



Data Collection Guide

Before using this information, be sure to read the general information under the “Notices” section on page 116.

This edition applies to **VERSION 4.0, Rational Dashboard** and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright 2004, 2009**

U.S. Government Users Restricted Rights - Use, duplication, or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

## TABLE OF CONTENTS

### Table of Contents

<b>1. Introduction</b> .....	<b>5</b>
1.1. Overview.....	5
1.2. What is Measurement? .....	5
1.3. IBM Rational Dashboard High Level Features .....	6
<b>2. Terminology</b> .....	<b>7</b>
<b>3. Overview of the Collection Process</b> .....	<b>13</b>
3.1. The Big Picture.....	13
3.1.1. Control Flow in the Collection Process.....	15
3.1.2. Data Flow in the Collection Process .....	16
3.2. Components of Collection .....	17
3.2.1. Interfaces – What To Collect and How To Process.....	17
3.2.2. Databases – Where to Store Data .....	26
3.2.3. Schedules - When to Collect.....	27
3.3. Item List, Item Details and Units .....	28
3.3.1. Overview of Items Concept .....	28
3.3.2. The Item List .....	29
3.3.3. Item Data .....	32
3.4. Overview of Queries and Graphs .....	34
3.5. Writing Queries .....	34
3.5.1. Sample Queries.....	35
3.5.2. Query Tags.....	36
3.5.3. Using Multi-Series Queries .....	38
3.5.4. Setting Up a Snapshot Grid to Show Tabular Data.....	46
3.5.5. Understanding the !FIELD! Tag in the Collector.....	51
3.5.6. Understanding the !SETTAG! in the Collector .....	52
<b>4. Collector Reference and Commands</b> .....	<b>58</b>
4.1. Menu Command Reference .....	58
4.1.1. File Menu .....	58
4.1.2. Collection Menu .....	60
4.1.3. Sources Menu .....	60
4.1.4. Tools Menu .....	62
4.2. Command Line Reference .....	62
4.2.1. Command Line Parameters .....	63
4.2.2. Collecting All Sources for Current Date .....	64
4.2.3. Collecting One Source for a Past Date.....	64
4.2.4. Collecting One Source for Multiple Dates in the Past .....	65
4.3. Technical References.....	65
4.3.1. Collector List File Format.....	66

4.3.2.	Collector Connection List Format .....	66
4.3.3.	Replacement Tags in GetDetails Query .....	68
4.3.4.	Date-Stamped File Names .....	69
<b>5.</b>	<b>Collector Operations .....</b>	<b>71</b>
5.1.	Configuring the Web Services .....	71
5.2.	Running a Collection .....	71
5.3.	Testing a Data Source .....	74
5.4.	Populating a Basic Unit from The Library .....	76
5.5.	Configuring and Editing the Organizational Tree .....	77
5.6.	Configuring Data Sources .....	79
5.6.1.	Pre-Requisites for Data Sources .....	79
5.6.2.	Configuring a CSV Source .....	80
5.6.3.	Configuring an HP Quality Center Source .....	82
5.6.4.	Configuring an IBM Rational ClearQuest Source.....	82
5.6.5.	Configuring a Microsoft Access Source .....	84
5.6.6.	Configuring a Microsoft Excel Source.....	85
5.6.7.	Configuring a Microsoft Project Desktop 2003 Source .....	87
5.6.8.	Configuring a Microsoft Project Server 2003 Source .....	88
5.6.9.	Configuring a Microsoft Project Server 2007 Source .....	89
5.6.10.	Configuring a Multi-Source SQL Source .....	89
5.6.11.	Configuring an ODBC Database Source .....	90
5.6.12.	Configuring an Oracle SQL Database Source.....	90
5.6.13.	Configuring a IBM Rational Change Source.....	91
5.6.14.	Configuring a IBM Rational DOORS Source .....	92
5.6.15.	Configuring a IBM Rational Synergy Source .....	94
5.6.16.	Configuring an XML ADO.NET Source.....	95
<b>6.</b>	<b>Portal Operations.....</b>	<b>97</b>
6.1.	Define a Unit.....	98
6.2.	Set the Default Schedule for the Unit .....	100
6.3.	Assign Collected Items.....	102
6.3.1.	Assign Sub-Tab with By Date Options .....	102
6.3.2.	Assign Sub-Tab with By Source Options.....	103
6.4.	Assign Information Needs to your Unit.....	106
6.5.	Refresh Data within the Unit .....	109
6.6.	See Your Graphs .....	111
<b>7.</b>	<b>Other Resources .....</b>	<b>112</b>
<b>8.</b>	<b>Contact Information.....</b>	<b>113</b>
	<b>Notices .....</b>	<b>117</b>

