf file and edit the following parameters:

as_host

The host address of your SPSS Analytics Server instance. For example, 9.112.229.106.

as_port

The port number of your SPSS Analytics Server instance. The default port number is 9080.

as_prompt_for_password

Set the value to Y so that SPSS Analytics Server prompts for a user name and password before it makes a connection.

4. Start SPSS Modeler Client.
5. Click the **Sources** tab and drag an **Analytic Server** node to the canvas.
6. Double-click the **Analytic Server** node.

7. In the **Analytic Server** window, click **Launch Datasource Editor**. If the IBM SPSS Analytic Server data source editor appears in a new browser tab, then the connection is set up correctly.

Creating input and output folders in HDFS

Procedure

1. Use SSH to log in to the Hadoop cluster management node and change to HDFS user:

```
[root@iopmgmt1 ~]# su hdfs
[hdfs@iopmgmt1 root]$
```

2. Create input and output data folders:

```
[hdfs@iopmgmt1 root]$ hdfs dfs -mkdir /inputdata
[hdfs@iopmgmt1 root]$ hdfs dfs -mkdir /outputdata
```

3. Change the permission of the following folders:

```
[hdfs@iopmgmt1 root]$ hdfs dfs -chmod -R 777 /inputdata
[hdfs@iopmgmt1 root]$ hdfs dfs -chmod -R 777 /outputdata
```

4. Check the result:

```
[hdfs@iopmgmt1 root]$ hdfs dfs -ls /
Found 9 items
drwxrwxrwx - yarn   hadoop      0 2016-03-30 15:34 /app-logs
drwxr-xr-x - hdfs   hdfs       0 2016-03-30 15:33 /apps
drwxrwxrwx - hdfs   hdfs       0 2016-04-18 15:53 /inputdata
drwxr-xr-x - hdfs   hdfs       0 2016-03-30 15:31 /iop
drwxr-xr-x - mapred hdfs      0 2016-03-30 15:31 /mapred
drwxr-xr-x - hdfs   hdfs       0 2016-03-30 15:31 /mr-history
drwxrwxrwx - hdfs   hdfs       0 2016-04-18 15:53 /outputdata
drwxrwxrwx - hdfs   hdfs       0 2016-03-30 15:37 /tmp
drwxr-xr-x - hdfs   hdfs       0 2016-04-15 15:13 /user
```

Creating an SPSS Modeler stream to load data into the data lake

Procedure

1. In SPSS Modeler client, create a new stream.
2. Click the **Sources** tab and drag **Var. File** to the canvas.
3. Double-click the **Var. File** node.
4. In the **File** tab of window that appears, browse to the machine event file on your computer.
5. Clear the **Read field names from file** check box, and click **OK**.
6. Click the **Export** tab and drag an Analytic Server node to the canvas.
7. Right-click the **Var. File** node and click **Connect** to connect it to **Analytic Server** node.
8. Double-click the **Analytic Server** node.
9. In the **Analytic Server** window, click **Launch Datasource Editor**. IBM SPSS Analytic Server data source editor appears in a new browser tab.
10. Click **New**.
11. Type a data source name (for example, *inputdata*), select the **Files** content type, and click **OK**.
12. In the IBM SPSS Analytic Server data source editor, under **Output**, select **Make writable**.
13. Click **Browse** to select an Output folder.

14. Under **Select an output folder**, in the **Mode** box, select *File System*. Select the *inputdata* folder, and click **OK**.
15. Under **Output**, in the **File format** box, select the file format (for example, CSV).
16. In SPSS Modeler client, select the newly created *inputdata* data source, and click **OK**.
17. Save the stream and click **Run**.

Checking the result of the data load

Procedure

1. Use SSH to log in to the Hadoop cluster management node and change to HDFS user:

```
[root@iopmgmt1 ~]# su hdfs
[hdfs@iopmgmt1 root]$
```

2. The *inputdata* folder should contain a new comma-separated values file (.csv) with the correct file size:

```
[hdfs@iopmgmt1 root]$ hdfs dfs -ls /inputdata
Found 2 items
```

```
-rw-r--r--  3 as_user hdfs  43533541 2016-04-19 11:22 /inputdata/data_1542c89afcb-e3bba1a3af
-rw-r--r--  3 as_user hdfs      3685 2016-04-19 11:21 /inputdata/data_dm0.xml
```

Configuring data sources in SPSS Analytics Server

Procedure

1. In the SPSS Modeler client, click **File > New Stream**.
2. Click the **Sources** tab and drag an **Analytic Server** node to the canvas.
3. Double-click the **Analytic Server** node.
4. In the **Analytic Server** window, click **Launch Datasource Editor**.
5. In the IBM SPSS Analytic Server data source editor, click the existing *inputdata* data source.
6. Under **File Input**, in the **Mode** box, select **File System**.
7. Click the *inputdata* folder.
8. Select the comma-separated values file (.csv) and click the double arrow button to add the file to the data source definition.
9. In the **Settings** tab, set **First row contains field names** to **Yes**, and click **OK**.
After processing, you see a new file in the data source definition pane.
10. In SPSS Modeler, double-click the **Analytic Server** node.
11. In the **Analytic Server** window, click **Select**. Select the *inputdata* data source, and click **OK**.
12. Click **Refresh** to confirm that all fields are updated.
13. Click **Preview**.
You should see a table of event data, which means that SPSS Modeler successfully retrieved data from HDFS.
14. You must create another data source to save the profiling result. In the IBM SPSS Analytic Server data source editor, click **Data Sources**.
15. Click **New**.
16. In the **New data source** window, type *outputdata* as the data source name and in the **Content type** box, select **Files**. Click **OK**.
17. Under **Output**, select **Make writable**.

18. Click **Browse** to select an Output folder.
19. Under **Select an output folder**, in the **Mode** box, select *File System*. Select the *outputdata* folder, and click **OK**.
20. Under **Output**, in the **File format** box, select the file format (for example, *CSV*).

Creating an SPSS Modeler stream to profile data

Procedure

1. In SPSS Modeler client, return to the stream that was created earlier.
2. Click the **Record Ops** tab and drag **Aggregate** to the canvas.
3. Right-click the **inputdata** node and click **Connect** to connect it to **Aggregate** node.
4. Double-click the **Aggregate** node.
5. In the **Aggregate** window, in the **Settings** tab, click **Pick**.
6. Select the desired fields, and click **OK**. Select *Temperature* for this example.
7. In the **Settings** tab, choose the **Default mode**. Clear **Sum** and select **Mean**, **Min** (minimum), **Max** (maximum), and **SDev** (standard deviation). Click **Apply**.
8. Click **Preview**. The aggregate result is displayed in a table.
9. Click **OK** twice.
10. In the canvas, select **File > Save**.
11. Click the **Export** tab and drag an Analytic Server node to the canvas.
12. Right-click the **Aggregate** node and click **Connect** to connect it to the **Analytic Server** node.
13. Double-click the **Analytic Server** node.
14. In the **Analytic Server** window, click **Select** to select a data source.
15. Select *outputdata*.
16. Click **OK** twice.
17. In the canvas, select **File > Save**.
18. Click **Run**.

Checking the result of the data profiling

Procedure

1. Use SSH to log in to the Hadoop cluster management node and change to HDFS user:

```
[root@iopmgmt1 ~]# su hdfs
[hdfs@iopmgmt1 root]$
```

2. The *outputdata* folder should contain new files:

```
[hdfs@iopmgmt1 ~]$ hdfs dfs -ls /outputdata
Found 2 items
drwxr-xr-x   - as_user hdfs          0 2016-04-19 15:03 /outputdata/data_1542d5492a9-f93eed2bf003
-rw-r--r--   3 as_user hdfs          509 2016-04-19 15:03 /outputdata/data_dm0.xml
```

3. Check the file in the *.csv folder. Use the **cat** command to view the aggregate result:

```
[hdfs@iopmgmt1 ~]$ hdfs dfs -ls /outputdata/data_1542d5492a9-f93eed2bf0030482_0.csv
Found 1 items
-rw-r--r--   3 as_user hdfs          65 2016-04-19 15:03 /outputdata/data_1542d5492a9-f93eed2bf003
[hdfs@iopmgmt1 ~]$ hdfs dfs -cat /outputdata/data_1542d5492a9-f93eed2bf0030482_0.csv/part-r-125.9
```

Use the HDFS command line and Apache Spark

Uploading raw data by using the HDFS command line

You can use the HDFS command line to load data into HDFS. Because this method is native to IBM Open Platform, it offers better performance than using IBM SPSS Modeler and IBM Analytics Server.

About this task

This task assumes that you already uploaded a machine event data file called 1M_MachineEvent_a1 to the remote server.

Procedure

1. Use the following command to load data into HDFS system:

```
[root@iopmgmt1 rawdata]# hdfs dfs -put /rawdata/1M_MachineEvent_a1 /inputdata/
```

2. Use the following command to check the result:

```
[root@iopmgmt1 rawdata]# hdfs dfs -ls /inputdata
```

```
Found 3 items
```

```
-rw-r--r--  3 hdfs    hdfs    43170721 2016-04-19 11:25 /inputdata/1M_MachineEvent_a1
-rw-r--r--  3 as_user hdfs    43533541 2016-04-19 11:22 /inputdata/data_1542c89afcb-e3bba1a3af
-rw-r--r--  3 as_user hdfs      3685 2016-04-19 11:21 /inputdata/data_dm0.xml
```

Use Apache Spark for data profiling

You can choose Java, Scala, or Python to compose an Apache Spark application. Scala is an Eclipse-based development tool that you can use to create Scala object, write Scala code, and package a project as a Spark application.

The following example calculates aggregate values from the MachineEvent data input:

```
package datalake.spark
```

```
import org.apache.spark._
import org.apache.spark.SparkContext._
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql._
import org.apache.hadoop.conf.Configuration
import org.apache.hadoop.fs._

object EventProcessing {
  def main(args: Array[String]) {

    //for scala test
    //System.setProperty("hadoop.home.dir", "D:/hadoop-common-2.2.0-bin-master")
    //System.setProperty("spark.master","local")

    //create new spark config object and set application name
    val sparkConf = new SparkConf()
    sparkConf.setAppName("EventProcessing")

    //create new spark context and sql context
    val sc = new SparkContext(sparkConf)
    val sqlContext = new SQLContext(sc)

    //spark-csv is a tool that can load a csv file directly to a data frame
    //you can check hdfs port from ambary console
    val df = sqlContext.read
      .format("com.databricks.spark.csv")
      .option("header", "true") // Use first line of all files as header
```

```

        .option("inferSchema", "true") // Automatically infer data types
        .load("hdfs://9.112.229.106:8020/inputdata/1M_MachineEvent_a1")

//select columns from a data frame
var selectedddf = df.select("IncomingEventCode", "EventTime", "Temperature")

//describe() is the spark native functions to calculate aggregate values
//here we will show the result in console, including mean, min, max, stdDev
selectedddf.describe().show()

//write aggregate value to a csv
selectedddf.describe().write
    .format("com.databricks.spark.csv")
    .option("header", "true")
    .save("hdfs://9.112.229.106:8020/outputdata/temp.csv")

//merge csv part files to a single csv file
merge("hdfs://9.112.229.106:8020/outputdata/temp.csv", "hdfs://9.112.229.106:8020/outputdata/ever
}

//function to merge distributed part files to a single csv file
def merge(srcPath: String, dstPath: String): Unit = {
    val hadoopConfig = new Configuration()
    val hdfs = FileSystem.get(hadoopConfig)
    FileUtil.copyMerge(hdfs, new Path(srcPath), hdfs, new Path(dstPath), false, hadoopConfig, n
}
}

```

Submitting an Apache Spark application

You package a project as a Spark application and then you submit the application.

Procedure

1. In Scala IDE, in the **Package Explorer** tab, right-click the package and click **Export**.
2. Save the package as a JAR file. In the **Export** window, click **Java**, and click **JAR file**.
3. Use the following commands to upload the application and dependency JAR files to IBM Open Platform management nodes:

```

[root@iopmgmt1 /]# cd /datalake
[root@iopmgmt1 datalake]# ls
commons-csv-1.2.jar datalake.jar spark-csv_2.10-1.3.0.jar

```

4. Use the following command to grant permission to hdfs user:

```

[root@iopmgmt1 datalake]# chmod 777 -R /datalake/

```

5. Use the following command to switch to hdfs user:

```

[root@iopmgmt1 datalake]# su hdfs
[hdfs@iopmgmt1 datalake]$

```

6. Use the following command to submit the Spark application:

```

[hdfs@iopmgmt1 root]$ spark-submit --master yarn-client
--jars /datalake/spark-csv_2.10-1.3.0.jar,/datalake/commons-csv-1.2.jar
--class datalake.spark.EventProcessing /datalake/datalake.jar

```

7. View the result from the Spark console:

summary	IncomingEventCode	Temperature
count	362821	362821
mean	9420.645629663111	25.91079134013243
stddev	6138.230667617183	7.416425065531075
min	1	-2.22728
max	24150	101.355341

Figure 94. Spark console result

- Check the output folder to see the new files that were created there. Use the `cat` command to preview the result:

```
[hdfs@iopmgmt1 root]$ hdfs dfs -ls /outputdata/
Found 2 items
-rw-r--r--  3 hdfs  hdfs      216 2016-04-21 14:56 /outputdata/eventaggregation.csv
drwxr-xr-x  - hdfs  hdfs         0 2016-04-21 14:56 /outputdata/temp.csv
[hdfs@iopmgmt1 root]$ hdfs dfs -ls /outputdata/temp.csv
Found 3 items
-rw-r--r--  3 hdfs hdfs         0 2016-04-21 14:56 /outputdata/temp.csv/_SUCCESS
-rw-r--r--  3 hdfs hdfs        99 2016-04-21 14:56 /outputdata/temp.csv/part-00000
-rw-r--r--  3 hdfs hdfs       117 2016-04-21 14:56 /outputdata/temp.csv/part-00001
[hdfs@iopmgmt1 root]$ hdfs dfs -cat /outputdata/eventaggregation.csv
summary,IncomingEventCode,Temperature
count,362821,362821
mean,9420.645629663111,25.91079134013243
summary,IncomingEventCode,Temperature
stddev,6138.230667617183,7.416425065531075
min,1,-2.22728
max,24150,101.355341
```

Chapter 12. Implement a data lake by using IBM DB2

A *data lake* is a large-scale data storage repository and processing engine. You can upload large amounts of raw data into a data lake, without any transformation, and use the data in IBM Predictive Maintenance and Quality for further analysis.

The most efficient way to upload raw data into a data lake that is implemented with an IBM DB2 database is to use IBM SPSS Modeler and IBM SPSS Analytic Server.

Creating an SPSS Modeler stream to load data into the data lake

Before you begin

In IBM DB2, create a staging database. The staging database can exist on any convenient server. Give the staging database a recognizable name, such as PMQ_STAGING. In the staging database, create a MACHINE_EVENT table to load the raw events data. The table structure must match the machine event file. For example, a machine event file might contain the following fields: IncomingEventCode, SourceSystem, ResourceCd1, ResourceCd2, ResourceLocation, EventTime, ValueType, Temperature, AmbientTemperature, Load, Vibration, Exception, Overload, CumulativeLoad, CumulativeOverload, TemperatureRatio, CumulativeStoppage.

Create an ODBC entry that points to the staging database server.

Procedure

1. In SPSS Modeler client, create a new stream.
2. Click the **Sources** tab and drag **Var. File** to the canvas.
3. Double-click the **Var. File** node.
4. In the **File** tab of window that appears, browse to the machine event file on your computer. Click **OK**.
5. Click the **Export** tab and drag a **Database** node to the canvas.
6. Right-click the **Var. File** node and click **Connect** to connect it to **Database** node.
7. Right-click the **Database** node. In the **Data source** box, select the staging table. For example, PMQ_STAGING.
8. Click **Select** and choose the table name. For example, PMQ_STAGING.MACHINE_EVENT.
9. Click **OK**.
10. Save the stream and click **Run**.

What to do next

Log in to the server on which the staging database exists. Check the MACHINE_EVENT table to confirm that the event data loaded correctly.

Appendix A. Accessibility features

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products.

For information about the commitment that IBM has to accessibility, see the IBM Accessibility Center (www.ibm.com/able).

IBM Cognos HTML documentation has accessibility features. PDF documents are supplemental and, as such, include no added accessibility features.

Report output

In IBM Cognos Administration, you can enable system-wide settings to create accessible report output. For more information, see the *IBM Cognos Business Intelligence Administration and Security Guide*. In IBM Cognos Report Studio, you can enable settings to create accessible output for individual reports. For more information, see the *IBM Cognos Report Studio User Guide*. You can access the previously mentioned documents at IBM Knowledge Center (<http://www.ibm.com/support/knowledgecenter>).

Appendix B. The flat file API

Use the flat file application programming interface (API) to supply and modify IBM Predictive Maintenance and Quality master data.

The IBM Predictive Maintenance and Quality API supports the **upsert** operation.

The **upsert** operation attempts to update an existing row. If the matching row cannot be found, a new row is created that uses the values in the input record.

All the values for the row must be included even if only a single value of the row is being changed.

An IS_ACTIVE indicator is used to mark records as no longer in use (IS_ACTIVE = 0).

The IS_ACTIVE indicator is not used to take any decisions while loading the master or event data. For example, while loading a resource, if the associated location has the following indicator: IS_ACTIVE=0, that resource is loaded and associated with that location. Similarly if the event is reported by the resource with IS_ACTIVE=0, the event is processed and stored in the datastore.

Master data in the API

Use master data to supply IBM Predictive Maintenance and Quality with information about the context in which events occur.

The following records are supported by the master data section of the application programming interface (API). They are listed in alphabetical order but functionally they fall into one of four logical groups:

- Resource-related records include the location, resource, and resource_type records
- Process-related records include the batch_batch, process, product, and production_batch records
- Material-related records include the material and material_type records
- Other records can be related to both devices and processes. These records include the group_dim, source_system, and supplier records

No Delete operation is supported for master data. The upsert API can be used to mark a master data row as no longer active. In this case, the item in the row is no longer used in reports.

Load order

Some tables include references to rows in other tables. A row must be loaded before it can be referenced from another table.

The language and tenant tables must be loaded before any other data is loaded. The language_cd and tenant_cd rows are referenced in many tables. The values that are supplied for the language_cd and tenant_cd rows must reference rows already present in the language and tenant tables.

In addition, the rows of some tables refer to other rows in the same table, for example, parent rows. The referenced rows must be added before the rows that reference them.

Master files must be loaded sequentially.

The following table lists tables that contain references to other tables.

Table 61. Tables that must exist before other tables can be loaded

Table	Prerequisite tables
batch_batch	production_batch
material	material_type, supplier
process	process (parent process) Note: No circular relationship is allowed. That is, a process_code cannot be a parent to itself.
production_batch	product
resource	group_dim, location, resource (parent resource)
profile_variable	measurement_type, material_type

batch_batch

Creates a many-to-many relationship between production batches.

Use the **batch_batch** for batch traceability so that batches that share materials can be enumerated when a defect is found at any point. Every batch must relate to every batch in its lineage for full traceability.

For example, batch 1 splits into 2 and 3, and batch 3 splits into 4 and 5.

batch_batch holds these pairs:

- 1,1
- 1,2
- 1,3
- 1,4
- 1,5
- 2,1
- 2,3
- 3,1
- 3,2
- 3,4
- 3,5
- 4,1
- 4,3
- 4,5
- 5,1
- 5,3
- 5,4

The fields in the **batch_batch** table are listed in the following table.