# 1. Introduction

This document provides an overview of data collection using IBM Rational Dashboard.

## 1.1. *Overview*

Today's organizations strive to meet the demands of their market while delivering quality products in a timely manner. To achieve this end, managers are required to review an overwhelming and often conflicting volume of reports to meet deadlines. Without adequate plans and the ability to evaluate the performance of a program or project against that plan, it would be increasingly difficult to make proper adjustments, learn from previous mistakes, and ensure that the project meets its ultimate goal and achieve business success.

IBM Rational Dashboard is a performance measurement application, accessible via a web browser. It allows executives, software managers, and project managers to keep track of the development of their applications, their resources, and the delivery of their products. It provides solutions for measurement and metrics, collects data from disparate teams and applications, and presents intuitive summary charts of key information in a single view. This consolidated view allows users to analyze trends, to manage by exception, and to drill down for more information when necessary.

IBM Rational Dashboard has built-in data collection interfaces for common file formats (such as CSV and Excel), databases (such as ODBC, Microsoft SQL and Oracle), as well as commercial applications (such as Microsoft Project, IBM Rational Change, Synergy, and DOORS). It has the ability to display up-to-date project status information in a graphical multi-level format from all these applications so that managers can focus on decision making rather than manually gathering data and compiling reports to ensure the success of their business.

## 1.2. *What is Measurement?*

Measurement is the process used by managers to monitor and control the processes, resources, and products that are part of their program. The purpose of measurement is to:

- Characterize (baseline performance)
- Evaluate (actual vs. plan)
- Predict (estimation and prediction)

- Improve (process improvement)

Managers need a mechanism to document expectations and plans for key aspects of their program, have confidence that the product will be delivered on time, and ensure that past mistakes will not reoccur. Engineering tools do not provide a way to setup a plan line, evaluate processes against the plan, and do management by exception, which is why measurement applications like IBM Rational Dashboard are necessary for a project's success.

Primary software measurement guidance is provided by ISO/IEC Software Measurement Processes, DOD PSM Practical Software and Systems Measurement and CMMI. These processes tell organizations how well they are doing in specific areas. They do not outline *how* to do measurement and *which* measurements must be deployed. They simply provide an organization with general guidelines in reaching compliance.

IBM Rational Dashboard helps in all process areas of deploying measurement: from planning to setting up your project accurately to monitoring to controlling. It helps establish measurement objectives; define measures; data collection procedures; analysis procedures; collect data; helps analyze and store data and results; and communicate the results to your team.

### ***1.3. IBM Rational Dashboard High Level Features***

IBM Rational Dashboard is a performance measurement application that provides a number of key features to managers, allowing them to control and monitor their projects, including:

- Standard items that simplify and speed up the deployment of popular standards and maturity models
- Automated extraction and analysis of data
- Dashboards based on lifecycle phase, ensuring managers have the most relevant and up-to-date information
- Delivery of Dashboards to managers using a standard web browser
- Extensive tailoring capabilities to support organizational-specific processes, products and measurement requirements
- Security features to control access to data
- A set of licensed interfaces for data collection
- Three types of user licenses (executives, managers and administrators)

## 2. Terminology

### **Organization Tree**

The Organization Tree is located on the left side of the main Status Tab. IBM Rational Dashboard utilizes a single tree for organizing information to provide managers with a controlled method of accessing data. This Organization Tree contains a hierarchy of Folders, Sub-folders and Units. Folders can be nested to represent the structure of the organization and the projects managed. Units represent departments, programs, projects or any group of resources that needs to be managed.

### **Folder**

A Folder is a hierarchical container in the Organization Tree for Units and other Folders. When a Folder is selected in the Status Tab, a summary of the status of each Unit contained within the Folder is displayed. A Folder is typically used to represent a logical group within the organization (e.g. a department responsible for a number of projects).

### **Unit**

A Unit in the IBM Rational Dashboard correlates to a project. You can create as many Units as you like and arrange them in the organization tree to form the management cockpit by which managers can monitor what is important to them. There are two types of units, basic and summary. Summary units provide a roll-up or aggregation of status contained in basic units.

Units contain data (Managed Items) collected by managers for a specific purpose. Managers assign Information Needs (e.g. Requirements Stability) to Units to provide a pre-defined way to access and display measurement data. Information Needs can be assigned individually or via a Unit Template (available when the Unit is first created).

### **Unit Template**

Unit Templates provide a quick and easy way to deploy a new Unit re-using common Information Needs and other Unit properties.

Unit Templates are ideal for projects that need to manage similar information. For example, if an organization is tracking five projects, with each project collecting the same data on cost, schedule, and requirements, then an administrator could create a Unit Template including Information Needs, Attributes, Phases, Roles and a default Schedule to meet the needs of all of the projects. When creating the Unit associated with their project a manager can then simply assign the Unit Template thereby ensuring that data collected for that project is displayed consistently.

IBM Rational Dashboard contains several pre-defined Unit Templates, including the DOORS Template containing Information Needs relevant to requirements management and IBM Rational Change/Synergy templates containing Information Needs relevant to change and configuration management. Unit Templates are managed through the Library Tab.

### **Form**

A form is a set of data, such as a set of risks or actions items, of interest to a manager. Forms are provided in the Portal to reduce or eliminate the need to store project data in other locations, such as spreadsheets. A unit may have any number of forms (including zero) as needed by management policy. Managers may create new form data records and review them as needed. Each form and form data record has a color-coded status to speed review and problem resolution.

### **Item**

An Item is a piece of collected data, for example a project schedule file, a risk plan, a DOORS module or a Synergy release. A Managed Item is an Item that has been assigned to a Unit. Managed Items support the Information Needs for the Unit and are related to the sources of collected data through Interfaces. From the Managed Item, you can drill down to corresponding graphs which show different representations of the information collected.

### **Information Need**

An Information Need is a collection of Graphs, series, status (current actual, target and alarm), and measurement guidance that a manager needs to assess one aspect of a project status. For example, if a manager wanted to control progress of project requirements, the Information Need would be a collection of graphs, series, and guidance that shows requirements progress.

Multiple Information Needs can be assigned to a Unit. A typical Unit for a software project would likely include the following Information Needs: schedule, requirements, cost, defects, testing, risk, configuration management, software size, and productivity.

IBM Rational Dashboard is shipped with a library of Information Needs and you can also create your own via the Library Tab.

### **Dimensions**

A Dimension allows status captured in Information Needs to be organized in user-defined categories. For example, the Information Needs could support three company standards such as a company management policy, a compliance standard, and a maturity model level. In this case, there would be three Dimensions corresponding to each of the three company standards.



A Dimension consists of one or more Dimension Elements, where each Element represents a sub-category of status. In our example, the Dimension “Company Management Policy” could be created with three Dimension elements (“Quality”, “Productivity”, and “Product size”). Dimensions and their associated Dimension Elements are created via the Library Tab.

Dimension Elements are assigned to Information Needs via the Library Tab. Each Information Need can have one or more Dimension Elements assigned to it. For example, the Information Need “Requirements Progress” could be assigned the Dimension Element “Requirements Management” which is part of the “Software CMMI Maturity Level 2” Dimension. When an Information Need is assigned to a Unit, the associated Dimensions and Dimension Elements are automatically created within the unit from the definitions in the Library. Users are able to view Unit status by Information Need and Dimension Element.