Table 62. Fields in the `batch_batch` table

Field	Type	Comments
<code>production_batch_cd</code>	<code>string(50)</code>	Required
<code>related_production_batch_cd</code>	<code>string(50)</code>	Required

batch_batch code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT PB1.PRODUCTION_BATCH_CD, PB2.PRODUCTION_BATCH_CD FROM
SYSREC.MASTER_BATCH_BATCH M JOIN SYSREC.MASTER_PRODUCTION_BATCH PB1 ON
M.PRODUCTION_BATCH_ID = PB1.PRODUCTION_BATCH_ID JOIN
SYSREC.MASTER_PRODUCTION_BATCH PB2 ON M.RELATED_PRODUCTION_BATCH_ID =
PB2.PRODUCTION_BATCH_ID;
```

event_code

Contains codes for alarms, failures, issues, and so on.

When an event arrives with a measurement type which has an event code indicator of 1, the text from the **event_observation_text** value is assumed to contain an event code. The measurement type of the event defines the **event_code_set** value.

The fields in the **event_code** table are listed in the following table.

Table 63. Fields in the `event_code` table

Field	Type	Comments
<code>event_code_set</code>	<code>string(50)</code>	Required
<code>event_code_set_name</code>	<code>string(200)</code>	Required
<code>event_code</code>	<code>string(50)</code>	Required
<code>language_cd</code>	<code>string(50)</code>	Optional. This value must reference a row in the language table.
<code>tenant_cd</code>	<code>string(50)</code>	Optional. This value must reference a row in the tenant table.

event_code code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.EVENT_CODE_SET, M.EVENT_CODE_SET NAME, M.EVENT_CODE, L.LANGUAGE_CD,
T.TENANT_CD FROM SYSREC.MASTER_EVENT_CODE M JOIN SYSREC.LANGUAGE L ON
M.LANGUAGE_ID = L.LANGUAGE_ID JOIN SYSREC.TENANT T ON M.TENANT_ID =
T.TENANT_ID;
```

group_dim

Provides classifications for resources.

Up to five classifications are possible for each resource. The classifications vary depending on how IBM Predictive Maintenance and Quality is used. For example, a classification may be manufacturer or organization.

The fields for the **group_dim** table are listed in the following table.

Table 64. Fields in the **group_dim** table

Field	Type	Comments
group_type_cd	string(50)	Required
group_type_name	string(200)	Required
group_member_cd	string(50)	Required
group_member_name	string(200)	Required
language_cd	string(50)	Optional. This value must reference a row in the language table.
tenant_cd	string(50)	Optional. This value must reference a row in the tenant table.

group_dim code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.GROUP_TYPE_CODE, M.GROUP_TYPE_TEXT, M.GROUP_MEMBER_CODE,
M.GROUP_MEMBER_TEXT, L.LANGUAGE_CD, T.TENANT_CD FROM SYSREC.MASTER_GROUP_DIM M
JOIN SYSREC.LANGUAGE L ON M.LANGUAGE_ID = L.LANGUAGE_ID
JOIN SYSREC.TENANT T ON M.TENANT_ID = T.TENANT_ID;
```

language

Contains the list of supported languages.

The fields in the **language** table are listed in the following table.

Table 65. Fields in the **language** table

Field	Type	Comments
language_cd	string(50)	Required. For example, EN
language_name	string(200)	Required. For example, English.
DEFAULT_IND	0 or 1	Optional. A value of 1 indicates that this language is the default language for the system. No value, or a value of 0, indicates the language is not the default.

Language code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line.

```
SELECT LANGUAGE_CD, LANGUAGE_NAME, DEFAULT_IND FROM SYSREC.LANGUAGE;
```

New languages and tenants

After you have added new languages, or new tenants, you must populate the NA rows in the database for all new valid combinations of language and tenant. See the following example.

```
db2 "call SCHEMA.POP_NA( 'LANGUAGE_CD' , 'LANGUAGE_NAME' , 'TENANT_CD' , 'TENANT_NAME' )" 
```

Where schema is a valid DB2 schema, such as db2inst1.

location

The location of a resource or event.

The location can be as specific, such as a room in a factory or general, such as a mine site.

The fields in the **location** table are listed in the following table.

Table 66. Fields in the **location** table

Field	Type	Comments
location_cd	string(50)	Required
location_name	string(200)	Required
region_cd	string(50)	Optional. The region_cd and region_name parameters must be supplied together.
region_name	string(200)	Optional
country_cd	string(50)	Optional. The country_cd and country_name parameters must be supplied together.
country_name	string(200)	Optional
state_province_cd	string(50)	Optional. The state_province_cd and state_province_name parameters must be supplied together.
state_province_name	string(200)	Optional
city_name	string(200)	Optional
latitude	decimal (in signed decimal degrees. N is + and S is -)	Optional
longitude	decimal (in signed decimal degrees. E is + and W is -)	Optional

Table 66. Fields in the **location** table (continued)

Field	Type	Comments
language_cd	string(50)	Optional. This value must reference a row in the language table.
tenant_cd	string(50)	Optional. This value must reference a row in the tenant table.
IS_ACTIVE	0 or 1	Optional. A value of 0 indicates that the record is inactive. No value, or a value of 1, indicates that the record is active.

Location code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.LOCATION_CD, M.LOCATION_NAME, M.REGION_CD, M.REGION_NAME, M.COUNTRY_CD,
M.COUNTRY_NAME, M.STATE_PROVINCE_CD, M.STATE_PROVINCE_NAME, M.CITY_NAME,
M.LATITUDE, M.LONGITUDE, L.LANGUAGE_CD, T.TENANT_CD, M.ISACTIVE FROM
SYSREC.MASTER_LOCATION M JOIN SYSREC.LANGUAGE L ON M.LANGUAGE_ID =
L.LANGUAGE_ID JOIN SYSREC.TENANT T ON M.TENANT_ID = T.TENANT_ID;
```

material

Defines the material that is used for an event.

The fields in the **material** table are defined as a specific instance of a material type, including a link to the supplier. It can be material that is used in a repair or material that is used in a production process.

The fields in the **material** table are listed in the following table.

Table 67. Fields in the **material** table

Field	Type	Comments
material_cd	string(50)	Required
material_name	string(200)	Required
material_type_cd	string(50)	Required
supplier_cd	string(50)	Required
language_cd	string(50)	Optional. This value must reference a row in the language table.
tenant_cd	string(50)	Optional. This value must reference a row in the tenant table.
IS_ACTIVE	0 or 1	Optional. A value of 0 indicates that the record is inactive. No value, or a value of 1, indicates that the record is active.

material code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.MATERIAL_CD, M.MATERIAL_NAME, MT.MATERIAL_TYPE_CD, S.SUPPLIER_CD,
L.LANGUAGE_CD, T.TENANT_CD, M.ISACTIVE FROM SYSREC.MASTER_MATERIAL M
JOIN SYSREC.LANGUAGE L ON M.LANGUAGE_ID = L.LANGUAGE_ID JOIN
SYSREC.TENANT T ON M.TENANT_ID = T.TENANT_ID JOIN
SYSREC.MASTER_MATERIAL_TYPE MT ON M.MATERIAL_TYPE_ID = MT.MATERIAL_TYPE_ID AND
M.LANGUAGE_ID = MT.LANGUAGE_ID JOIN SYSREC.MASTER_SUPPLIER S ON M.SUPPLIER_ID =
S.SUPPLIER_ID AND M.LANGUAGE_ID = S.LANGUAGE_ID;
```

material_type

A categorization of material by type.

Material type is material that is used in a repair, such as engine filters or parts, or it can be material that is used in a production process.

The fields in the **material type** table are listed in the following table.

Table 68. Fields in the material type table

Field	Type	Comments
material_type_cd	string(50)	Required
material_type_name	string(200)	Required
language_cd	string(50)	Optional. This value must reference a row in the language table.
tenant_cd	string(50)	Optional. This value must reference a row in the tenant table.

material_type code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.MATERIAL_TYPE_CD, M.MATERIAL_TYPE_NAME, L.LANGUAGE_CD, T.TENANT_CD FROM
SYSREC.MASTER_MATERIAL_TYPE M JOIN SYSREC.LANGUAGE L ON M.LANGUAGE_ID =
L.LANGUAGE_ID JOIN SYSREC.TENANT T ON M.TENANT_ID = T.TENANT_ID;
```

process

Represents a production process.

A process can be part of a hierarchy of processes.

The fields in the **process** table are listed in the following table.

Table 69. Fields in the process table

Field	Type	Comments
process_cd	string(50)	Required
process_name	string(200)	Required
parent_process_cd	string(50)	Optional
language_cd	string(50)	Optional. This value must reference a row in the language table.
tenant_cd	string(50)	Optional. This value must reference a row in the tenant table.

process code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.PROCESS_CD, M.PROCESS_NAME, P.PROCESS_CD AS PARENT_PROCESS_CD,
L.LANGUAGE_CD, T.TENANT_CD FROM SYSREC.MASTER_PROCESS M JOIN
SYSREC.LANGUAGE L ON M.LANGUAGE_ID = L.LANGUAGE_ID JOIN
SYSREC.TENANT T ON M.TENANT_ID = T.TENANT_ID JOIN SYSREC.MASTER_PROCESS
P ON M.PARENT_PROCESS_ID = P.PARENT_PROCESS_ID AND M.LANGUAGE_ID = P.LANGUAGE_ID;
```

product

Defines the product being produced by the events.

The fields in the **product** table are listed in the following table.

Table 70. Fields in the product table

Field	Type	Comments
product_cd	string(50)	Required
product_name	string(200)	Required
language_cd	string(50)	Optional. This value must reference a row in the language table.
tenant_cd	string(50)	Optional. This value must reference a row in the tenant table.
IS_ACTIVE	0 or 1	Optional. A value of 0 indicates that the record is inactive. No value, or a value of 1, indicates that the record is active.

product code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.PRODUCT_CD, M.PRODUCT_NAME, L.LANGUAGE_CD, T.TENANT_CD, M.ISACTIVE FROM
SYSREC.MASTER_PRODUCT M JOIN SYSREC.LANGUAGE L ON M.LANGUAGE_ID =
L.LANGUAGE_ID JOIN SYSREC.TENANT T ON M.TENANT_ID = T.TENANT_ID;
```

production_batch

Contains information about product groupings during the production event.

A batch can split and merge throughout the production process, and so one batch can be related to many other batches.

The fields in the **production_batch** table are listed in the following table.

Table 71. Fields in the production_batch table

Field	Type	Comments
production_batch_cd	string(50)	Required
production_batch_name	string(200)	Required
product_cd	string(50)	Required
product_type_cd	string(50)	Required
language_cd	string(50)	Optional. This value must reference a row in the language table.
tenant_cd	string(50)	Optional. This value must reference a row in the tenant table.

production_batch code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.PRODUCTION_BATCH_CD, M.PRODUCTION_BATCH_NAME, P.PRODUCT_CD,
L.LANGUAGE_CD, T.TENANT_CD FROM SYSREC.MASTER_PRODUCTION_BATCH M JOIN
SYSREC.LANGUAGE L ON M.LANGUAGE_ID = L.LANGUAGE_ID JOIN
SYSREC.TENANT T ON M.TENANT_ID = T.TENANT_ID JOIN SYSREC.MASTER_PRODUCT
P ON M.PRODUCT_ID = P.PRODUCT_ID AND M.LANGUAGE_ID = P.LANGUAGE_ID;
```

profile_calculation

These records define a set of profile calculation names.

Profile calculations aggregate event values into KPIs and Profiles.

The fields in the **profile_calculation** table are listed in the following table.

Table 72. Fields in the `profile_calculation` table

Field	Type	Comments
<code>profile_calculation_name</code>	string(200)	Required
<code>language_cd</code>	string(50)	Optional
<code>tenant_cd</code>	string(50)	Optional

profile_calculation code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.PROFILE_CALCULATION_NAME, T.TENANT_CD FROM
SYSREC.MASTER_PROFILE_CALCULATION M JOIN SYSREC.TENANT T ON M.TENANT_ID
= T.TENANT_ID;
```

resource

Defines resources of type asset or agent. Asset or agent are the only resource types allowed.

An asset is a piece of equipment. An agent is the operator of the equipment. Some asset resources can form a hierarchy. For example, a truck is a parent of a tire.

Parent resources must be loaded before child resources. Resources cannot be their own parent.

More specific types of resources can be named in the `resource_sub_type` column.

The fields in the **resource** table are listed in the following table.

Table 73. Fields in the `resource` table

Field	Type	Comments
<code>resource_cd1</code>	string(50)	Optional. Used to provide serial Number. Either <code>resource_cd1</code> and <code>resource_cd2</code> are required or <code>operator_cd</code> is required.
<code>resource_cd2</code>	string(50)	Optional. Used to provide model information of resource.
<code>operator_cd</code>	string(50)	Optional
<code>resource_name</code>	string(500)	Required
<code>resource_type_cd</code>	string(50)	Required
<code>resource_sub_type</code>	string(50)	Optional
<code>parent_resource_serial_no</code>	string(50)	Optional. The <code>parent_resource_serial_no</code> and <code>parent_resource_model</code> parameters must be supplied together.

Table 73. Fields in the resource table (continued)

Field	Type	Comments
parent_resource_model	string(50)	Optional
parent_resource_operator_cd	string(50)	Optional
standard_production_rate	decimal	Optional
production_rate_uom	string(40)	Optional
preventative_maintenance_interval	decimal	Optional
group_dim_type_cd_1	string(50)	Optional. The type and a member must be supplied together.
group_dim_member_cd_1	string(50)	Optional
group_dim_type_cd_2	string(50)	Optional
group_dim_member_cd_2	string(50)	Optional
group_dim_type_cd_3	string(50)	Optional
group_dim_member_cd_3	string(50)	Optional
group_dim_type_cd_4	string(50)	Optional
group_dim_member_cd_4	string(50)	Optional
group_dim_type_cd_5	string(50)	Optional
group_dim_member_cd_5	string(50)	Optional
location_cd	string(50)	Optional
language_cd	string(50)	Optional. This value must reference a row in the language table.
mfg_date	date	Optional. Manufacturing date of resource.
tenant_cd	string(50)	Optional. This value must reference a row in the tenant table.
IS_ACTIVE	0 or 1	Optional. A value of 0 indicates that the record is inactive. No value, or a value of 1, indicates that the record is active.

resource code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.RESOURCE_CD1, M.RESOURCE_CD2, M.RESOURCE_NAME,
       RT.RESOURCE_TYPE_CD,M.RESOURCE_SUB_TYPE, P.RESOURCE_CD1 AS
       PARENT_RESOURCE_CD1,P.RESOURCE_CD1 AS PARENT_RESOURCE_CD2,
       M.STANDARD_PRODUCTION_RATE, M.PRODUCTION_RATE_UOM,M.PREVENTIVE_MAINTENANCE_INTERVAL,
       G1.GROUP_TYPE_CD AS GROUP_TYPE_CD_1,G1.GROUP_MEMBER_CD AS GROUP_MEMBER_CD_1,
       G2.GROUP_TYPE_CD AS GROUP_TYPE_CD_2,G2.GROUP_MEMBER_CD AS GROUP_MEMBER_CD_2,
       G3.GROUP_TYPE_CD AS GROUP_TYPE_CD_3,G3.GROUP_MEMBER_CD AS GROUP_MEMBER_CD_3,
       G4.GROUP_TYPE_CD AS GROUP_TYPE_CD_4,G4.GROUP_MEMBER_CD AS GROUP_MEMBER_CD_4,
```

```
G5.GROUP_TYPE_CD AS GROUP_TYPE_CD_5,G5.GROUP_MEMBER_CD AS GROUP_MEMBER_CD_5,
LC.LOCATION_CD,M.MFG_DATE, L.LANGUAGE_CD,T.TENANT_CD, M.IS_ACTIVE FROM
SYSREC.MASTER_RESOURCE M JOIN SYSREC.LANGUAGE L ON M.LANGUAGE_ID = L.LANGUAGE_ID JOIN
SYSREC.TENANT T ON M.TENANT_ID =T.TENANT_ID LEFT OUTER JOIN SYSREC.MASTER_RESOURCE P ON
M.PARENT_RESOURCE_ID =P.MASTER_RESOURCE_ID AND M.LANGUAGE_ID = P.LANGUAGE_ID JOIN
SYSREC.MASTER_GROUP_DIM G1 ONM.GROUP_DIM_ID_1 = G1.MASTER_GROUP_DIM_ID AND
M.LANGUAGE_ID = G1.LANGUAGE_ID JOINSYSREC.MASTER_GROUP_DIM G2 ON M.GROUP_DIM_ID_2 =
G2.MASTER_GROUP_DIM_ID AND M.LANGUAGE_ID= G2.LANGUAGE_ID JOIN SYSREC.MASTER_GROUP_DIM G3
ON M.GROUP_DIM_ID_3 =G3.MASTER_GROUP_DIM_ID AND M.LANGUAGE_ID = G3.LANGUAGE_ID JOIN
SYSREC.MASTER_GROUP_DIM G4ON M.GROUP_DIM_ID_4 = G4.MASTER_GROUP_DIM_ID AND
M.LANGUAGE_ID = G4.LANGUAGE_ID JOINSYSREC.MASTER_GROUP_DIM G5 ON M.GROUP_DIM_ID_5 =
G5.MASTER_GROUP_DIM_ID AND M.LANGUAGE_ID= G5.LANGUAGE_ID JOIN SYSREC.MASTER_LOCATION LC
ON M.LOCATION_ID = LC.MASTER_LOCATION_ID AND M.LANGUAGE_ID = LC.LANGUAGE_ID JOIN
SYSREC.MASTER_RESOURCE_TYPE RT ONM.RESOURCE_TYPE_ID = RT.MASTER_RESOURCE_TYPE_ID AND
M.LANGUAGE_ID = RT.LANGUAGE_ID;
```

resource_type

These records categorize resources.

The two supported resource types are asset and agent. An asset is a piece of equipment that is used in the production process. An agent is the operator of the equipment.

The fields in the **resource_type** table are listed in the following table.

Table 74. Fields in the **resource_type** table

Field	Type	Comments
resource_type_cd	string(50)	Required
resource_type_name	string(200)	Required
language_cd	string(50)	Optional. This value must reference a row in the language table.
tenant_cd	string(50)	Optional. This value must reference a row in the tenant table.

resource_type code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.RESOURCE_TYPE_CD, M.RESOURCE_TYPE_NAME, L.LANGUAGE_CD, T.TENANT_CD FROM
SYSREC.MASTER_RESOURCE_TYPE M JOIN SYSREC.LANGUAGE L ON M.LANGUAGE_ID =
L.LANGUAGE_ID JOIN SYSREC.TENANT T ON M.TENANT_ID = T.TENANT_ID;
```

source_system

Contains information about the system generating an event.

The fields in the **source_system** table are listed in the following table.

Table 75. Fields in the `source_system` table

Field	Type	Comments
<code>source_system_cd</code>	string(50)	Required.
<code>source_system_name</code>	string(200)	Required.
<code>language_cd</code>	string(50)	Optional. This value must reference a row in the language table.
<code>tenant_cd</code>	string(50)	Optional. This value must reference a row in the tenant table.
<code>IS_ACTIVE</code>	0 or 1	Optional. A value of 0 indicates that the record is inactive. No value, or a value of 1, indicates that the record is active.

source_system code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.SOURCE_SYSTEM_CD, M.SOURCE_SYSTEM_NAME, L.LANGUAGE_CD, T.TENANT_CD,
M.ISACTIVE FROM SYSREC.MASTER_SOURCE_SYSTEM M JOIN SYSREC.LANGUAGE L ON
M.LANGUAGE_ID = L.LANGUAGE_ID JOIN SYSREC.TENANT T ON M.TENANT_ID =
T.TENANT_ID;
```

supplier

Contains material supplier information.

The fields in the **supplier** table are listed in the following table.

Table 76. Fields in the `supplier` table

Field	Type	Comments
<code>supplier_cd</code>	string(50)	Required.
<code>supplier_name</code>	string(200)	Required.
<code>language_cd</code>	string(50)	Optional. This value must reference a row in the language table.
<code>tenant_cd</code>	string(50)	Optional. This value must reference a row in the tenant table.
<code>IS_ACTIVE</code>	0 or 1	Optional. A value of 0 indicates that the record is inactive. No value, or a value of 1, indicates that the record is active.

supplier code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.SUPPLIER_CD, M.SUPPLIER_NAME, L.LANGUAGE_CD, T.TENANT_CD, M.ISACTIVE
FROM SYSREC.MASTER_SUPPLIER M JOIN SYSREC.LANGUAGE L ON M.LANGUAGE_ID =
L.LANGUAGE_ID JOIN SYSREC.TENANT T ON M.TENANT_ID = T.TENANT_ID;
```

tenant

Contain the list of supported tenants.

The fields in the **tenant** table are listed in the following table.

Table 77. Fields in the **tenant** table

Field	Type	Comments
tenant_cd	string(50)	Required.
tenant_name	string(200)	Required.
DEFAULT_IND	0 or 1	Optional. A value of 0 indicates that the record is inactive. No value, or a value of 1, indicates that the record is active.

tenant code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line.

```
SELECT TENANT_CD, TENANT_NAME, DEFAULT_IND FROM SYSREC.TENANT;
```

For information on adding new languages and tenants, see the following information: “New languages and tenants” on page 245.