Through Dimensions, IBM Rational Dashboard can support any number of standards, policy, maturity model, quality model, or compliance documents. For example, a Dimension can be created for company development policy, ISO-9000, one of the maturity levels in the SEI’s CMMI, a government regulation, or an industry practice. Each one of these documents becomes a Dimension, and the major elements, sub-categories or process areas become a Dimension Element. The Portal then displays the status provided by the associated Information Needs against the related Dimension and Dimension Elements.

### **Graphs**

Graphs are defined and associated with Information Needs via the Library Tab. The predefined Information Needs have a set of associated Graphs which can be used “as is” or re-configured to meet your requirements. You can also create and associate new Graphs with existing or new Information Needs.

When a Managed Item is assigned to a Unit, Graphs are automatically created for the Item based on the Information Needs associated with that Unit.

Users can tailor the appearance of graphs, enter data, or expand the definition of graphs. The Portal allows users to perform data entry (for manual data entry only) and change all visual aspects of the graph using the built-in graph formatting pages.

### **Interface**

Interfaces provide instructions for one or more data sets to collect and the fields within each data set, as well as a set of queries which generate metric/series values. The Portal includes pre-defined Interfaces (such as Microsoft Project, IBM Rational Doors, IBM

Rational Change, and IBM Rational Synergy) which contain data definitions for the attributes required for the pre-defined Information Needs.

To support tailoring, you can modify existing interfaces or create a new one to match an interface to the processes and tools in use at your organization. Each Interface has a set of fields describing what data is to be collected from the (external) application.

Interfaces are managed from the Collection Tab and are intended primarily for those users responsible for data collection.

### **GANTT View**

The Gantt view provides a compact, color coded status summary of the Items within a Unit. With the Gantt view, a manager can review the plan and progress for the Unit, including start date, end date, and status for each Managed Item. This provides a quick summary of past performance, current status, and future plans.

### **Phases**

Phases are periods between a Unit's start and end date which can be used for labeling graphs. For example, typical project phases are: Plan, Requirements, Build, Test, and Deploy. Seeing Phase titles is useful to managers and other stakeholders to indicate what Phase the Unit is currently in, what Phases have completed, and what Phases are up coming.

Each Phase has a title, start and end date, where the Phase dates cannot overlap with another Phase. There is no limit to how many can be created, but practically, five to seven Phases.

### **Schedules**

Schedules play an important part in defining when data is collected and how it is displayed. An administrator can create as many schedules as needed by your organization. A schedule is simply a named collection of periods. Each schedule period consists of a start date, an end date, and a collect date. The collect date is either the same as the end date or after. There are two primary uses for schedules: collection and display.

The Collector uses the collect date for a schedule to determine when to actually collect item data. For example, if you assign an Excel document (as an item) to a Unit, and that item uses a weekly schedule, then the Collector will process the Excel data for only one day of each week regardless of how many times the Collector runs. The specific day would, in this example, be based on the collect date. So, if you had set all the periods in the weekly schedule to start on Monday, end on Sunday, and collect on Sunday, the Collector will collect the Excel data on Sunday.

Schedules are also used for displaying data. Each schedule-based graph has a schedule assigned to it. Event-based graphs don't use schedules. The graph schedule defines the possible data points for each series in the graph. For example, the most common application of a schedule is as the X-axis in a run graph. Along the bottom of the run graph are the periods of the schedule, each period having a data value for each series. Notice that in a run graph, you can select a date range of the available data to see, for example, only this year or a rolling date range (e.g. 12 months back and 3 months forward). The data is still maintained for the entire schedule, though some of it is hidden based on the date range.

### **Status**

The status of Units, Items, Dimensions, and Graphs are all driven by the Alarm series assigned to Graphs. All status in Portal is based on the worst-case status of Alarms, the period of performance, and the alarm color ranking.

### **Period of Performance**

The last completed period in the schedule is used to determine Status. The Alarm series on a graph typically displays the full history of Status but only the last completed period is used for the Unit, Item, Dimension, or Graph status indicators. For example: A Graph with a monthly schedule is reporting from January 1, 2006 to December 1, 2006. The current date is May 15, 2006. Even if data has already been inserted for May, the last completed period is April. The alarm color reported in April is the status that will be used.

### **Worst-case**

Status in IBM Rational Dashboard is reported using the worst-case value, starting at the Graph level and rolling up to the Unit.

Some Graphs may have more than one Alarm series. In this case, an option is available to use a specified Alarm series to report Status or to use the worst-case indicator across all Alarm series on the Graph. For example: a Graph may have one Alarm series that reports the status of the actual data compared to the plan data. Additional Alarm series could be added to report missing data, a value out of range or a data pattern. These additional Alarm series may not directly indicate status and would not be used in reporting.

### **Alarm Color Ranking**

Alarm colors ranking determine a value or the “good” or “bad” worth of a color. To quickly inform users the current health of their project, the status colors are displayed on the GANTT, Unit, Item, Dimension, and Graph views as well as an individual graphs as alarm series.

## **Alerts**

Alerts are the mechanism for notifying a user that something they are interested in has changed. Alerts help managers administer the system and manage their projects. An alert is a notification that explains what happened and includes a URL to go see, fix, and/or review an event. Alerts save time by pin-pointing problem areas. There are two parts to the alert system: Alert Registration and Alert Notification.

An Alert Registration is an indication that a user wants to see a specific Alert. A user selects one of the available Alert Event Types, along with any parameters, and adds it to their list of Registrations. The application then keeps track of the events that each user has registered for and generates Alert Notifications when the user clicks the process button.

The Alert Notification List is a table of Alerts that have occurred and that are ready for review by the user. Users can view individual Alerts by clicking on the Alert in the Alert Notification List. Once an Alert is selected, the information regarding the alert will display below. Five descriptive fields are displayed to inform the user why the alert was triggered:

- Source - displays the collector that was used for the collection.
- Event - displays the alert event.
- Date/Time - indicates when the failure took place.
- URL - displays the URL that failed.
- Description - details the reason the alert was triggered.

The user may clear an Alert to remove it from the list.

### **3. Overview of the Collection Process**

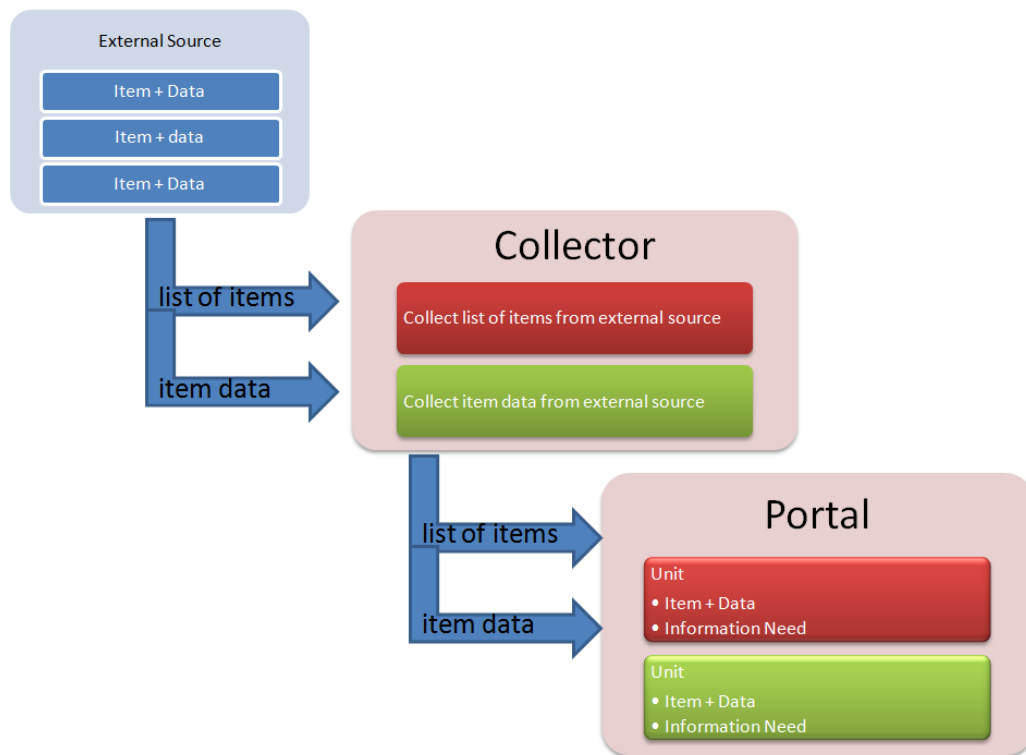
Data collection is one of the key aspects of the measurement process. Configuring and monitoring data collection involves a number of concepts explained in this section. In IBM Rational Dashboard, collection tasks are performed in the Collector, a Windows application, and in the Portal.