Changing the tenant code and name

You can rename the tenant code and name. For example, in the sample data, the tenant code and name by default is PMQ.

Procedure

1. Type the following command to connect to the **IBMPMQ** database by connecting to the DB2 node:

```
db2 "connect to IBMPMQ user user_name using password"
```

2. Type the following command:

```
db2 "update sysrec.master_tenant set tenant_code='CODE',
tenant_name='NAME' where tenant_code='PMQ'"
```

Where *CODE* is the tenant code, and *NAME* is the tenant name.

For example, the following code renames the tenant code to XY, and the tenant name to XY Ltd.

```
db2 "update sysrec.master_tenant set tenant_code='XY',
tenant_name='XY Ltd' where tenant_code='PMQ'"
```

3. Type the following command to commit the transaction:
db2 "commit"
4. Type the following command to disconnect from the database:
db2 "connect reset"

value_type

Defines the set of possible numerical observations, including actual, planned or forecast.

The fields for the **value_type** table are listed in the following table.

Table 78. Fields for the **value_type** table

Field	Type	Comments
value_type_cd	string(50)	Required
value_type_name	string(200)	Required
language_cd	string(50)	Optional. This value must reference a row in the language table.
tenant_cd	string(50)	Optional. This value must reference a row in the tenant table.

value_type code snippet

You can use the following SQL Code snippet to retrieve master data in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load master data, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.VALUE_TYPE_CD, M.VALUE_TYPE_NAME, L.LANGUAGE_CD, T.TENANT_CD FROM
SYSREC.MASTER_VALUE_TYPE M JOIN SYSREC.LANGUAGE L ON M.LANGUAGE_ID =
L.LANGUAGE_ID JOIN SYSREC.MASTER_TENANT T ON M.TENANT_ID = T.TENANT_ID;
```

Metadata in the API

The following records are supported by the metadata section of the application programming interface (API). The records are listed in alphabetical order.

event_type

These records define a categorization of events.

Some examples of event types are measurement, alarm, and inspection.

The fields in the **event_type** table are listed in the following table.

Table 79. Fields in the **event_type** table

Field	Type	Comments
event_type_cd	string(50)	Required.
event_type_name	string(200)	Required.

Table 79. Fields in the `event_type` table (continued)

Field	Type	Comments
<code>language_cd</code>	string(50)	Optional. This value must reference a row in the language table.
<code>tenant_cd</code>	string(50)	Optional. This value must reference a row in the tenant table.

event_type code snippet

You can use the following SQL Code snippet to retrieve metadata in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load metadata, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.EVENT_TYPE_CD, M.EVENT_TYPE_NAME, L.LANGUAGE_CD, T.TENANT_CD FROM
SYSREC.MASTER_EVENT_TYPE M JOIN SYSREC.LANGUAGE L ON M.LANGUAGE_ID =
L.LANGUAGE_ID JOIN SYSREC.TENANT T ON M.TENANT_ID = T.TENANT_ID
```

measurement_type

Contains all the measures and event code sets that can be observed for the **resource**, **process**, and **material** records.

Some examples of measurement types are engine oil pressure, ambient temperature, fuel consumption, conveyor belt speed, capping pressure, and so on.

In the case of measurement types where the **event_code_indicator** value is 1, there is a special class to capture failure codes, issue codes, and alarm codes as **event_code** records. The **measurement_type_code** and **measurement_type_name** records become the **event_code_set** and **event_code_set_name** records respectively. This is a trigger to the event integration process to begin recording event codes from the **observation_text** record.

The fields for the **measurement_type** table are listed in the following table.

Table 80. Fields for the `measurement_type`

Field	Type	Comments
<code>measurement_type_cd</code>	string(50)	Required
<code>measurement_type_name</code>	string(200)	Required
<code>unit_of_measure</code>	string(100)	Optional
<code>carry_forward_indicator</code>	0 or 1	Optional
<code>aggregation_type</code>	string(100)	Optional
<code>event_code_indicator</code>	0 or 1	Optional
<code>language_cd</code>	string(50)	Optional. This value must reference a row in the language table.
<code>tenant_cd</code>	string(50)	Optional. This value must reference a row in the tenant table.

measurement_type code snippet

You can use the following SQL Code snippet to retrieve metadata in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load metadata, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.MEASUREMENT_TYPE_CD, M.MEASUREMENT_TYPE_NAME, M.UNIT_OF_MEASURE,  
M.CARRY_FORWARD_INDICATOR, M.AGGREGATION_TYPE, M.EVENT_CODE_INDICATOR,  
L.LANGUAGE_CD, T.TENANT_CD FROM SYSREC.MASTER_MEASUREMENT_TYPE M JOIN  
SYSREC.LANGUAGE L ON M.LANGUAGE_ID = L.LANGUAGE_ID JOIN  
SYSREC.TENANT T ON M.TENANT_ID = T.TENANT_ID;
```

profile_variable

These records relate the measurement_type, resource_type, and material_type values to profile calculations.

The fields in the **profile_variable** table are listed in the following table.

Table 81. Fields in the profile_variable table

Field	Type	Comments
profile_variable_cd	string(50)	Required
profile_variable_name	string(200)	Required
profile_calculation_name	string(200)	Required
measurement_type_cd	string(50)	Required
resource_type_cd	string(50)	Optional
material_type_cd	string(50)	Optional
profile_units	string(100)	Optional
comparison_string	string(200)	Optional
low_value_date	datetime	Optional
high_value_date	datetime	Optional
low_value_number	decimal	Optional
high_value_number	decimal	Optional
kpi_indicator	0 or 1	Optional. To disable a profile variable, set its kpi_indicator and profile_indicator to 0
profile_indicator	0 or 1	Optional. To disable a profile variable, set its kpi_indicator and profile_indicator to 0
data_type	string(100)	Optional
aggregation_type	string(100)	Optional
carry_forward_indicator	0 or 1	Optional
process_indicator	0 or 1	Optional
variance_multiplier	-1 or 1	Required. A value of 1 indicates that a higher measurement value is preferred. A value of -1 that a lower value is preferred.
tenant_cd	string(50)	Optional. This value must reference a row in the tenant table.

Due to references from the KPI and Profile tables, the upsert API for a profile_variable only allows the values of the following fields to be updated

- profile_units
- comparison_string
- low_value_date
- high_value_date
- low_value_number
- kpi_indicator
- profile_indicator
- data_type
- aggregation_type
- process_indicator
- profile_variable_name

profile_variable code snippet

You can use the following SQL Code snippet to retrieve metadata in the format that is required by the **upsert** API.

For example, if you lost the original files that are used to load metadata, you can use the snippet to retrieve the data, make changes, and submit the changes by using the **upsert** API.

The command must be on a single line, not as shown here.

```
SELECT M.PROFILE_VARIABLE_CD, M.PROFILE_VARIABLE_NAME, PC.PROFILE_CALCULATION_NAME,
MSRT.MEASUREMENT_TYPE_CD, RT.RESOURCE_TYPE_CD, MT.MATERIAL_TYPE_CD, M.PROFILE_UNITS,
M.COMPARISON_STRING, M.LOW_VALUE_DATE, M.HIGH_VALUE_DATE, M.LOW_VALUE_NUMBER,
M.HIGH_VALUE_NUMBER, M.KPI_INDICATOR, M.PROFILE_INDICATOR, M.DATA_TYPE,
M.AGGREGATION_TYPE, M.CARRY_FORWARD_INDICATOR, M.PROCESS_INDICATOR,
M.VARIANCE_MULTIPLIER, L.LANGUAGE_CD, T.TENANT_CD FROM
SYSREC.MASTER_PROFILE_VARIABLE M JOIN SYSREC.LANGUAGE L ON M.LANGUAGE_ID =
L.LANGUAGE_ID JOIN SYSREC.TENANT T ON M.TENANT_ID = T.TENANT_ID JOIN
SYSREC.MASTER_PROFILE_CALCULATION PC ON M.PROFILE_CALCULATION_ID =
PC.PROFILE_CALCULATION_ID JOIN SYSREC.MASTER_MEASUREMENT_TYPE MSRT ON
M.MEASUREMENT_TYPE_ID = MSRT.MEASUREMENT_TYPE_ID AND M.LANGUAGE_ID =
MSRT.LANGUAGE_ID JOIN SYSREC.MASTER_RESOURCE_TYPE RT ON M.RESOURCE_TYPE_ID =
RT.RESOURCE_TYPE_ID AND M.LANGUAGE_ID = RT.LANGUAGE_ID JOIN
SYSREC.MASTER_MATERIAL_TYPE MT ON M.MATERIAL_TYPE_ID = MT.MATERIAL_TYPE_ID AND
M.LANGUAGE_ID = MT.LANGUAGE_ID;
```

Mandatory profile variables and measurement types

To be able to process some events, you must load mandatory profile variables and measurement types.

Mandatory profile variables

The following profile variables must be loaded:

HS Required for health score related calculations.

RC Required for calculations that are related to the recommendations count.

You can see examples in the profile_variable_upsert_sample_pmq.csv file. This is installed on the Enterprise Service Bus (ESB) node computer in the /var/PMQ/MQSIFileInput/PMQSampleData/Sample_PMQ/MasterData-Set2 folder.

Define profile variables that are based on the design of the IBM Cognos Business Intelligence reports and predictive models.

For example, for the sample models shipped with IBM Predictive Maintenance and Quality, the following profile variables and corresponding measurement types must be defined for the field `profile_variable_cd`:

- AC
- ATIME
- CELLLDX
- CELLLDXX
- CLTX
- CLTXX
- FAIL
- HS
- INSP
- ITIME
- OPHD
- QTY
- RC
- REPC
- REPT
- SETX
- SETXX
- SLTX
- SLTXX

Mandatory measurement types

The following measurement types must be loaded:

HS Required for health score related calculations.

You can see examples of these measurement types in the `measurement_type_upsert_sample_pmq.csv` file. This is installed on the Enterprise Service Bus (ESB) node computer in the `/var/PMQ/MQSIFileInput/PMQSampleData/Sample_PMQ/MasterData-Set1`.

Sample health score and IBM Analytical Decision Management services are configured for these measurement types:

- FAIL
- INSP
- LUBE
- OPHR
- PRS1
- PRS2
- PRS3
- RELH
- REPT
- REPX

- RPM
- R_B1
- R_F1
- TEMP

For health score, define profile variables with the profile calculations for the listed measurement types:

- Measurement of Type
- Measurement Above Limit (except for FAIL)
- Measurement Below Limit (except for FAIL)

Remove master data

Normally master data is not deleted from the analytic database. During testing and development, master data that is not referenced can be removed.

Sample code to remove master data

The following SQL code is an example and must be modified.

```
-- batch batch
DELETE FROM SYSREC.MASTER_BATCH_BATCH M WHERE
M.PRODUCTION_BATCH_ID = (SELECT PB1.PRODUCTION_BATCH_ID FROM
  SYSREC.MASTER_PRODUCTION_BATCH PB1
JOIN SYSREC.LANGUAGE L ON PB1.LANGUAGE_ID = L.LANGUAGE_ID
JOIN SYSREC.TENANT T ON PB1.TENANT_ID = T.TENANT_ID WHERE
PB1.PRODUCTION_BATCH_CD = '1007' AND L.LANGUAGE_CD = 'EN' AND T.TENANT_CD = 'PMQ')
AND
M.RELATED_PRODUCTION_BATCH_ID = (SELECT PB2.PRODUCTION_BATCH_ID FROM
  SYSREC.MASTER_PRODUCTION_BATCH PB2
JOIN SYSREC.LANGUAGE L ON PB2.LANGUAGE_ID = L.LANGUAGE_ID
JOIN SYSREC.TENANT T ON PB2.TENANT_ID = T.TENANT_ID WHERE
PB2.PRODUCTION_BATCH_CD = '1010' AND L.LANGUAGE_CD = 'EN' AND T.TENANT_CD = 'PMQ');

-- event code
DELETE FROM SYSREC.MASTER_EVENT_CODE M WHERE
M.EVENT_CODE_SET = 'FAIL' AND
M.EVENT_CODE = 'X101' AND
M.LANGUAGE_ID = (SELECT L.LANGUAGE_ID FROM SYSREC.LANGUAGE L WHERE
  L.LANGUAGE_CD = 'EN') AND
M.TENANT_ID = (SELECT T.TENANT_ID FROM SYSREC.TENANT T WHERE
  T.TENANT_CD = 'PMQ');

-- event type
DELETE FROM SYSREC.MASTER_EVENT_TYPE M WHERE
M.EVENT_TYPE_CD = 'ALARM' AND
M.LANGUAGE_ID = (SELECT L.LANGUAGE_ID FROM SYSREC.LANGUAGE L WHERE
  L.LANGUAGE_CD = 'EN') AND
M.TENANT_ID = (SELECT T.TENANT_ID FROM SYSREC.TENANT T WHERE
  T.TENANT_CD = 'PMQ');

-- group dim
DELETE FROM SYSREC.MASTER_GROUP_DIM M WHERE
M.GROUP_TYPE_CODE = 'ORG' AND
M.GROUP_MEMBER_CODE = 'C1' AND
M.LANGUAGE_ID = (SELECT L.LANGUAGE_ID FROM SYSREC.LANGUAGE L WHERE
  L.LANGUAGE_CD = 'EN') AND
M.TENANT_ID = (SELECT T.TENANT_ID FROM SYSREC.TENANT T WHERE
  T.TENANT_CD = 'PMQ');

-- location
DELETE FROM SYSREC.MASTER_LOCATION M WHERE
M.LOCATION_CD = 'Room1' AND
```

```

M.LANGUAGE_ID = (SELECT L.LANGUAGE_ID FROM SYSREC.LANGUAGE L WHERE
L.LANGUAGE_CD = 'EN') AND
M.TENANT_ID = (SELECT T.TENANT_ID FROM SYSREC.TENANT T WHERE
T.TENANT_CD = 'PMQ');

-- material
DELETE FROM SYSREC.MASTER_MATERIAL M WHERE
M.MATERIAL_CD = '20390' AND
M.LANGUAGE_ID = (SELECT L.LANGUAGE_ID FROM SYSREC.LANGUAGE L WHERE
L.LANGUAGE_CD = 'EN') AND
M.TENANT_ID = (SELECT T.TENANT_ID FROM SYSREC.TENANT T WHERE
T.TENANT_CD = 'PMQ');

-- material type
DELETE FROM SYSREC.MASTER_MATERIAL_TYPE M WHERE
M.MATERIAL_TYPE_CD = 'PROD' AND
M.LANGUAGE_ID = (SELECT L.LANGUAGE_ID FROM SYSREC.LANGUAGE L WHERE
L.LANGUAGE_CD = 'EN') AND
M.TENANT_ID = (SELECT T.TENANT_ID FROM SYSREC.TENANT T WHERE
T.TENANT_CD = 'PMQ');

-- measurement type
DELETE FROM SYSREC.MASTER_MEASUREMENT_TYPE M WHERE
M.MEASUREMENT_TYPE_CD = 'SET' AND
M.LANGUAGE_ID = (SELECT L.LANGUAGE_ID FROM SYSREC.LANGUAGE L WHERE
L.LANGUAGE_CD = 'EN') AND
M.TENANT_ID = (SELECT T.TENANT_ID FROM SYSREC.TENANT T WHERE
T.TENANT_CD = 'PMQ');

-- process hierarchy
DELETE FROM SYSREC.PROCESS_HIERARCHY M WHERE
M.PROCESS_ID = (SELECT P.PROCESS_ID FROM SYSREC.MASTER_PROCESS P WHERE
P.PROCESS_CD = 'SET') AND
M.LANGUAGE_ID = (SELECT L.LANGUAGE_ID FROM SYSREC.LANGUAGE L WHERE
L.LANGUAGE_CD = 'EN') AND
M.TENANT_ID = (SELECT T.TENANT_ID FROM SYSREC.TENANT T WHERE
T.TENANT_CD = 'PMQ');

-- process
DELETE FROM SYSREC.MASTER_PROCESS M WHERE
M.PROCESS_CD = 'SET' AND
M.LANGUAGE_ID = (SELECT L.LANGUAGE_ID FROM SYSREC.LANGUAGE L WHERE
L.LANGUAGE_CD = 'EN') AND
M.TENANT_ID = (SELECT T.TENANT_ID FROM SYSREC.TENANT T WHERE
T.TENANT_CD = 'PMQ');

-- product
DELETE FROM SYSREC.MASTER_PRODUCT M WHERE
M.PRODUCT_CD = '2190890' AND
M.LANGUAGE_ID = (SELECT L.LANGUAGE_ID FROM SYSREC.LANGUAGE L WHERE
L.LANGUAGE_CD = 'EN') AND
M.TENANT_ID = (SELECT T.TENANT_ID FROM SYSREC.TENANT T WHERE
T.TENANT_CD = 'PMQ');

-- production_batch
DELETE FROM SYSREC.MASTER_PRODUCTION_BATCH M WHERE
M.PRODUCTION_BATCH_CD = '1000' AND
M.LANGUAGE_ID = (SELECT L.LANGUAGE_ID FROM SYSREC.LANGUAGE L WHERE
L.LANGUAGE_CD = 'EN') AND
M.TENANT_ID = (SELECT T.TENANT_ID FROM SYSREC.TENANT T WHERE
T.TENANT_CD = 'PMQ');

-- profile variable
DELETE FROM SYSREC.MASTER_PROFILE_VARIABLE M WHERE
M.PROFILE_VARIABLE_CD = 'SET' AND
M.LANGUAGE_ID = (SELECT L.LANGUAGE_ID FROM SYSREC.LANGUAGE L WHERE
L.LANGUAGE_CD = 'EN') AND

```

```

M.TENANT_ID = (SELECT T.TENANT_ID FROM SYSREC.TENANT T WHERE
  T.TENANT_CD = 'PMQ');

-- resource hierarchy
DELETE FROM SYSREC.RESOURCE_HIERARCHY M WHERE
M.RESOURCE_ID = (SELECT R.RESOURCE_ID FROM SYSREC.MASTER_RESOURCE R WHERE
  R.SERIAL_NO = '13580' AND R.MODEL = 'M100' ) AND
M.LANGUAGE_ID = (SELECT L.LANGUAGE_ID FROM SYSREC.LANGUAGE L WHERE
  L.LANGUAGE_CD = 'EN') AND
M.TENANT_ID = (SELECT T.TENANT_ID FROM SYSREC.TENANT T WHERE
  T.TENANT_CD = 'PMQ');

-- resource
DELETE FROM SYSREC.MASTER_RESOURCE M WHERE
M.SERIAL_NO = '13580' AND
M.MODEL = 'M100' AND
M.LANGUAGE_ID = (SELECT L.LANGUAGE_ID FROM SYSREC.LANGUAGE L WHERE
  L.LANGUAGE_CD = 'EN') AND
M.TENANT_ID = (SELECT T.TENANT_ID FROM SYSREC.TENANT T WHERE
  T.TENANT_CD = 'PMQ');

-- source system
DELETE FROM SYSREC.MASTER_SOURCE_SYSTEM M WHERE
M.SOURCE_SYSTEM_CD = 'PREDMAIT' AND
M.LANGUAGE_ID = (SELECT L.LANGUAGE_ID FROM SYSREC.LANGUAGE L WHERE
  L.LANGUAGE_CD = 'EN') AND
M.TENANT_ID = (SELECT T.TENANT_ID FROM SYSREC.TENANT T WHERE
  T.TENANT_CD = 'PMQ');

-- supplier
DELETE FROM SYSREC.MASTER_SUPPLIER M WHERE
M.SUPPLIER_CD = 'WS' AND
M.LANGUAGE_ID = (SELECT L.LANGUAGE_ID FROM SYSREC.LANGUAGE L WHERE
  L.LANGUAGE_CD = 'EN') AND
M.TENANT_ID = (SELECT T.TENANT_ID FROM SYSREC.TENANT T WHERE
  T.TENANT_CD = 'PMQ');

```

Note:

The contents of the SYSREC.LANGUAGE, SYSREC.MASTER_PROFILE_CALCULATION, SYSREC.TENANT, SYSREC.MASTER_VALUE_TYPE, and SYSREC.MASTER_RESOURCE_TYPE tables are normally not deleted when master data is removed.

Appendix C. Data access API

The data access API for Predictive Maintenance and Quality provides a way to query the database. You use the data access API to run basic and generic queries, and data ingestion into the Resource KPI table.

Basic query

Predictive Maintenance and Quality provides an API that you can use to run basic queries against the resource_kpi table.

To run a basic query, run a shell command similar to the following example:

```
curl -k -u user1:passw0rd "https://<hostname>:<port>/ibm/pmq/api/kpi?columns=<COLUMN_1>,<COLUMN_2>,<COLUMN_3>&timeStart=<yyyy-mm-dd>&timeEnd=<yyyy-mm-dd>&filter=<COLUMN_FILTER>&entities=(resource_sub_type,resource_name)"
```

The options of this command are described in the following table.

Table 82. Curl command for basic queries

Field	Definition	Comments
columns	Collects attributes and key values. The following attribute values are the potential attribute values: ID, RESOURCE_CD1, RESOURCE_CD2, RESOURCE_SUB_TYPE, RESOURCE_NAME. If no key is defined, all possible keys are returned.	Required
timeStart, timeEnd	Time range of the events.	Required
filter	Provides a filter so that the API returns rows that meet the filter.	Optional
entities	Entity list that you want to query. For example, resource_sub_type, resource_name, or resource_sub_type.	Optional

The following shell commands are examples:

```
curl -k -u user1:passw0rd "https://localhost:9443/ibm/pmq/api/kpi?columns=ID,RESOURCE_SUB_TYPE,RESOURCE_NAME,FAIL_Below_Limit&timeStart=2011-11-17&timeEnd=2011-11-17T00:00:00-06:00&filter=FAIL_Below_Limit=1"
```

```
curl -k -u user1:passw0rd "https://localhost:9443/ibm/pmq/api/kpi?columns=ID,RESOURCE_SUB_TYPE,RESOURCE_NAME&timeStart=2011-11-17&timeEnd=2011-11-17T00:00:00-06:00&entities=(Pole,7053272),(Pole,7053289)"
```

```
curl -k -u user1:passwd "https://localhost:9443/ibm/pmq/api/
kpi?columns=ID,RESOURCE_SUB_TYPE,RESOURCE_NAME&timeStart=2011-11-17
&timeEnd=2011-11-17T00:00:00-06:00&entities=(Pole,)"
```

Generic query

Predictive Maintenance and Quality provides an API that you can use to run generic queries.

To run a generic query, run a shell command similar to the following example:

```
curl -k -u user1:passwd "https://<hostname>:<port>/ibm/pmq/api/
common?table=<table_name>&kpi=<true|false>&style=<transaction>
&columns=<COLUMN_1>,<COLUMN_2>&timeStart=<yyyy-mm-dd>&timeEnd=<yyyy-mm-dd>
&filter=<COLUMN_FILTER>&entities=<entity_list>"
```

The options of this command are described in the following table.

Table 83. Curl command for generic queries

Field	Definition	Comments
table	Defines on which table you want to run the query.	Required
kpi	Defines if the table is a KPI table or not. The values include true or false.	Optional
style	Defines if the KPI table saves transactional data or dimensional data. The values include transaction and dimension. Currently, only transaction is available.	Required when the KPI value is true.
columns	A collection of attributes and key values that are only available when the KPI value is true. Any column names except KEY/VALUE can be used as attributes. If no attribute is defined, all possible attributes are returned. Any value in the KEY column can be used as a key value. If no key is defined, all possible keys are returned.	Required when the KPI value is true.
timeStart, timeEnd	Time range of the events.	Required when the KPI value is true.
filter	Provides a filter so that the API returns rows that meet the filter. The filter on works when the KPI value is true.	Optional
entities	Entity list that you want to query.	Optional

Note: To avoid queries that return too much data, there is a limit of 100,000 rows.

The following shell commands are examples:

```
curl -k -u user1:passw0rd "https://localhost:9443/ibm/pmq/api/
common?table=language"
```

```
curl -k -u user1:passw0rd "https://localhost:9443/ibm/pmq/api/
common?table=anylift"
```

```
curl -k -u user1:passw0rd "https://localhost:9443/ibm/pmq/api/
common?table=anylift&kpi=true&timeStart=2011-11-17&timeEnd=2017-11-17
&style=transaction&columns=rms_x,rms_y,re_opens&filter=re_opens=0"
```

```
curl -k -u user1:passw0rd "https://localhost:9443/ibm/pmq/api/
common?table=anylift&kpi=true&timeStart=2011-11-17&timeEnd=2017-11-17
&style=transaction&entities=357042064191878,357042064191879"
```

Generic ingestion for KPI table

Predictive Maintenance and Quality provides an API that you can use to run a generic ingestion for the KPI table to insert data into the database.

To run a generic ingestion for the KPI table, run a shell command similar to the following example:

```
curl -k -X POST -H "Content-Type:application/json" -u user1:passw0rd --data
@c:\sample_json.txt "https://<hostname>:<port>/ibm/pmq/api/common"
```

The following text is an example of a sample_json.txt file.

```
[{"table": "anylift",
"style": "transaction",
"timestamp": {"timestamp": "yyyy-MM-dd HH:mm:ss.SSS"},
"content": {
"orgId": "5w6626",
"entityId": "357042064191878",
"eventType": "kpi-d",
"timestamp": "2017-05-02 11:29:01.358",
"payload": {
"d": {
"closed_time": "2017-05-01T11:29:56+00:00",
"closing_duration": 3.048,
"current_floor": 0,
"initial_open_duration": 1.985,
"ken": "11161293",
"noise_door": -36.03,
"opening_duration": 6.167,
"opening_start": "2017-05-01T11:29:45+00:00",
"re_opens": "0.00",
"re_opens_col": 0,
"rms_x": 0.06928,
"rms_y": 0.07132,
"rms_z": 0.1712,
"version": 0
}
}
}
}]
```

The options of this command are described in the following table.

Table 84. Curl command for generic ingestion for KPI table

Option	Definition	Comments
table	Defines on which table you want to run the query.	Required
style	Defines if the KPI table saves transactional data or dimensional data. The values include transaction and dimension. Currently, only transaction is available.	Required
timestamp	A JSON that defines the Java format of the timestamp. If you need any value to be stored as timestamp, you must provide a format so that the data access API can parse the time value correctly. Note: Values under d are stored as a string.	Required if the content has timestamp values.

The following shell commands are examples:

```
curl -k -X POST -H "Content-Type:application/json" -u user1:passw0rd --data @c:\da_json.txt "https://localhost:9443/ibm/pmq/api/common"
```

Appendix D. IBM Cognos Framework Manager model description

IBM Predictive Maintenance and Quality uses IBM Cognos Framework Manager to model the metadata for reports.

IBM Cognos Framework Manager is a metadata modeling tool that drives query generation for IBM Cognos software. A model is a collection of metadata that includes physical information and business information for one or more data sources. IBM Cognos software enables performance management on normalized and denormalized relational data sources and a variety of OLAP data sources.

For information on modifying or creating Framework Manager models, see the *IBM Cognos Framework Manager User Guide* and *IBM Cognos Framework Manager - Guidelines for Modeling Metadata*. These documents are available at IBM Cognos Business Intelligence Knowledge Center (<http://www.ibm.com/support/knowledgecenter/SSEP7J>).

The Framework Manager model consists of three layers:

- Database layer
- Logical layer
- Dimensional layer

Each of these layers is in a separate namespace. The dimensional layer is published to a package for use in reporting.

IBM Cognos Framework Manager model database layer

The physical, or database, layer contains a database query subject for every table in the physical data model. The database layer also contains alias shortcuts, which behave as if they were a copy of the original object with completely independent behavior.

The alias shortcuts are provided for two situations:

- To eliminate ambiguity for an entity that may be involved in multiple relationships, including the following items:
 - location and location (resource)
 - material_type and material_type (profile_variable)
 - resource_type and resource_type (profile_variable)
 - production_batch and production_batch (related)
- To enable you to query multiple copies of the same table in different roles, including the group_dim_1 to 5 values

If a database entity includes the language_id or tenant_id attributes, the database query subject includes a parameterized filter for each that selects only one tenant or language. Language is based on the used locale settings. Localization is implemented for the FM model as well. Users can select the language of their choice from the Active Language drop down menu and change the model language.

The database layer contains all of the entity relationships. The central entities are largely modeled in star or snowflake schemas, shown in the following diagrams. These parameters must be set after master data has been loaded or reloaded and before publishing the package. If these parameters are not set correctly, no data will be returned in the reports. To change the values, simply open the parameter map, double click on the value for each parameter and type over it.

A parameter map for language supports the localization of report data. Language codes for English (EN), Simplified Chinese (SC), Traditional Chinese (TC), French (FR), Japanese (JP), and Portuguese (Brazil)(PT) are configured in the parameter map.

Typically the central fact has cardinality 1,N and related objects are 1,1, in order to eliminate the need for relationships outside of the database layer. All joins are modeled as inner joins on the understanding that the data integration layer populates a default value for all references in the absence of a valid value.

The following diagram shows the star schema for the event_observation table.

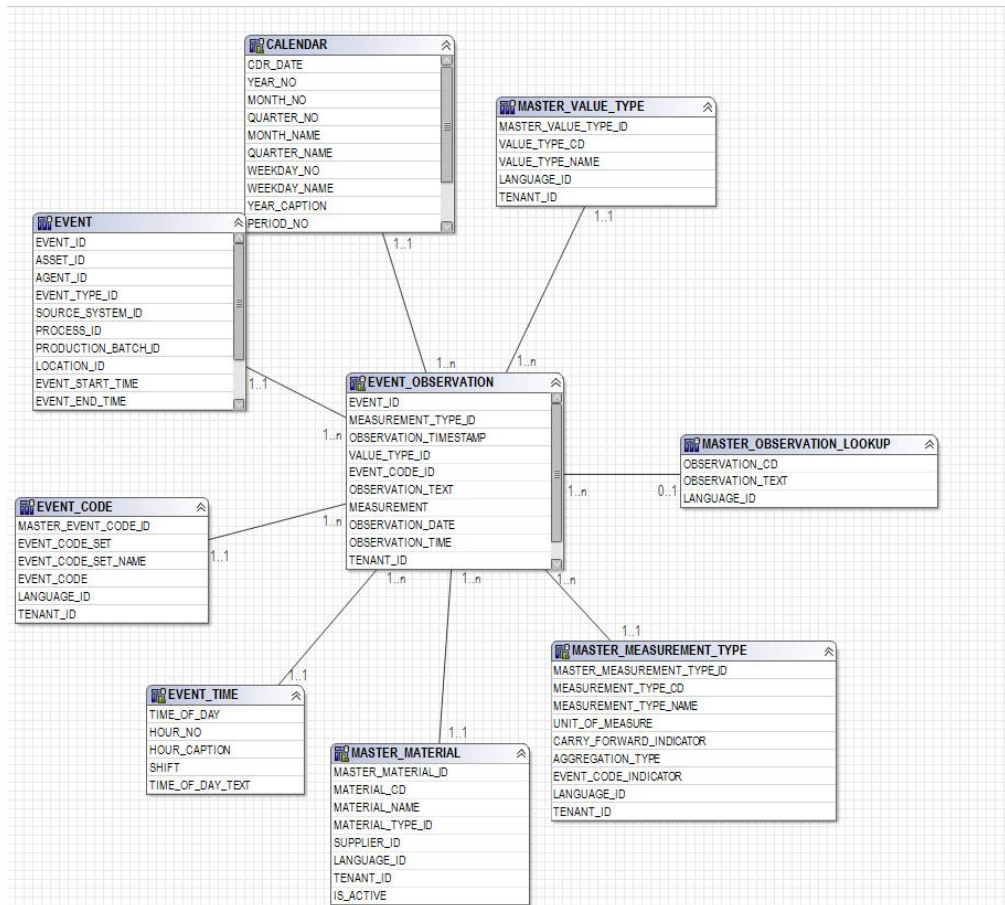


Figure 95. The event_observation star schema

The following diagram shows the star schema for the resource_profile table.

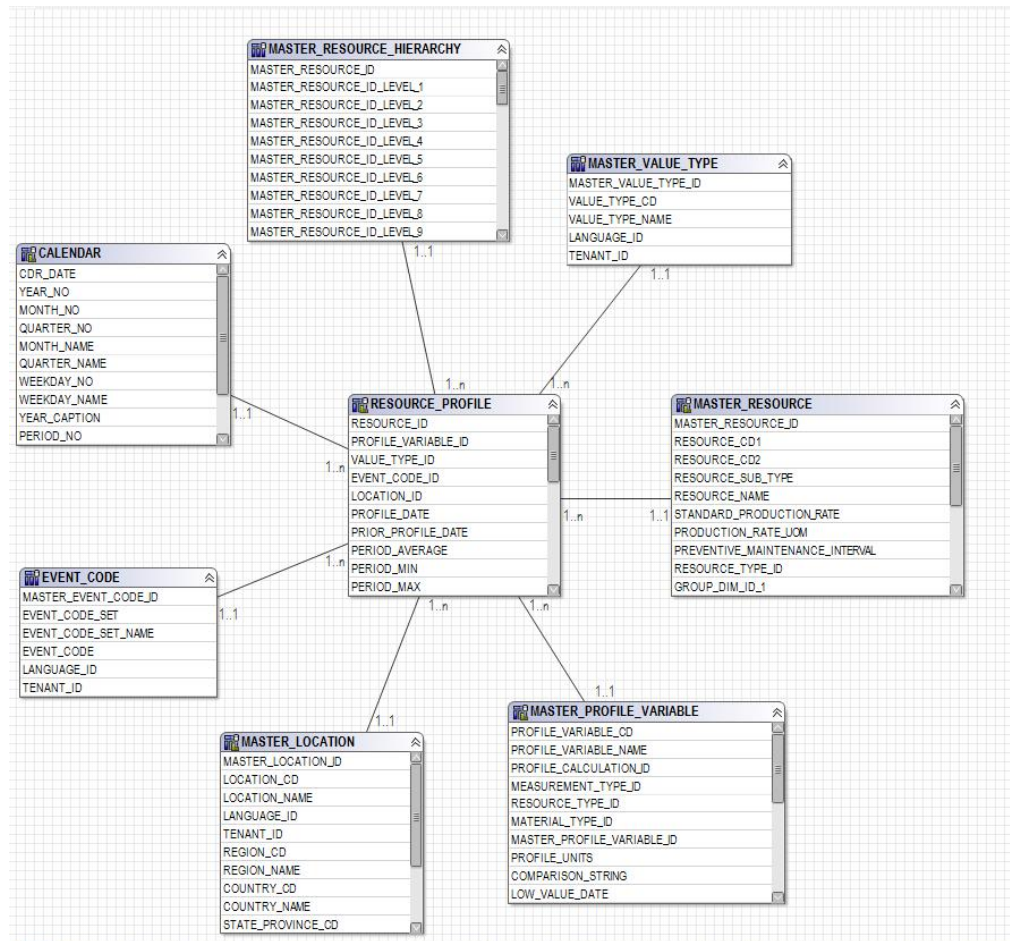


Figure 96. The resource_profile star schema

The following diagram shows the star schema for the resource_kpi table.

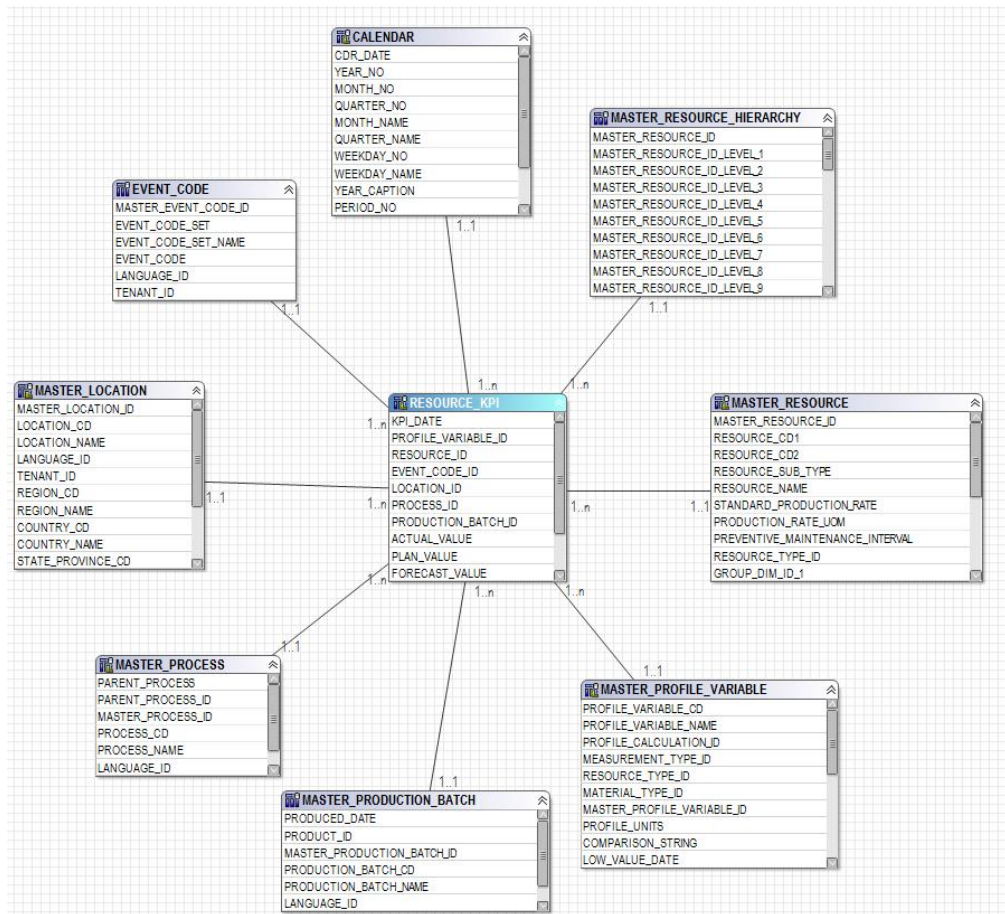


Figure 97. The resource_kpi star schema

The following diagram shows the star schema for the material_profile table.

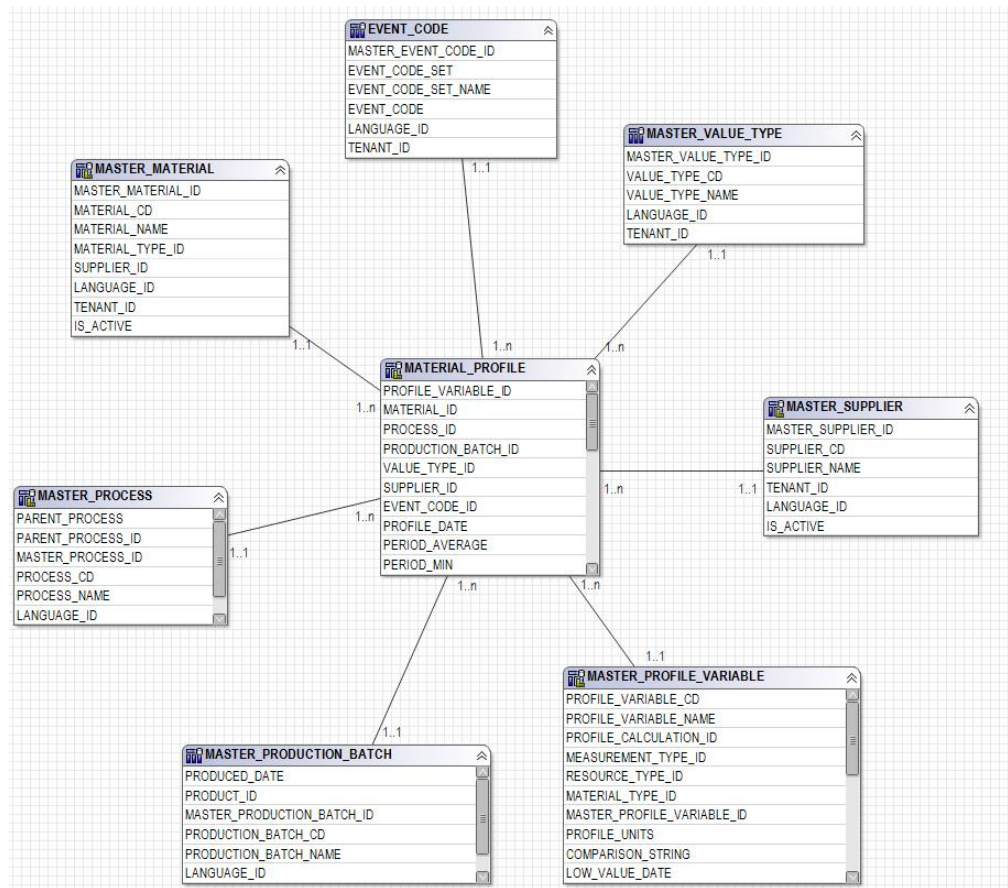


Figure 98. The material_profile star schema

The following diagram shows the star schema for the process_profile table.

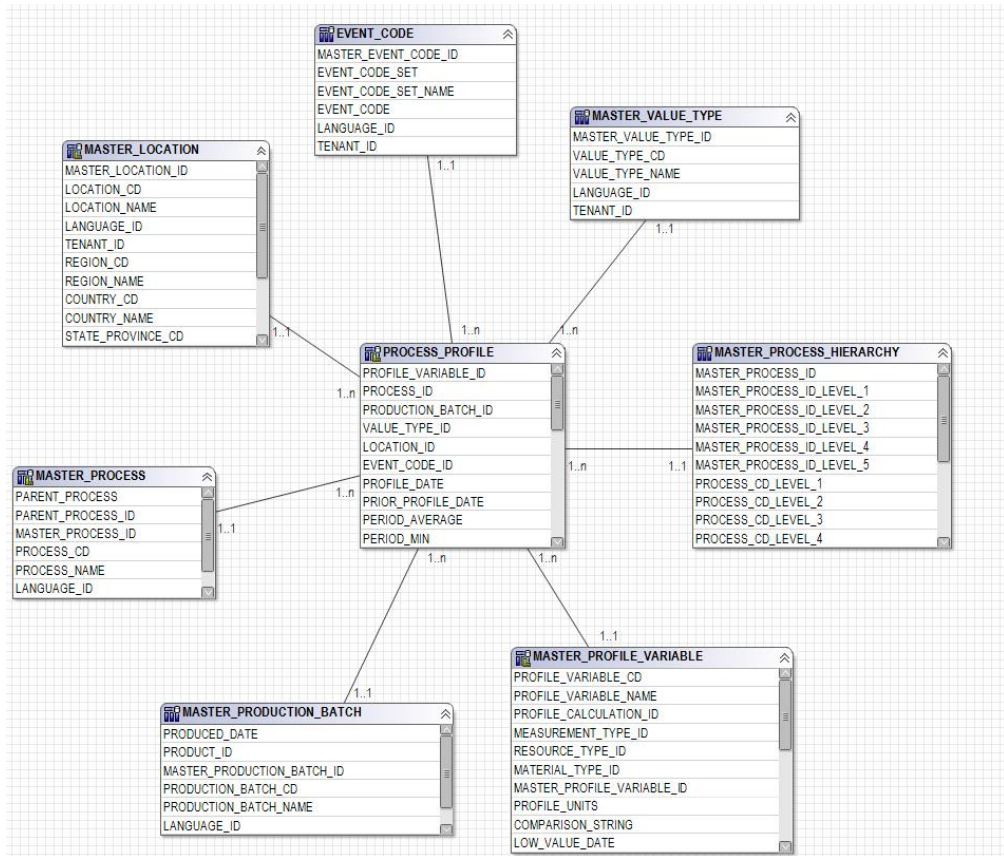


Figure 99. The process_profile star schema

The following diagram shows the star schema for the process_kpi table.

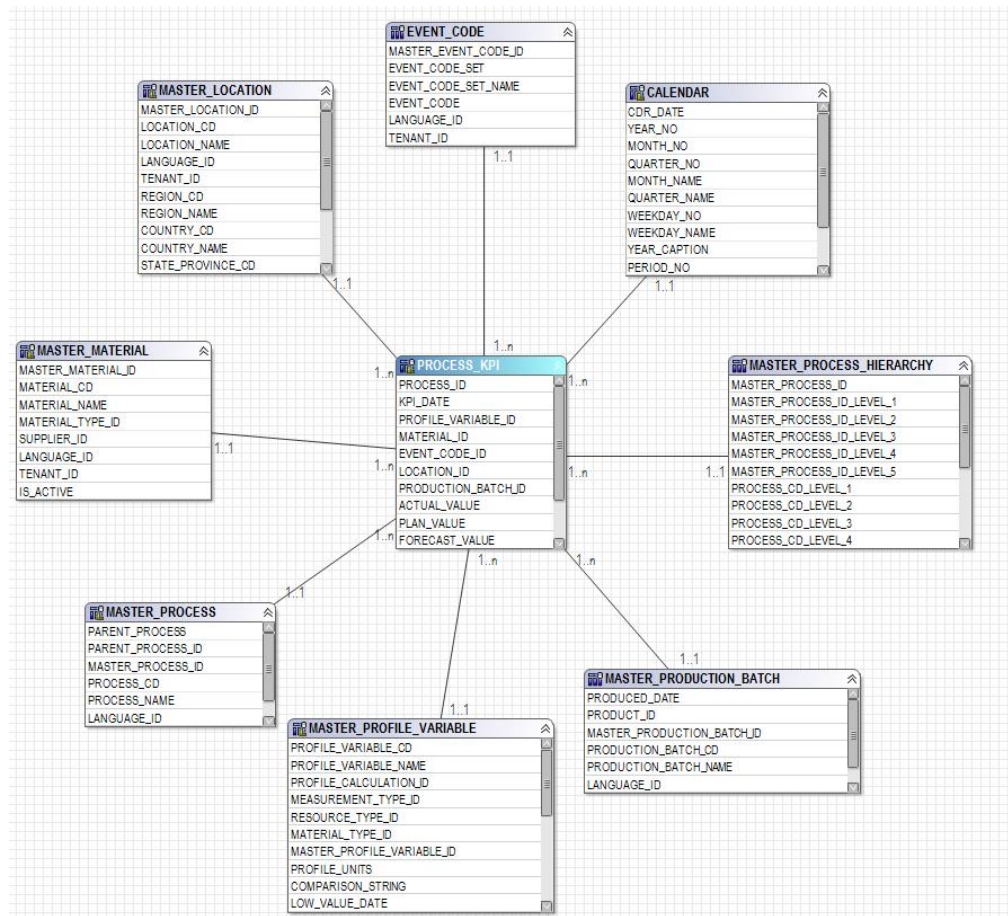


Figure 100. The process_kpi star schema

The following diagram shows the star schema for the lifetime_profile table.

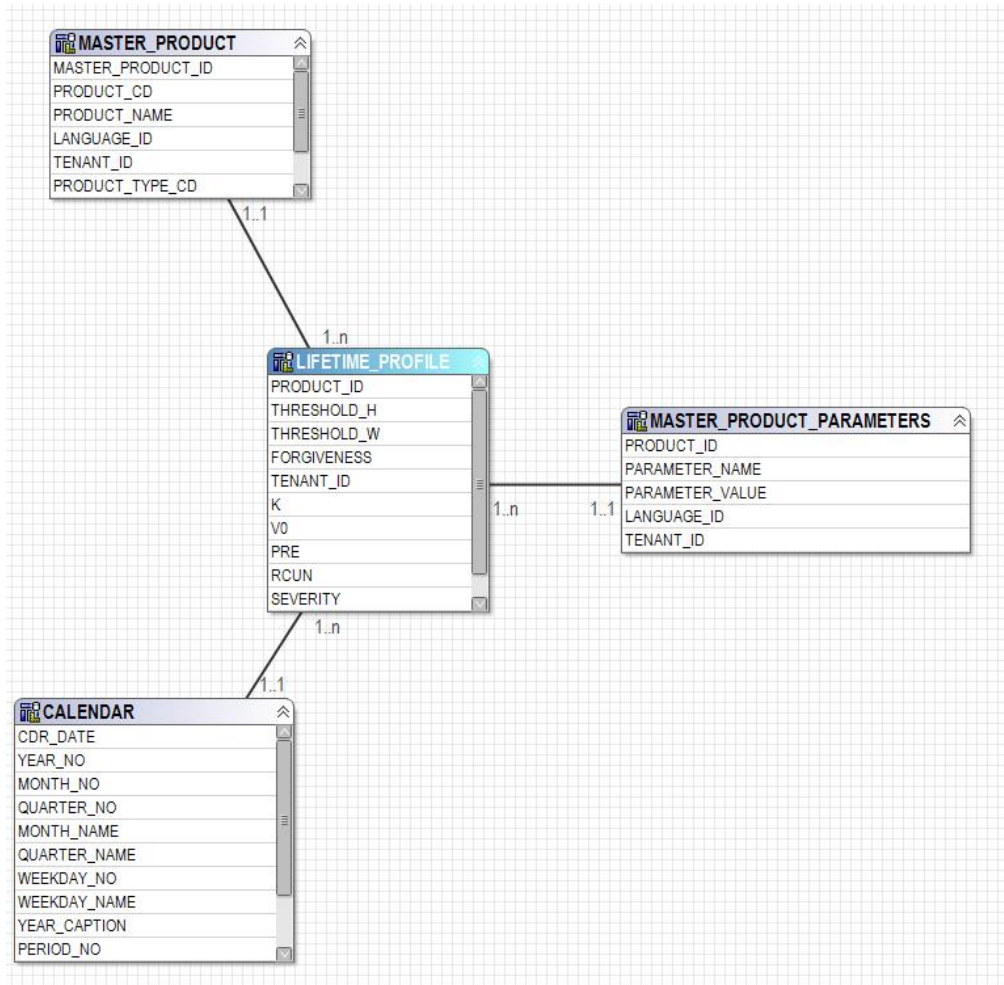


Figure 101. The lifetime_profile star schema

The following diagram shows the star schema for the lifetime_kpi table.

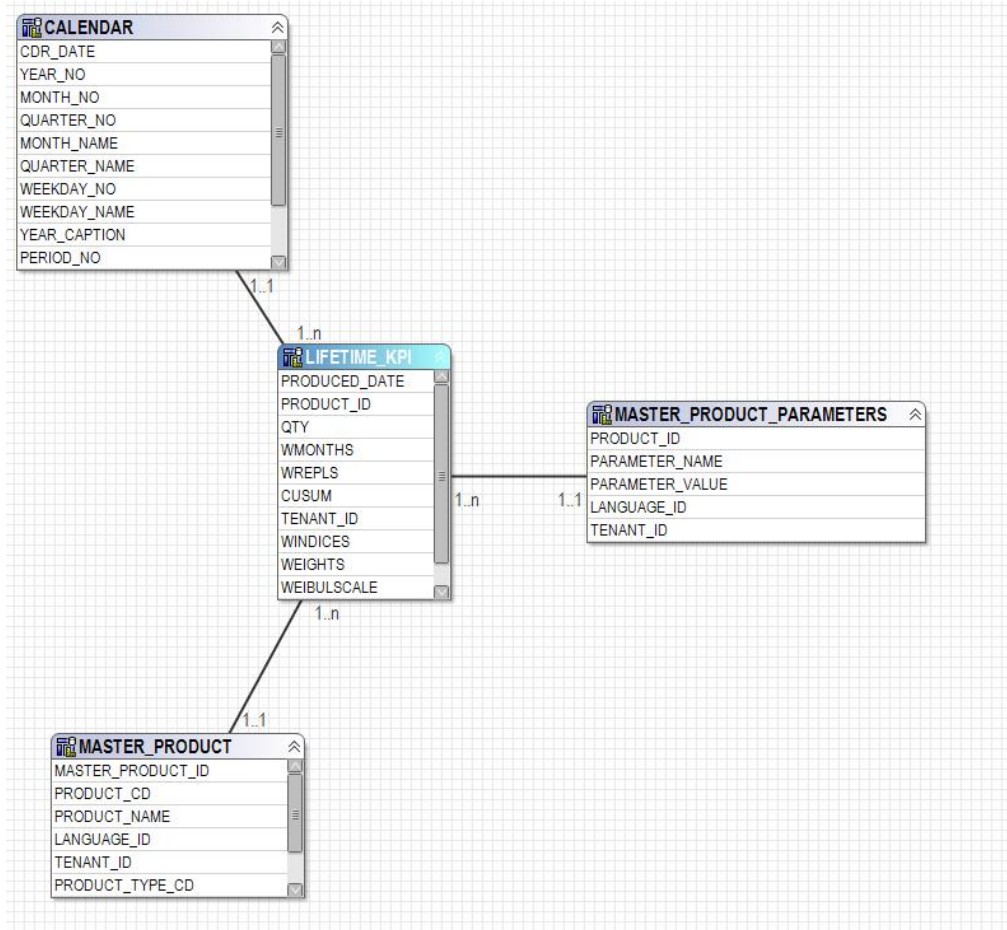


Figure 102. The `lifetime_kpi` star schema

The following diagram shows the star schema for the `maintenance_trends` table.

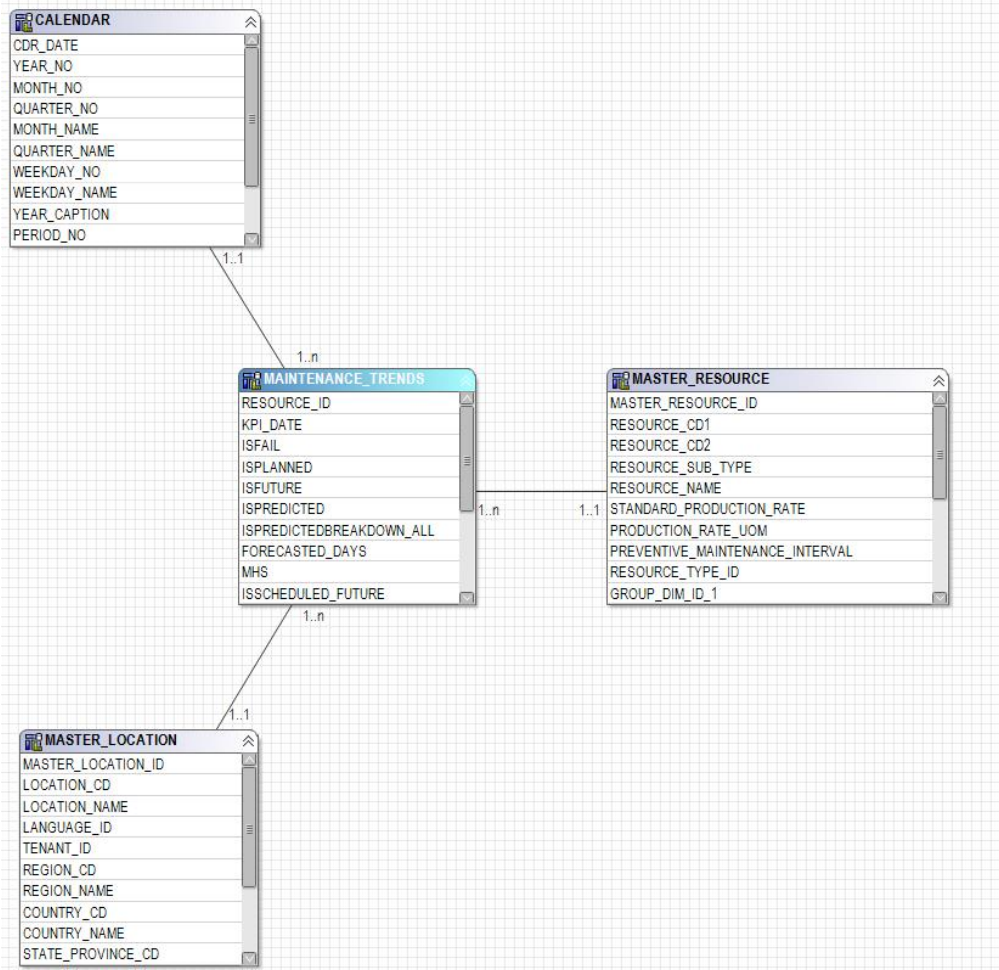


Figure 103. The maintenance_trends star schema