#### ***3.1. The Big Picture***

In starting to understand collection, let's first consider how data "naturally" exists in a data source. In almost all cases, data is grouped logically into sets. For example, with Microsoft Project Server, the sets of data are called schedules. In IBM Rational Doors they are called "modules". In defect tools, the sets might be called "releases". In IBM Rational Dashboard, we generally call these "items". An item is a set of data which can be identified in an external data source. From the previous examples, a project schedule, a requirements module and a software release are all examples of items.

Both the Collector and Portal rely on this concept of items. There are numerous tools and functionality that automates handling items and the data that is contained within an item. The Collector is responsible for collecting items, and the Portal is responsible for assigning items to units and for updating graphs which are attached to items. The general flow of an item plus associated data is shown in the following figure.

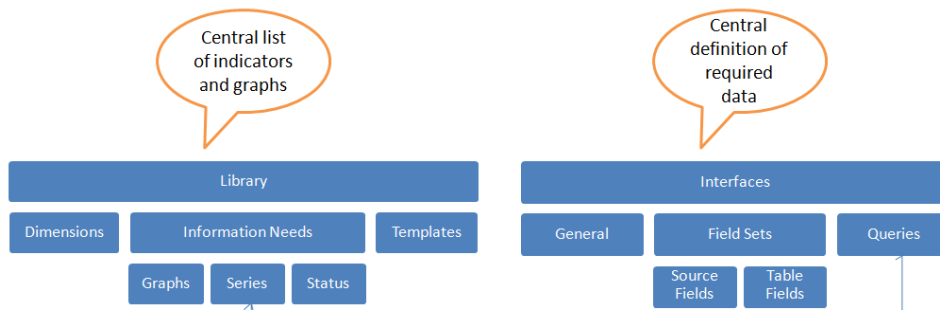
Using the Collector, a user creates a "data source" which allows the Collector to utilize the appropriate API/interfacing mechanism to retrieve a list of items. The Collector then sends this list of items to the Portal. Using the Portal, a user assigns an item to a unit (remember that a unit represents a project or program). From this point on, when the Collector collects data from that data source, it will collect data for the item according to the schedule that was defined in the Portal. Using this general approach, IBM Rational Dashboard is able to collect data from external tools and update graphs on a periodic basis.



From the above figure, notice that in addition to the “Item + Data” in the Portal, there is also an “Information Need” within each unit. In addition to the concept of an “item”, IBM Rational Dashboard also uses the concept of an “information need” to represent the visual data that should be displayed in a Unit. For example, the Library in the Portal might contain an information need called “Requirements Volatility” that contains graphs and indicators that aid in tracking and monitoring the volatility of project requirements.

In the Portal, when a user assigns an item to a unit, the Portal will determine whether there are any information needs which have been configured for that kind of item. The Portal can determine this because the Collector actually tags each item with the type of interface that the item was collected from. When the item is assigned, the Portal checks to see if any information needs have been configured for that interface. If there are, then the Portal creates the graphs defined for that information need within the unit AND wires up the graphs to the item so that live data is shown in the graph.

The relationship between information needs and interfaces is shown in the following figure.



### 3.1.1. Control Flow in the Collection Process

This subsection describes how the collection processed is controlled. “Control” means the mechanisms that determine which external data sources are used, and which objects in IBM Rational Dashboard are involved in the collection process. The concept of “control” also involves deciding when to collect data and for which objects to collect data.

Control flow involves the following steps:

1. The users runs the Collector to perform data collection process by selecting one or more data sources to run and selecting a date to be used as the collection date.
2. The Collector requests from the Portal the interface associated with the selected data source. This step ensures that the Collector knows the required fields to collect for the selected data source.
3. The Collector initializes the selected data source(s), validating parameters and checking the system configuration for required components and files (when possible).
4. The Collector requests that the selected data source connect to the external application and retrieve a list of containers and items. (Note: some external applications do not support the concept of containers.) This is the “current list of items.”
5. The Collector requests from the Portal a list of items that have been previously (this is the “previous list of items”) found for the associated data source. This list indicates contains all items and identifies those items that have a schedule period where the period collect date matches the selected collect date.
6. The Collector compares the current list of items to the previous list of items. The Collector then determines which of the current list of items are new, that is, have not been found/collected before.

7. The Collector then builds a list of items to be collected, which contains new items and those that have a schedule period whose collection date matches the selected collect date.
8. For each item in the list of items to be collected, the Collector requests that the selected data source connect to the external application and collect data for the fields specified in the interface. The Collector generates whatever script or interface mechanism is required to perform this task.
9. For each item in the list of items to be collected, the Collector sends to the Portal the data for the associated item.
10. The Collector requests that the Portal perform a data refresh of all units that contain items that were collected (i.e. contained in the list of items to be collected). The Portal performs a refresh for the selected collection date.
11. The Collector sends the Portal a run log that identifies the data source(s) collected (in case there was more than one) and summary information about the items and data found during the collection, as well as any errors encountered.
12. The Portal builds a list of units that use each item sent by the Collector, and refreshes the graphs which use those items. The Portal saves all refreshed data.

### ***3.1.2. Data Flow in the Collection Process***

This subsection describes the flow of data through the process of collection.

Data flow involves the following steps:

1. Each interface in the Portal Collection tab contains a definition of which transform database will be used to store data, and a list of field sets that define a table (in the transform database), attributes/fields in the external application and the name of a transform table field to store the attributes. Notice that the interface definition
2. As the Collector is run and performs the collection process, it returns a list of items that were found from each data source. This list is shown in the assign sub-tab of the Status page where each item is assigned to one or more basic units (which represent projects or programs).
3. In the Status tab, the Portal is used to assign a schedule to each item assigned to a unit. If the unit is using “basic scheduling mode, then all items and graphs are using the same schedule. If the unit is using advanced schedule mode, then each item and graph could have a different schedule.
4. As the Collector obtains a list of items for each data source, and then collects data for each one, it forwards the data to Web Services after each item is collected.



5. For each item from the Collector, the Web Services process the data from the Collector row by row. The Web Services validate the connection information for the transform database and opens a connection to the associated transform database.
6. The Web Services uses the definition of the associated Interface (from the Portal) to map each attribute/field in the data set to a table field in the transform database. The Web Services coerces the data type in the attribute/field into that of the transform database table field. The Web Services also inserts default values when a null value is detected in the data.
7. The Collector then requests that the Web Services performs a data refresh for all affected items, which causes the Web Services to open each affected basic unit and refresh the data in the order that was specified in the Unit Properties. Refresh includes evaluating and running the queries, then the equations in data series, then any region equations and then finally the equations in alarm series.

## **3.2. Components of Collection**

### **3.2.1. Interfaces – What To Collect and How To Process**

An interface describes what data to collect, where to store the collected data and how to analyze (i.e. query) the stored data in order to generate one or more data points. Each interface is associated with an external application or file format, called an interface type, such as Microsoft Project, IBM Rational Doors or a set of CSV files.