The following diagram shows the star schema for the product_kpi table.

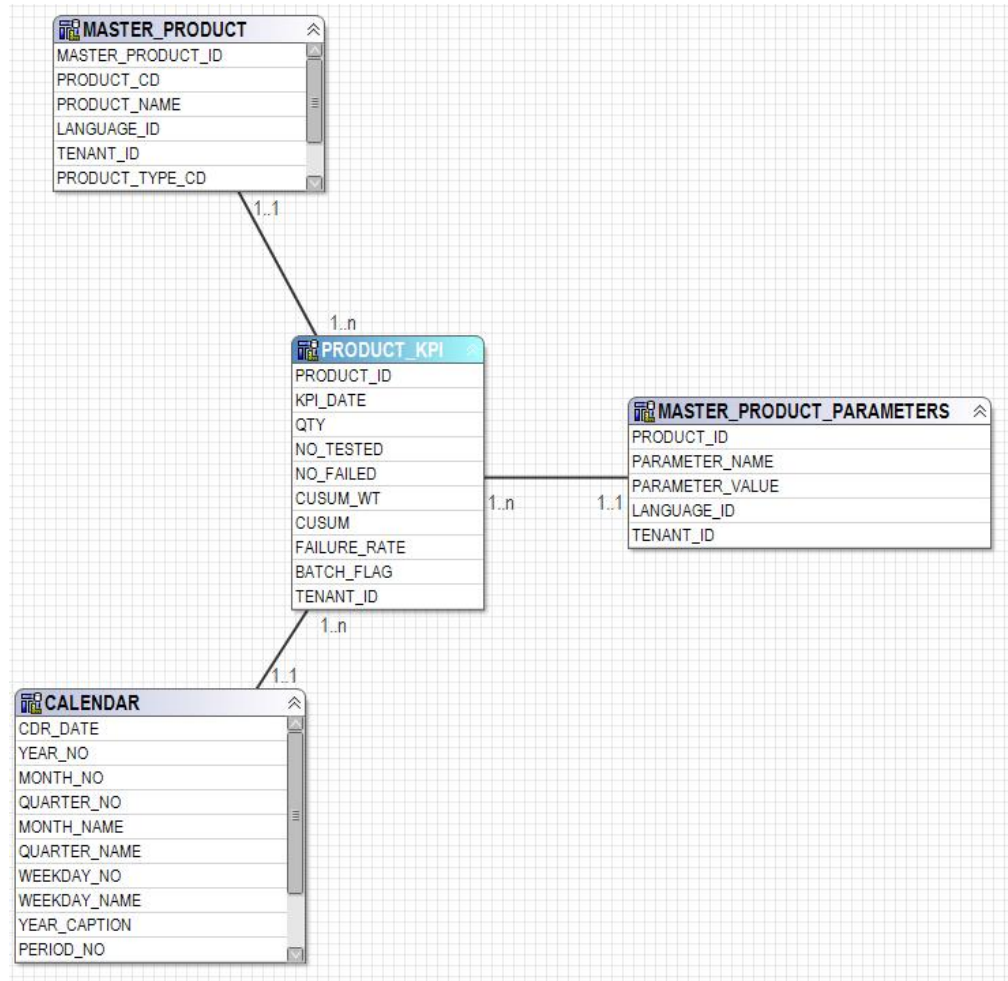


Figure 104. The product_kpi star schema

The following diagram shows the star schema for the product_profile table.

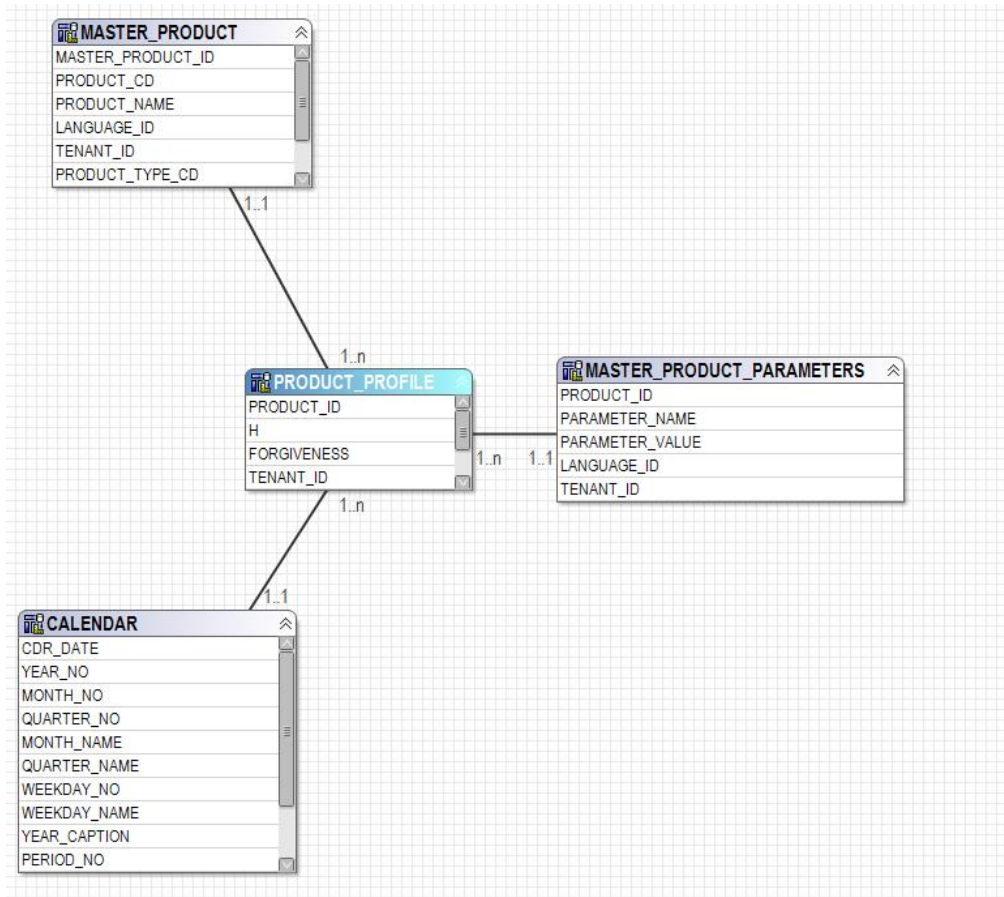


Figure 105. The product_profile star schema

The following diagram shows the star schema for the service table.

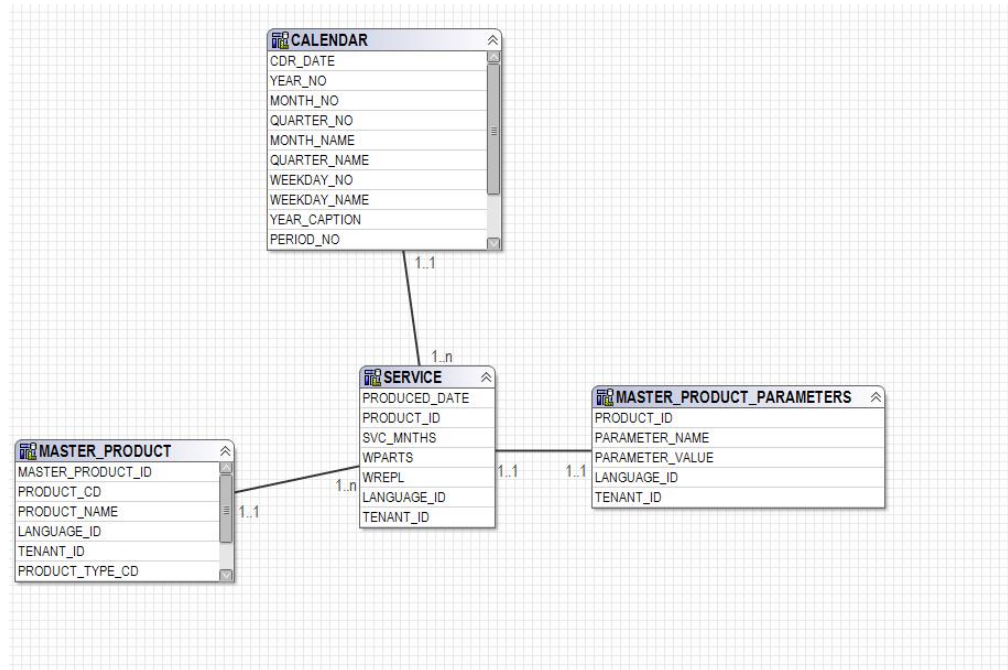


Figure 106. The service star schema

The following diagram shows the star schema for the parametric_kpi table.

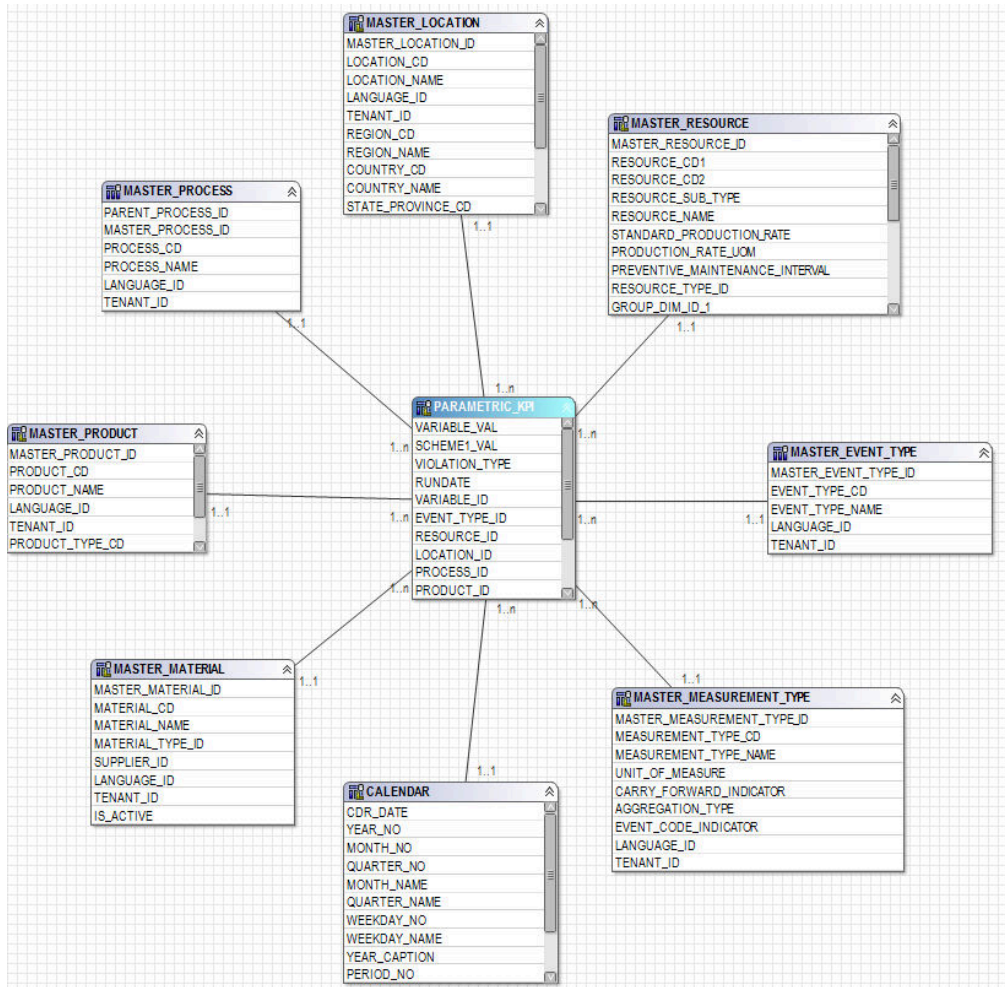


Figure 107. The `parametric_kpi` star schema

The following diagram shows the star schema for the `parametric_profile` table.

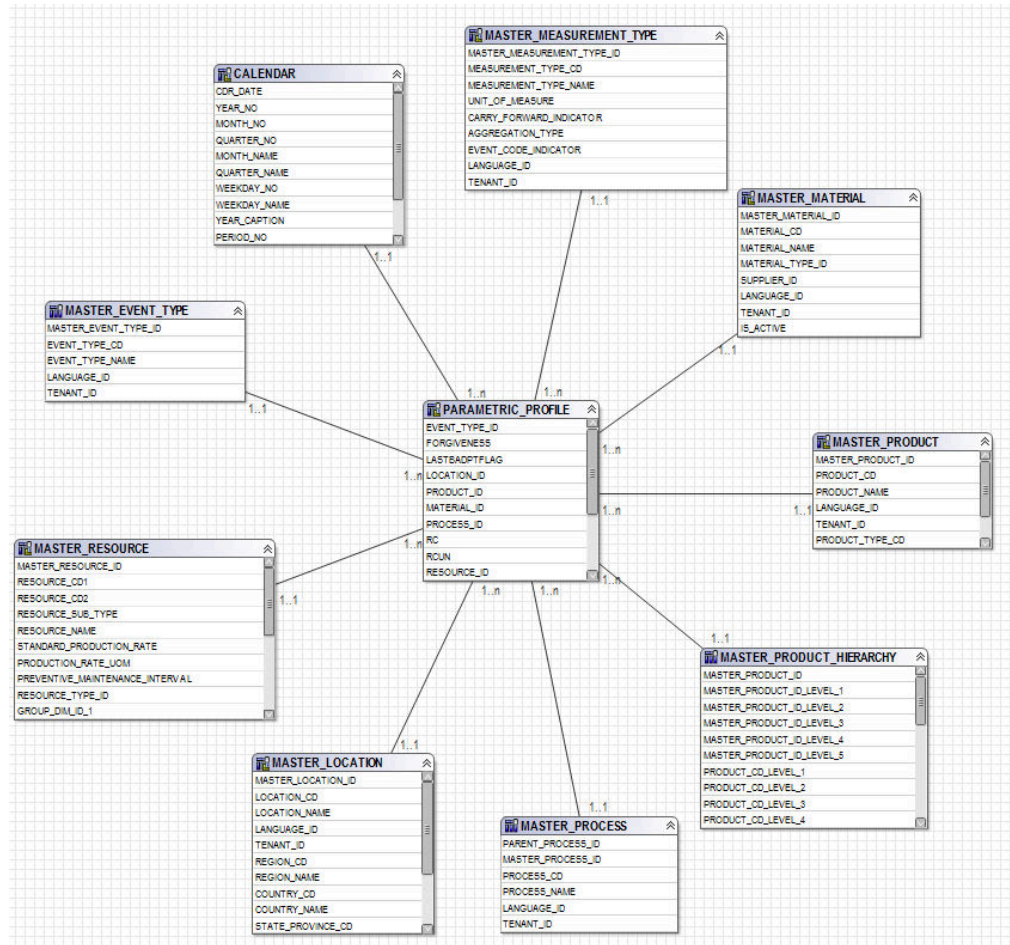


Figure 108. The parametric_profile star schema

The following diagram shows the star schema for the profile_parameter table.

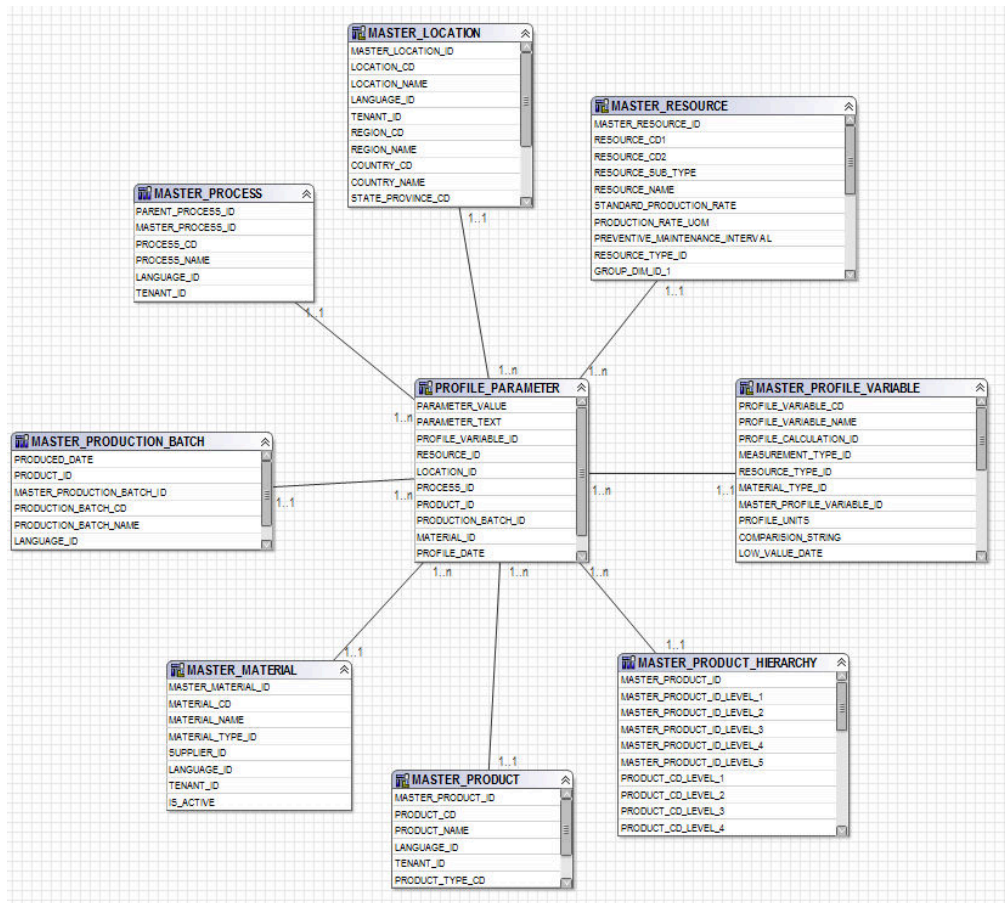


Figure 109. The profile_parameter star schema

IBM Cognos Framework Manager model logical layer

The logical layer contains query subjects that draw data from the database query subjects and present it in a more consumable format.

Attributes are renamed to eliminate underscores and to use sentence case. In some cases, physical entities are combined into one query subject. In particular, the following snowflake dimensions are combined together to accommodate master data reporting and avoid cross product result sets:

- Query subject Profile Variable contains attributes of profile_variable, measurement_type, profile_calculation, resource_type (profile_variable), and material_type (profile_variable).
- Query subject Material contains attributes of material, supplier and material_type.
- Query subject Production Batch contains attributes of production_batch and product
- Query subject Related Batch contains attributes of production_batch, batch_batch and production_batch (related).
- Query subject Resource contains attributes of resource, resource_type, location (resource), and group_dim_1 to 5.

- Query subject Event Observation contains attributes of event, event_observation, and event_resource.

The query subjects are organized into a folder for dimensions and a separate namespace for each logical fact. The fact query subjects contain additional calculated attributes that are included in dimensional layer measure dimensions.

IBM Cognos Framework Manager model dimensional layer

The dimensional layer contains the hierarchies and measure dimensions for publication to a package. Each dimension in the logical layer has a dimension in the dimensional layer with one or more hierarchies defined. The hierarchies usually include the caption field twice, once as a caption for the level, once as an attribute that can be used in report filters. All hierarchies are sorted.

Each measure dimension is in a separate namespace for the fact. Also included in the namespace are shortcuts to all the dimensions that have scope for that fact. Any dimension shortcut that is inside the namespace of the fact can also be consumed from outside of the namespace by IBM Cognos Business Intelligence reports.

Key Performance Indicator (KPI) tables include one measure with flexible aggregation. Based on the aggregation type in Profile Variable, the measure will either total the Actual Value or will calculate an average based on Sum of Actual Value / Sum of Measure Count. This requires that the data integration layer populates Measure Count with the actual number of observations for measures with aggregation type of Average and that it adds together measures that do not naturally seem to be additive, for example, temperature or pressure. The profile tables include a similar measure for flexible aggregation, with the addition of a check for Value Type = Actual.

IBM Cognos Framework Manager model security

No security is defined for the IBM Cognos Framework Manager model other than the provision for filtering by the tenant_id parameter on the physical layer. These query subject filters can be converted to security filters that are based on user IDs, allowing multi-tenant access to one database.

The Framework Manager model gives the ability to filter by the tenant_id parameter on the physical layer. As a preliminary measure to define security for the Framework Manager model, convert database query subject filters to security filters that are based on user IDs, allowing multi-tenant access to one database.

Query mode

The IBM Predictive Maintenance and Quality reports uses IBM Cognos Compatible Query Mode, which is the supported mode for all of the reports.

Using Compatible Query Mode to see real-time data

To see real-time data, you must ensure that caching is disabled in Dynamic Query mode and switch IBM Predictive Maintenance and Quality to use Compatible Query Mode.

Procedure

1. To turn off query reuse, open the CQEConfig.xml file that is in {IBM Cognos Install Directory}/configuration and edit the QueryEngine section by typing the following information.

```
<section name="QueryEngine">
  <!-- Description: queryReuse feature -->
  <!-- value="0" means disable the feature -->
  <!-- default is value="5" which means cache up to 5result sets per session -->
  <entry name=queryReuse" value="0"/>
  ...
</section>
```

2. Restart the IBM Cognos Business Intelligence server.
3. In IBM Cognos Administration, ensure that the data source defined for the IBM Predictive Maintenance and Quality database has native and JDBC connection definitions.
4. In IBM Framework Manager, select the project and change the **Query Mode** property to Compatible.
5. Publish the **IBMPMQ** package in Compatible mode by not selecting the check box to publish in Dynamic Query Mode when prompted.

Appendix E. IBM Predictive Maintenance and Quality Artifacts

IBM Predictive Maintenance and Quality (PMQ) artifacts contain the configuration files that provide connections to customer data, predictive models, rules, dashboards, reports, and external systems.

PMQ artifacts also contain sample data to aid in the understanding of how PMQ connects, manages, and analyzes data to produce business tools in the form of reports, dashboards, or maintenance work orders. These artifacts can be modified, as explained in this solution guide, for additional asset model requirements, event types, custom reports, or connections to other external data sources or systems of engagement.

Data model

The data model file name is `IBMPMQ.sql`. This DDL contains scripts to create all the tables that form the IBM Predictive Maintenance and Quality Master/Event/Profile Data mart. It contains stored procedures for the initial set up of Language and Tenant Data to perform basic operations required by Predictive Maintenance and Quality functions.

IBM InfoSphere Master Data Management Collaboration Server file

The IBM InfoSphere MDM Collaboration Server data model file name is `IBMPMQ.zip`. This is a company archive file that contains all the templates, reports, and data of the MDM CE data model specific to PMQ Master Data.

IBM Integration Bus and ESB artifacts

IBM Integration Bus (IIB) and Enterprise Service Bus (ESB) artifacts are provided.

IBM Integration Bus archive files

IBM Integration Bus archive files are shown in the following table:

Table 85. IBM Integration Bus archive files

SI. Number	BAR Files	Description
1.	PMQMasterDataLoad	onboard Master Data information on to the PMQ Data mart.
2.	PMQEventDataLoad	Onboard and process event Data information on to the PMQ Event Store Integrate with SPSS scoring services (sensor health score and integrated health score) and process score results
3.	PMQMaintenance	Perform data preparations and invoke SPSS Maintenance Job as per the schedule
4.	PMQTopNFailure	Perform data preparations and invoke SPSS TopN Failure Job as per the schedule

Table 85. IBM Integration Bus archive files (continued)

SI. Number	BAR Files	Description
5.	PMQQEWSInspection	Prepares data and invokes the QEWS algorithm to perform inspection early warning analysis and loads the results back to the Profile data Mart of PMQ.
6.	PMQQEWSWarranty	Gathers data from the Service tables of PMQ data mart and passes as input to the QEWSL analysis and loads the results to the Profile data mart of PMQ.
7.	PMQMaximoIntegration	load master data and workorder from Maximo in PMQ and also supports creation/updating of Maximo workorders
8.	PMQQEWSIntegration	provide integration support to call Inspection and Warranty flows as per the required sequence or schedule and to invoke SPSS warranty stream
9.	PMQModelTraining	Invoke SPSS Job for training SPSS streams for sensor health score and integrated health score

Supported JAR files

Supported JAR files are shown in the following table:

Table 86. Supported JAR files

SI. Number	JAR/Properties/XML files	Description
1.	foundation-engine-api-1.5.0.0-SNAPSHOT.jar	APIs provided by Analytic Solution Foundation 1.5
2.	foundation-engine-core-1.5.0.0-SNAPSHOT.jar	Implementation jar of Analytics Solution Foundation 1.5
3.	org.apache.commons-collections-3.2.1.jar	This jar provides utility methods for most of the collection interfaces.
4.	commons-io-2.4.jar	This is a library of utilities to assist with developing IO functionality
5.	org.apache.commons-lang-2.4.jar	Provides a host of helper utilities for the java.lang API, notably String manipulation methods
6.	commons-pool-1.6.jar	This open source software library provides an object-pooling API and a number of object pool implementations.
7.	hamcrest-core-1.3.jar	Provides a library of matcher objects allowing 'match' rules to be defined declaratively, to be used in other frameworks.
8.	log4j-1.2.16.jar	Serves methods for logging purpose.
9.	icu4j.53.1.jar	Serves for internationalization
10.	pmq-foundation.jar	PMQ custom calculations over what is supported by Foundation
11.	ews.jar	The early warning system java module to analyze inspection and warranty usecases.

Supported property files and XML files

Supported property files and XML files are shown in the following table:

Table 87. Supported property files and XML files

Sl. No.	JAR / Properties / XML files
1	SetPerm.sh - Used to set 755 on the folder structure containing Warranty and Inspection charts
2	credentials.properties - Used to store SPSS credentials and joblocation urls
3	loc.properties - This is a properties file which maintains the location information of where the outputs to Warranty and Inspection is to be rendered.
4	log4j.properties - This is to set the logging levels and paths for the logs to be persisted.
5	orchestration_definition.xsd - foundation orchestration schema
6	solution_definition.xsd - foundation solution schema
7	<ul style="list-style-type: none"> • PMQ_orchestration_definition_inspection.xml • PMQ_orchestration_definition_maintenance.xml • PMQ_orchestration_definition_measurement.xml • PMQ_orchestration_definition_topnfailure.xml • PMQ_orchestration_definition_warranty.xml <p>These Foundation specific orchestration XML contains the orchestration mapping definitions to carry out the sequence of adapter calls to fulfill an operation. We have a separate XML for each of the use case/event type.</p>
8	PMQ_solution_definition.xml. This Foundation specific XML contains the table definitions and the relations to carry the DML and DDL operations.
13	<ul style="list-style-type: none"> • PMQEventLoad.properties • PMQMaintenance.properties • PMQMaximoIntegration.properties • PMQModelTraining.properties • PMQQEWSIntegration.properties • PMQTopNFailure.properties <p>These properties files will contain the webservice endpoint urls and are used to override the bar files with the correct endpoint urls as per customer needs</p>
14	Queues.txt - Contains all supporting queue definitions' and this is executed to create queues

Sample master data, event data, and QEWS data files

Sample master data files, event data files, and QEWS data files are provided.

The sample master data files are shown in the following list:

- language_upsert.csv
- tenant_upsert.csv
- event_code_upsert.csv
- event_type_upsert.csv
- group_dim_upsert.csv
- location_upsert.csv

- material_type_upsert.csv
- measurement_type_upsert.csv
- observation_lookup_upsert.csv
- process_upsert.csv
- product_upsert.csv
- profile_calculation_upsert.csv
- resource_type_upsert.csv
- source_system_upsert.csv
- supplier_upsert.csv
- value_type_upsert.csv
- material_upsert.csv
- production_batch_upsert.csv
- profile_variable_upsert.csv
- resource_upsert.csv

The sample event data files are shown in the following list:

- event_observation_maintenance_training.csv
- event_observation_maintenance_training_recommendation.csv
- event_observation_sensor_training.csv
- event_observation_process_material.csv
- event_observation_spc.csv
- event_observation_sensor.csv

The QEWS data files are shown in the following list:

- parameter_upsert.csv
- resource_production_batch_upsert.csv
- batchdata_inspection.csv
- event_observation_warranty.csv
- qewsrundate.txt

IBM SPSS artifacts

IBM SPSS streams and jobs are provided as artifacts.

Warranty - Streams and Jobs

Warranty artifacts are shown in the following table:

Table 88. Warranty - Streams and Jobs

.pes file	Modeler Stream / ADM Streams / CaDS jobs	Description
IBMPMQ_QEWSL	IBMPMQ_QEWSL_WARR.str	The manufacturing or production warranty stream built to do a ETL sort of processing. No modeling activity involved here

Table 88. Warranty - Streams and Jobs (continued)

.pes file	Modeler Stream / ADM Streams / CaDS jobs	Description
Not applicable	IBMPMQ_QEWSL_JOB	CaDS job used to invoke IBMPMQ_QEWSL_WARR.str for the manufacturing (MFG) or production (PROD) use cases
Not applicable	IBMPMQ_QEWSL_SALES.str	CaDS job used to invoke IBMPMQ_QEWSL_JOB for the sales (SALES) use case
Not applicable	IBMPMQ_QEWSL_ SALES_JOB	CaDS job used to invoke IBMPMQ_QEWSL_SALES.str for the SALES use cases

Maintenance - Stream and Jobs

Maintenance artifacts are shown in the following table:

Table 89. Maintenance - Stream and Jobs

.pes file	Modeler Stream / ADM Streams / CaDS jobs	Description
IBMPMQ_MAINTENANCE_ ANALYTICS	MAINTENANCE.str	Main stream in maintenance to identify and forecast the forecasted days to next maintenance and calculate the Maintenance Health score value .
Not applicable	MAINTENANCE_DAILY.str	Gives the Maintains details for a specific day
Not applicable	MAINTENANCE_ RECOMMENDATIONS.str	The ADM Stream to give the Maintenance recommendations
Not applicable	IBMPMQ_MAINTENANCE_ ANALYTICS_JOB	CaDS job used to invoke MAINTENANCE.str, MAINTENANCE_DAILY.str, MAINTENANCE_ RECOMMENDATIONS.str, and IBMPMQ_MAINTENANCE_ ANALYTICS_JOB

TopN failure predictors - Stream and Jobs

TopN failure predictors artifacts are shown in the following table:

Table 90. TopN failure predictors - Stream and Jobs

.pes file	Modeler Stream / ADM Streams / CaDS jobs	Description
IBMPMQ_TOP_FAILURE_PREDICTORS	TopN_MODEL.str	Modeling stream to extract and store the PMML giving the predictor importance of various configured parameters in predicting a resource's failure.
Not applicable	TopN_XML.str	This stream uses the PMML generated by the TopN_MODEL.str stream and extracts the necessary information from it and does essential transformation such that the output can be consumed by Cognos
Not applicable	IBMPMQ_TOP_FAILURE_PREDICTORS_JOB	CaDS job used to invoke the TopN_MODEL.str and the TopN_XML.str streams
Not applicable	TOPN_EVENTS.str	Create csv with of the Top N data in a format that can be loaded into the PMQ event table using the IIB flows

SENSOR-based health analytics - Stream and Jobs

SENSOR-based health analytics artifacts are shown in the following table:

Table 91. SENSOR-based health analytics - Stream and Jobs

.pes file	Modeler Stream / ADM Streams / CaDS jobs	Description
IBMPMQ_SENSOR_ANALYTICS	SENSOR_HEALTH_DATA_PREP.str	A Data preparation stream which retrieves the data from IBM PMQ tables and prepares the data to be used in the modeling, the eligible data is exported to a csv file for the modeling
Not applicable	SENSOR_HEALTH_COMBINED.str	The combined stream helps in training the models and also refreshes them for the scoring service
Not applicable	SENSOR_HEALTH_ANALYTICS_JOB	CaDS job used to invoke the SENSOR_HEALTH_COMBINED.str stream

Table 91. SENSOR-based health analytics - Stream and Jobs (continued)

.pes file	Modeler Stream / ADM Streams / CaDS jobs	Description
Not applicable	IBMPMQ_SENSOR_ ANALYTICS.str	This stream is auto generated when a training happens and for the real time scoring -- SENSOR_HEALTH_SCORE service configured to be used

Integrated analytics - Stream and Jobs

Integrated analytics artifacts are shown in the following table:

Table 92. Integrated analytics - Stream and Jobs

.pes file	Modeler Stream / ADM Streams / CaDS jobs	Description
IBMPMQ_INTEGRATED_ FEATURE_BASED_ ANALYTICS	INTEGRATION_FBA_ DATA_PREP.str	A Data preparation stream which retrieves the data from IBM PMQ tables and prepares the data to be used in the modeling, the eligible data is exported to a csv file for the modeling
Not applicable	INTEGRATION_FBA_ IHS_T.str	This stream helps in training the health score models and also refreshes them for the scoring service
Not applicable	INTEGRATION_FBA_ IFDM_T.str	This stream helps in training the forecasted days to maintenance model and also refreshes the model for the scoring service
Not applicable	IBMPMQ_INTEGRATED_ FEATURE_BASED ANALYTICS	CaDS job used to invoke the INTEGRATION_FBA_ DATA_PREP.str , INTEGRATION_ FBA_IHS_T.str, and INTEGRATION_FBA_IFDM_ T.str streams
Not applicable	INTEGRATION_ FBA_IHS.str	This stream is auto generated when a training happens, and is invoked for real time health score prediction within the INTEGRATED_FBA.str stream

Table 92. Integrated analytics - Stream and Jobs (continued)

.pes file	Modeler Stream / ADM Streams / CaDS jobs	Description
Not applicable	INTEGRATION_FBA _IFDM.str	This stream is auto generated when a training happens, and is invoked for real time forecasted days to maintenance prediction within the INTEGRATED_FBA.str stream
Not applicable	INTEGRATED_FBA.str	This is ADM published stream which invokes the health score and days to maintenance models, and gives a final recommendation based on business rules. It is configured for real time scoring - INTEGRATED_FBA service is configured for scoring services.

Feature-based analytics - Stream and Jobs

Feature-based analytics artifacts are shown in the following table:

Table 93. Feature-based analytics - Stream and Jobs

.pes file	Modeler Stream / ADM Streams / CaDS jobs	Description
IBMPMQ_SENSOR_ FEATURE_BASED ANALYTICS	SENSOR_FBA_DATA_ PREP.str	A Data preparation stream which retrieves the data from IBM PMQ tables and prepares the data to be used in the modeling, the eligible data is exported to a csv file for the modeling
Not applicable	SENSOR_FBA_FHS_T.str	This stream helps in training the health score models and also refreshes them for the scoring service
Not applicable	SENSOR_FBA_FFDM_T.str	This stream helps in training the forecasted days to maintenance model and also refreshes the model for the scoring service
Not applicable	IBMPMQ_SENSOR_ FEATURE_BASED_ ANALYTICS	CaDS job used to invoke the SENSOR_FBA_DATA_ PREP.str, SENSOR_FBA_ FHS_T.str, and SENSOR_FBA_ FFDM_T.str streams

Table 93. Feature-based analytics - Stream and Jobs (continued)

.pes file	Modeler Stream / ADM Streams / CaDS jobs	Description
Not applicable	SENSOR_ FBA_FHS.str	This stream is auto generated when a training happens, and is invoked for real time health score prediction within the INTEGRATED_FBA.str stream
Not applicable	SENSOR_FBA_FFDM.str	This stream is auto generated when a training happens, and is invoked for real time forecasted days to maintenance prediction within the FBA.str stream
Not applicable	FBA.str	This is ADM published stream which invokes the health score and days to maintenance models, and gives a final recommendation based on business rules. It is configured for real time scoring - FBA service configured to be used.

Features for the energy and utility industry - Stream and Jobs

Energy and utility industry artifacts are shown in the following table:

Table 94. Features for the energy and utilities industry - Stream and Jobs

.pes file	Modeler Stream / ADM Streams / CaDS jobs	Description
IBMPMQ_ ARMOR_ANALYTICS	IBMPMQ_ARMOR_ DTCA_CABLE_ FEATURES	CaDS job used to invoke the IBMPMQ_FEATURES_ DTCA.str and IBMPMQ_FEATURES_ CABLE.str to generate features for distribution transformer and cable assets respectively.
Not applicable	IBMPMQ_FEATURES _DTCA.str	This stream helps to generate the csv file with Distribution Transformer specific overload features, as well as current aging measure. The csv file is used in Feature-Based Analytics modeling.
Not applicable	IBMPMQ_FEATURES _CABLE.str	This stream helps to generate the csv file with Cable specific overload features, which is used in FBA modeling.

Table 94. Features for the energy and utilities industry - Stream and Jobs (continued)

.pes file	Modeler Stream / ADM Streams / CaDS jobs	Description
Not applicable	IBMPMQ_ARMOR _DTPA	CaDS job used to invoke the IBMPMQ_FEATURES _DTPA.str stream to generate projected aging for distribution transformer assets.
Not applicable	IBMPMQ_FEATURES _DTPA.str	This stream helps to generate the csv file with distribution transformer projected aging for user-specified future years (based on start year, number of years, and interval size), and degradation factor.

IBM Cognos Business Intelligence Artifacts

An IBM Framework Manager model, and a compressed file that contains reports and dashboards is provided.

Framework Manager model

The Framework Manager model is described in the following table:

Table 95. Framework Manager model

SI. Number	FM Model	Purpose
1.	IBMPMQ	<p>IBM Predictive Maintenance and Quality uses IBM Cognos Framework Manager to model the metadata for reports. IBM Cognos Framework Manager is a metadata modeling tool that drives query generation for IBM Cognos software.</p> <p>A model is a collection of metadata that includes physical information and business information for one or more data sources. IBM Cognos software enables performance management on normalized and denormalized relational data sources and a variety of OLAP data sources.</p>

Site Overview dashboard

The Site Overview dashboard is described in the following table:

Table 96. Site Overview dashboard

Sl. Number	Report/Dashboard	Purpose
1.	Overview	<p>Provides a high-level summary of the health of all of your assets at all sites, it shows the key performance indicators (KPIs) with the greatest impact.</p> <p>You can change the detail that is displayed by selecting items from the list boxes. For example, you can change the date and the equipment type.</p>
2.	Top 10 Contributors	Identifies the equipment, locations, and operators responsible for the most failures.
3.	KPI Trending	<p>You can select multiple key performance indicators (KPIs) to be plotted side-by-side in a line chart.</p> <p>You can identify correlations between the KPIs and see whether there is any lagging behavior.</p> <p>For example, if there is a spike in one KPI, how long does it take to impact the other KPIs?</p>
4.	Actual vs Plan	<p>You can monitor how closely the metrics track against the plan.</p> <p>Variances are highlighted.</p>
5.	Equipment Listing	<p>The health score for a site is derived from the lower-level scores from each piece of equipment in the site.</p> <p>This report shows you all the pieces of equipment on the site and the health scores and relevant KPIs for that equipment.</p>
6.	Equipment Outliers	Lists the equipment (or assets) that are performing outside of allowable limits. The measures that are shown differ depending on the equipment, but examples are operating temperature, lateral strain, hydraulic pressure, average value, last value, and control limits.
7.	Recommended actions	A summary of all recommended actions for each piece of equipment, for the health score measurement.

Equipment Reports dashboard

The Equipment Reports dashboard is described in the following table:

Table 97. Equipment Reports dashboard

Sl. Number	Report/Dashboard	Purpose
1.	Equipment Profile	A detailed report that shows everything that is known about a piece of equipment: how it is performing today and how it performed in the past.
2.	Equipment Control Chart	Shows the upper and lower control limits and the average limits for selected measures.
3.	Equipment Run Chart	Shows the measures for a particular piece of equipment.
4.	Equipment Outliers	Shows detailed measures for a piece of equipment that shows anomalies.
5.	Event Type History	Lists the events for a device.

Product Quality dashboard

The Product Quality dashboard is described in the following table:

Table 98. Product Quality dashboard

Sl. Number	Report/Dashboard	Purpose
1.	Defect Analysis	Shows product defects and inspection rates.
2.	Inspection Rate Analysis	Examines the relationship between inspections and defects over time to find the optimal rate of inspection.
3.	Material Usage By Process	Provides an overview of material usage in the production processes.

SPC reports

SPC reports are described in the following table:

Table 99. SPC reports

Sl. Number	Report/Dashboard	Purpose
1.	SPC - Histogram	This report allows a visual interpretation of data by indicating the number of data points (events) that lie within a range of values, called a class or a bin. The frequency of the data that falls in each bin is depicted by the use of a bar.

Table 99. SPC reports (continued)

Sl. Number	Report/Dashboard	Purpose
2.	SPC - X Bar and S / R Charts	To track instantaneous variations and to evaluate the stability of the variability with in the process for smaller sample sizes (R Chart) and for bigger sample sizes (S Chart)

Other reports

Other reports are described in the following table:

Table 100. Other reports

Sl. Number	Reports/Dashboard	Purpose
1.	Advance KPI Trend Report	This chart compares multiple key performance indicators (KPIs) across multiple resources. You can use this chart to analyze variations in a resource against a set of profiles.
2.	Material Usage by Production Batch	This report provides an overview of material usage by production batch. By correlating production batches with defects to material usage by production batch, you can begin to trace the impact of defective materials.
3.	Audit Report	Shows the counts of rows in the major master data tables.

Drill Through reports from the Audit report

The following table lists Drill Through reports from the Audit report.

Table 101. Drill Through reports from the Audit report

Sl. Number	Reports/Dashboard	Purpose
1.	Resource List	Lists the resources by resource type.
2.	Profile Variables	Lists all measures and key performance indicators that are being tracked in daily profiles and historical snapshots.
3.	Process List	Lists all production processes.
4.	Material List	Lists materials that are used in the production process.
5.	Production Batch List	Lists production batches.

Table 101. Drill Through reports from the Audit report (continued)

Sl. Number	Reports/Dashboard	Purpose
6.	Material Usage By Production Batch	This report provides an overview of material usage by production batch. By correlating production batches with defects to material usage by production batch, the impact of defective materials can begin to be traced.
7.	Measurement Type List	Lists measurement types. For each measurement type, the report shows unit of measure and aggregation type.

Maintenance dashboard and Top N Failure reports

The Maintenance dashboard and Top N Failure reports are described in the following table:

Table 102. Maintenance dashboard and Top N Failure reports

Sl. Number	Reports/Dashboard	Purpose
1.	Maintenance Overview Dashboard	This dashboard provides an overview of health score for the last current day in the record. Along with maintenance health score, report also shows a comparative view with sensor health score and integrated health score.
2.	Maintenance Advance Sorting Report	This chart displays the same measures as the main report (Maintenance Overview Dashboard) in a tabular format. Users can sort on a column by clicking the column header.
3.	Maintenance Health Score and Failure Detail Report	This report will help user to see the historical and forecasted health scores of a machine along with Historical Breakdown, Forecasted Breakdown, Planned Maintenance Schedules.
4.	Top N Failure Report	The plot shows the UNSIGNED predictor importance, indicating the absolute importance of any predictor in predicting a failure or non-failure condition.

QEWS quality dashboards and reports

QEWS quality dashboards and reports are described in the following table:

Table 103. Inspection and Warranty reports

Sl. Number	Reports/Dashboard	Purpose
1.	Quality dashboard - Inspection	This dashboard provides an overview of the state of products at a selected run date.
2.	Quality dashboard - Inspection Detail History	This dashboard provides details about the state of products and the various threshold values for a selected product category at a selected run date.
3.	QEWS - Inspection Chart	This chart reports the failure rates and CUSUM values for a specific product type and product code over a time period.
4.	Quality dashboard - Warranty	This dashboard provides an overview of the state of products at a selected run date.
5.	Quality dashboard - Warranty Detail History	This dashboard provides details about the state of products and the various threshold values for a selected product category at a selected run date.
6.	QEWSL - Warranty Chart	This chart reports the replacement rates for a specific product type and product code over a time period.
7.	Quality dashboard - Parametric	This dashboard provides an overview of the state of products at a selected run date for a variable.
8.	Quality dashboard - Parametric Detail History	This dashboard provides details about the state of products and the various threshold values for a selected product category, at a selected run date for a variable.
9.	QEWSV - Parametric chart	<p>This report is used for monitoring variable-type data and CUSUM values that are obtained from the QEWSV batch along with threshold levels.</p> <p>The report is designed to support five different validation types: Material Validation, Process-Resource Validation, Production Batch Validation, Resource Health Check, and Location Suitability.</p>

Appendix F. Troubleshooting

Troubleshooting is a systematic approach to solving a problem. The goal of troubleshooting is to determine why something does not work as expected and how to resolve the problem.

Review the following table to help you or customer support resolve a problem.

Table 104. Troubleshooting actions and descriptions

Actions	Description
A product fix might be available to resolve your problem.	Apply all known fix packs, or service levels, or program temporary fixes (PTF).
Look up error messages by selecting the product from the IBM Support Portal, and then typing the error message code into the Search support box (http://www.ibm.com/support/entry/portal/).	Error messages give important information to help you identify the component that is causing the problem.
Reproduce the problem to ensure that it is not just a simple error.	If samples are available with the product, you might try to reproduce the problem by using the sample data.
Ensure that the installation successfully finished.	The installation location must contain the appropriate file structure and the file permissions. For example, if the product requires write access to log files, ensure that the directory has the correct permission.
Review all relevant documentation, including release notes, technotes, and proven practices documentation.	Search the IBM knowledge bases to determine whether your problem is known, has a workaround, or if it is already resolved and documented.
Review recent changes in your computing environment.	Sometimes installing new software might cause compatibility issues.

If the items on the checklist did not guide you to a resolution, you might need to collect diagnostic data. This data is necessary for an IBM technical-support representative to effectively troubleshoot and assist you in resolving the problem. You can also collect diagnostic data and analyze it yourself.

Troubleshooting resources

Troubleshooting resources are sources of information that can help you resolve a problem that you are having with an IBM product.

Support Portal

The IBM Support Portal is a unified, centralized view of all technical support tools and information for all IBM systems, software, and services.

The IBM Support Portal lets you access all the IBM support resources from one place. You can tailor the pages to focus on the information and resources that you need for problem prevention and faster problem resolution.

Find the content that you need by selecting your products from the IBM Support Portal (<http://www.ibm.com/support/entry/portal/>).

Gathering information

Before contacting IBM Support, you will need to collect diagnostic data (system information, symptoms, log files, traces, and so on) that is required to resolve a problem. Gathering this information will help to familiarize you with the troubleshooting process and save you time

Service requests

Service requests are also known as Problem Management Reports (PMRs). Several methods exist to submit diagnostic information to IBM Software Technical Support.

To open a PMR or to exchange information with technical support, view the IBM Software Support Exchanging information with Technical Support page (<http://www.ibm.com/software/support/exchangeinfo.html>).

Fix Central

Fix Central provides fixes and updates for your system's software, hardware, and operating system.

Use the pull-down menu to navigate to your product fixes on Fix Central (<http://www-947.ibm.com/systems/support/fixes/en/fixcentral/help/getstarted.html>). You may also want to view Fix Central help.

Knowledge bases

You can find solutions to problems by searching IBM knowledge bases.

You can use the IBM masthead search by typing your search string into the Search field at the top of any [ibm.com](http://www.ibm.com) page.

IBM Redbooks

IBM Redbooks® are developed and published by IBM's International Technical Support Organization, the ITSO.

IBM Redbooks (<http://www.redbooks.ibm.com/>) provide in-depth guidance about such topics as installation and configuration and solution implementation.

IBM developerWorks

IBM developerWorks provides verified technical information in specific technology environments.

As a troubleshooting resource, developerWorks provides easy access to the top ten most popular practices for Business analytics, in addition to videos and other information: developerWorks for Business analytics (<http://www.ibm.com/developerworks/analytics/practices.html>).

Software support and RSS feeds

IBM Software Support RSS feeds are a quick, easy, and lightweight format for monitoring new content added to websites.

After you download an RSS reader or browser plug-in, you can subscribe to IBM product feeds at IBM Software Support RSS feeds (<https://www.ibm.com/software/support/rss/>).

Log files

Log files can help you troubleshoot problems by recording the activities that take place when you work with a product.

IBM Integration Bus log files

Errors that occur within IBM Integration Bus message flows are written to error logs in the following folder: /error. The location of this folder is determined by the **MQSI_FILENODES_ROOT_DIRECTORY** environment variable during the installation process.

Errors for message flows are as follows:

Master data flows

Rejected records are written to *input_filename_error.csv*

Errors are logged in *input_filename_error.txt*

Event flow - MultiRowEventLoad

Rejected records are written to *input_filename_error.csv*

Errors are logged in *input_filename_error.txt*

Event flow - StdEventLoad

Failed event messages are written to the error queue `PMQ.EVENT.ERROR`

Errors are logged in *EventError.txt*

PMQIntegration flow

Failed event request and web service fault messages are written to the error queue: `PMQ.INTEGRATION.ERROR`

Errors are logged in *IntegrationError.txt*

Maximo flow - Maximomasterdataasset, Maximomasterdataclassification, Maximomasterdatalocation

Rejected records are written to *input_filename_error.xml*

Errors are logged in *input_filename_error.txt*

Maximo flow - WorkorderCreation

Failed Maximo requests and web service fault message are written to the error queue: `PMQ.MAXIMO.ERROR`

Log files generated during the installation process

Errors that occur during the prerequisite checks that happen during the installation process are written to the following location on the node where installation is taking place:

```
/var/IBMPMQ/PreReq.log
```

The following errors can be reported:

Error, Can't proceed as the user is a Non Root User

The installer must be run as a Root user.

Error, <package_name> not installed

Install the package using the following command:

```
# rpm -i software-2.3.4.rpm
```

Error <MEM> is less than the required 8GB memory

Ensure that there is 8 GB of memory available.

Error <TMP> KB is available for TMP, need 100GB

Error <File System Size in KB> KB is available for /opt, need 100GB

The /opt filesystem must have a minimum of 100 GB space for installation.

Error / filesystem requires more than 150 GB of freespace

Ensure that the file system has at least 150 GB available.

Error <Version information> is not supported for IBMPMQ

Uninstall the current DB2 version, and ensure that the system is clean.

Error, Port <portno> is not open

Ensure that the port is open to the firewall, if used.

Error, Connection to <SERVER> on port <PORT> failed

Ensure that the port is open to the firewall, if used.

Unresolved library error on import of PMQDancingCharts or PMQMasterDDLGenerator

PMQDancingCharts and PMQMasterDDLGenerator are applications that are provided with Predictive Maintenance and Quality. When you import these applications into the IBM Integration Toolkit, you get an error.

Symptoms

You see an error message similar to the following example:

```
Project 'PMQDancingChartsJava' is missing required library: 'C:\ProgramData\IBM\
MQSI\shared-classes\junit-4.11.jar'
```

Causes

This issue is caused by references to some unnecessary Java library files in PMQDancingCharts and PMQMasterDDLGenerator application projects.

Diagnosing the problem

Import the project interchange files for PMQDancingCharts and PMQMasterDDLGenerator applications into the IBM Integration Toolkit. Change to the **Problems** view of the IBM Integration Toolkit to see the list of missing library references.

Resolving the problem

To resolve the problem, remove the unnecessary Java library references from the build path of the applications.

1. Change to the Java perspective of the IBM Integration Toolkit.
2. Right-click the **PMQDancingChartsJava** or the **PMQMasterDDLGeneratorJava** project and select **Build Path > Configure Build Path > Java Build Path > Libraries**.
3. The unnecessary Java library references are marked by a red X. Remove these references and click **OK**.
4. Rebuild the project.

SPSS job invocation from the orchestration message flow fails

Operating system or application user credentials have changed, which causes the credentials on IBM Predictive Maintenance and Quality subcomponents to stop working. For example, if the user credentials on the Predictive Maintenance and Quality Analytics node change, you must also change the user credentials that are configured to start SPSS jobs on the Integration Bus node.

Symptoms

The orchestration message flow that calls a job in the event orchestration adapter fails. There is an SPSS credentials error reported in the StdEventLoad_Error and foundation log files.

Resolving the problem

To resolve the problem, you must update the credentials on the Integration Bus node.

1. Log in to the Integration Bus node computer as the **mqm** user.
2. To update both the user ID and password, type the following command:

```
mqsichangeproperties pmqbroker -c UserDefined -o SPSS -n  
UserName,Password -v SPSS_UserID,SPSS_Password
```

Where *SPSS_UserID* is the new user ID and *SPSS_Password* is the new password.
3. To update only the password, type the following command:

```
mqsichangeproperties pmqbroker -c UserDefined -o SPSS -n Password -v  
SPSS_Password
```

Where *SPSS_Password* is the new password.
4. Restart the broker on the Integration Bus node. Type the following commands:

```
mqsisstop broker_name  
mqsisstart broker_name
```

Where *broker_name* is the broker name.

Performance tuning guidelines

You can tune the performance of your IBM Predictive Maintenance and Quality environment.

Deadlock errors happen when parallel processing is enabled

Deadlock errors in IBM Predictive Maintenance and Quality typically happen when parallel processing is enabled by increasing extra instances, and all messages are routed to single folders and queues.

About this task

The error message is named `EventError.txt` and is found in the `\error` folder in the IBM Integration Bus node, location that is defined by the **MQSI_FILENODES_ROOT_DIRECTORY** environment variable.

The error message is as follows:

```
"Error:Label:StdEventLoad_1.LoadEvent:TransactionId:fbcb6b4c0-b434-11e2-8336  
-09762ee50000TransactionTime:2013-05-04 02:34:022322:Child SQL exception:[unixODBC]  
[IBM][CLI Driver][DB2/LINUX8664] SQL0911N The current transaction has been rolled  
back because of a deadlock or timeout. Reason code "2". SQLSTATE=40001"
```

For more information, see “Parallel processing” on page 61.

Procedure

1. Connect to the database with the following command: `db2 connect to db <dbname> [IBMPPMQ]>`
2. Set the isolation level to RR with the following command: `db2 set isolation level to RR`
3. Check the value of the dead lock check time setting with the following command: `db2 get db cfg |grep DL`

The suggested values are:

Interval for checking deadlock (ms)

(DLCHKTIME) = 20000

Deadlock events

(MON_DEADLOCK) = WITHOUT_HIST

4. If the value for the **DLCHKTIME** property is less than 2000, then set the value with the following command: `db2 update db cfg for <dbname> using DLCHKTIME 20000 immediate`
5. Check the value of Lock list and percentage of Locks that are allowed per application `db2 get db cfg |grep LOCK`

The suggested values are:

Maximum storage for lock list (4 KB)

(LOCKLIST) = 100000

Percentage of lock lists per application

(MAXLOCKS) = 97

Lock timeout (sec)

(LOCKTIMEOUT) = -1

Block non logged operations

(BLOCKNONLOGGED) = NO

Lock timeout events

(MON_LOCKTIMEOUT) = NONE

Deadlock events

(MON_DEADLOCK) = WITHOUT_HIST

Lock wait events

(MON_LOCKWAIT) = NONE

6. If the value for the **LOCKLIST** property is less than 1000, then set the value with the following command: `db2 update db cfg for <dbname> using LOCKLIST 100000 immediate`
7. If the value for the **MAXLOCKS** property is less than 97, then set the value with the following command: `db2 update db cfg for <dbname> using MAXLOCKS 97 immediate`

Event processing performance

There are two approaches for increasing the performance of event processing. Events can be processed in multiple threads and events can be processed as a batch.

The event processing flow `StdEventLoad` processes messages that contain a single event or that contain a collection of events. The flow `MultiRowEventLoad` is an example of a flow that loads events and sends them for processing as a collection.

Processing events as collections has the best performance improvement when the events in the collection update the same profile rows. Sort the events so that similar events are processed together. For example, sorting them by device, time, and measurement.

Events that are processed as a collection can be processed only by a single thread. The exception is when the collections that are processed in separate threads do not update any of the same profile rows.

Processing single events by using multiple threads improves performance when the events are updating different profile rows. If the events are all updating the same profile rows, then there is little advantage in using multiple threads. A thread locks the profile rows that it is updating and the other threads must wait until the lock is released. The lock is released when the transaction is committed.

Calculations that are identified as `is_increment` also have improved performance because they can update a profile row in the database without first having to retrieve it and lock it.

Troubleshooting reports

Reports in IBM Predictive Maintenance and Quality are created in IBM Cognos Report Studio. You may encounter problems when using some of the reports included with IBM Predictive Maintenance and Quality.

For further information about troubleshooting reports, see the *IBM Cognos Business Intelligence Troubleshooting Guide*, and the *IBM Cognos Report Studio User Guide*. These documents are available at IBM Cognos Business Intelligence Knowledge Center (<http://www.ibm.com/support/knowledgecenter/SSEP7J>).

Audit Report fails with error DMB-ECB-0088 A DMB cube build limit has been exceeded

This error can occur in any report when the master table contains more than 1 million resources but occurs most commonly in the Audit Report.

About this task

To fix the problem, you must increase the **MaxCacheSize** and **MaxNumberOfRecordRows** parameter values in the `qfs_config.xml` file.

Procedure

1. Go to the following IBM Cognos Business Intelligence configuration folder path: `/opt/ibm/cognos/analytics/configuration`.
2. Open the `qfs_config.xml` file and increase the value of the following parameters:
 - `MaxCacheSize`
 - `MaxNumberOfRecordRows`
3. Save the `qfs_config.xml` file and run the report.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those

websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample

programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year).

Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. _enter the year or years_.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's name, user name, password, or other personally identifiable information for purposes of session management, authentication, single sign-on configuration or other usage tracking or functional purposes. These cookies can be disabled, but disabling them will also likely eliminate the functionality they enable.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at <http://www.ibm.com/privacy> and IBM's Online Privacy Statement at <http://www.ibm.com/privacy/details> in the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at <http://www.ibm.com/software/info/product-privacy>.

Index

A

actual values 60
Actual vs plan report 207
Advanced KPI Trend Chart 220
aggregation_type 256, 257
API 21, 241
asset management and manufacturing
 execution systems integration 9
Audit Report 215, 307

B

batch processing 60
batch processing events 60
batch_batch 242
benefits 72, 84, 94
business challenges 66, 74, 86

C

calculations 57
calculations, custom 59
carry_forward_indicator 256, 257
city_name 245
Cognos BI artifacts 294
company 26, 36
comparison_string 257
compatibility query mode
 using to see real-time data 284
compatible query mode 283
configure 7
configuring Maximo for OutBound work
 orders using a web service 38
configuring Maximo for OutBound work
 orders using an XML file 40
configuring solution.xml for event
 flow 62
country_cd 245
country_name 245
creating a workorder 45
custom applications 9

D

dashboard application 9
dashboards 203
 creating 10
data exports Master Data
 Management 28
data model 285
data_type 257
defect analysis 212
defect summary report 212
Defects by event code 212
Defects by location 212
Defects by production batch 212
defects vs inspection rate line chart 214
DMB-ECB-0088 307
drill through reports 215

E

enabling master data loading in realtime
 mode 36
environment variables for MDM 25
Equipment control chart 210
Equipment listing report 208
Equipment outliers 211
Equipment profile report 209
Equipment reports 209
Equipment Reports 203
Equipment run chart 211
error messages 303
error reporting 53
event data, configuring 49
event definition 50
event file, sample 287
event format 53
event of type count 57
event processing 49, 60
event type history report 212
event_code 243
event_code_indicator 256
event_code_set 243
event_code_set_name 243
event_type 255

F

feature-based analytics 185
 data preparation 190
 deployment of model 192
 training job 189
Feature-based predictive model
 input data for training 188
 minimum data requirements 188
 resource sub type level modeling 189
file format 22
file location 22
Fix Central 302
flat file API 241
flat file event input 51
forecast values 60
Framework Manager model database
 layer 267
Framework Manager model
 description 267
Framework manager model dimensional
 layer 283
Framework Manager model logical
 layer 282
Framework manager model security 283

G

generic batch orchestration 16
group_dim 32
group_type_cd 244
group_type_name 244

H

Health score contributors 204
Health score trend 204
high_value_date 257
high_value_number 257

I

IBM Integration Bus 49
IBM Redbooks 302
import metadata into MDM 29
Incident/recommendation analysis 204
InfoSphere MDM Collaboration
 Server 21, 26
Integrated analytics
 deployment of model 198
 training job 195
Integrated analytics predictive model
 input data for training 194
 orchestration rules 196
Integrated predictive model
 minimum data requirements 194
 resource sub type level modeling 194
Integration analytics
 data preparation 195
interval calculation 57
IS_ACTIVE 241

K

knowledge bases 302
KPI table 53
KPI trending report 207
kpi_indicator 257
KPIs 49, 208

L

language 244
last date of event type 57
Last date of measurement in range 57
last date of measurement type 57
latitude 245
location 32, 245
location example 22
location_name 245
log files 303
longitude 245
low_value_date 257
low_value_number 257

M

Maintenance Advanced Sorting
 chart 216
maintenance analytics 172, 181
maintenance analytics data 172
Maintenance Health and Failure Detail
 report 216

- Maintenance Overview Report 216
- master data 21, 241
- Master Data Management 24
- master files, sample 287
- material usage by process crosstab 215
- Material usage by production batch 216
- material_cd 246
- material_name 246
- material_type_cd 246, 247, 257
- material_type_name 247
- Maximo 32, 37, 202
- Maximo Asset Management 9
- MDM company archive file 285
- MDM guidelines 27
- measurement above limit 57
- measurement below limit 57
- measurement data 49
- measurement delta 57
- Measurement in range count 57
- measurement of type 57
- measurement of type count 57
- measurement text contains count 57
- measurement_type 256
- measurement_type_cd 257
- message flows 13
- metadata 255
- model 250
- modeling 173, 174
- modify a process 22
- modify a resource 22

O

- operator_cd 250
- orchestration 13
 - generic batch 16
- Outliers 208

P

- parallel processing 60
- parametric overview 86
- parent_process_cd 247
- parent_resource_serial_no 250
- planned values 60
- pre-modeling data 173
- predictive models 171
- predictive scores 60
- predictive scoring 59
- Problem Management Reports
 - logging 302
 - PMR
 - See Problem Management Reports
- process_cd 247
- process_indicator 257
- process_kpi 54
- process_name 247
- process_profile 56
- Product quality dashboard 212
- product_cd 248, 249
- product_name 248
- production_batch_cd 242, 249
- production_batch_name 249

- profile 56
- profile calculations 57
- profile table 53
- profile_calculation 249
- profile_calculation_cd 257
- profile_indicator 257
- profile_units 257
- profile_variable 53
- profiles 49

Q

- QEWS - Inspection Chart 222
- QEWS quality dashboard - inspection 221
- QEWS quality dashboard - inspection detail history 222
- QEWS quality dashboard - parametric 225
- QEWS quality dashboard - parametric detail history 226
- QEWS quality dashboard - warranty 223
- QEWS quality dashboard - warranty detail history 224
- QEWS quality dashboards 221
- QEWS use case
 - parametric 84
- QEWSC parametric chart 226
- QEWSL - Warranty Chart 224
- quality inspection overview 66
- query modes 283
- queue 60

R

- real-time data 284
- recommendations 32, 60, 201
- Recommended actions report 209
- region_cd 245
- region_name 245
- related_production_batch_cd 242
- remove events 62
- remove master data 260
- resource 32
- resource example 22
- resource_kpi 54
- resource_name 250
- resource_profile 56
- resource_sub_type 250
- resource_type_cd 250, 252, 257
- resource_type_name 252
- results 72, 84, 94
- rules 201

S

- schema definition for events 53
- scoring 59
- scoring, preventing 202
- sensor health analytics 181
- Sensor Health analytics 177

- Sensor Health predictive model 177
- serial_no 250
- service requests
 - PMR 302
- site overview dashboard 204
- Site Overview Dashboard 203
- software support and RSS feeds 302
- source_system_cd 252
- SPC - Histogram 219
- SPC - X Bar R/S Chart 220
- SPSSTRIGGER 202
- state_province_cd 245
- state_province_name 245
- statistical process control 219
- supplier_cd 253
- supplier_name 253
- supply_cd 246
- Support Portal 301

T

- technical challenges 66, 74, 86
- tenant 254
- threads 60
- Top 10 Contributors dashboard 206
- Top N Failure Analysis Report 216
- TopN Failure Analysis Report 227
- troubleshooting
 - getting fixes 302
 - IBM Redbooks 302
 - identifying problems 301
 - MustGather information 302
 - proven practices documentation 302
 - reports 307
 - software support and RSS feeds 302
 - Support Portal 301
- troubleshooting resources 301

U

- unit_of_measure 256
- updating recommendations 43
- upsert 241
- use case
 - quality inspection 65
 - warranty 72

V

- value_type_cd 255
- value_type_name 255
- video documentation
 - YouTube 302
- viewing recommendations 44

W

- warranty overview 74
- work order creation, disabling 202
- work order service 37
- work orders 32