The main communication between the Portal and the Collector is defined by **Interfaces**. An Interface essentially tells the collector **what** to collect and **how** to use the information once it is collected. You create an interface for each external application you're using and for each variation of fields within the same external application.

For example, in DOORS, different projects may be using differently structured DOORS modules to capture requirements (e.g. different attributes, types or values). These projects could require different Interfaces to support the same Information Need (e.g. Requirements Progress).

Before data can be collected, there must be an Interface describing the data to be collected that can be provided for to the Collector. The Collector reads the interface definition, generates the appropriate integration, validates that the data is available and then collects data from the external application. This mechanism results in not having to perform any script or software development in order to gather data for the measurement process.

More information on how interfaces are used is contained in the next section.

### 3.2.1.1. General Properties of the Interface

Each interface contains the following general properties.

Name	description
Title	a unique name that identifies the interface within the Portal
interface type	indicates the type of external application, database or file format that this interface will be used to process data
database server	selects a SQL database (defined on the Admin tab) where data will be stored. Notice that the table within the selected database is specified in each field set (since there may be more than one field set within an interface)

The interface type must be selected from a list of available types within the Portal. The list of available interface types is read-only and cannot be modified except by update of the Portal and Collector software. New releases often contain new interface types, so check with the support site for new/modified interface types.

To simplify creating a new interface, the Portal allows you to copy the field sets, fields and queries from an existing interface that you select. When you copy an existing interface, all the field sets, fields and queries that exist in the current interface are deleted and the new ones created from the selected interface. Some comments on using the copy interface:

- Before using the copy command, you should select the interface type and save the interface.
- You cannot change an interface type after the initial save, but you can use the copy command to copy one interface into another one of a different type.

### 3.2.1.2. Overview of Field Sets

The Field Set defines what source attributes or fields are to be collected and where the resulting data is stored. The field set consists of the following:

- A title
- A database table where the data for this field set will be stored
- A replacement tag
- A list of fields

The interface provides the database server and the table name is defined in the field set. Each field set should use a different database table name to ensure that data is not combined. When the interface is saved (and the “add” and “update” checkboxes are selected), the Portal will create the specified field set table, and then verify that each field in the list of fields.

The field set definition is used by the Collector when collecting data from a data source. For example, you may decide to collect field1, field2 and field3 and store this data in table called ‘Defects’. In this case, the field set could be called “Defect Set”, the table “Defects” and there would be three fields defined in the field set.

An interface may contain more than one field set. The purpose of having multiple field sets is to allow the collection of multiple sets of data from the same source. For example, you would define two field sets to collect both cost and resource data from Microsoft project, or defect and workspace data from IBM Rational Synergy. The Collector will collect as many sets of data for each interface as defined by the number of field sets.

### **3.2.1.3. Overview of a Field Set Parameter Replacement Tag**

Each field set in an interface has an optional parameter replacement tag, which can save a text string. The parameter replacement tag is a mechanism to pass data from the portal to the collector at collection run time. The use of this tag is optional -- many field sets do not use them. When the parameter replacement tag is used, the value set in the interface is passed to the Collector which inserts the parameter replacement tag text into the ‘Get Details SQL’ field in place of the !SETTAG! reserved word. A different parameter replacement tag can be entered for each field set in an interface.

For example, the parameter replacement tag is used with a IBM Rational Synergy interface to differentiate between the ccm queries used for each of the five field sets. When the Collector collects for a IBM Rational Synergy data source, it will process each one of the five sets individually. When the Collector gets the details for the first field set, it replaces !SETTAG!, saved in the ‘Get Details SQL’ field, with the parameter replacement tag text for the first IBM Rational Synergy field set. When the Collector gets the details for the second field set, it replaces !SETTAG! with the parameter replacement tag text for the second IBM Rational Synergy field set. It proceeds in this manner until all five field sets for the interface have been processed.

The parameter replacement tag is also beneficial for other tool types where it is necessary to collect multiple sets of information from the same database.

### 3.2.1.4. Overview of a Field

Each field set consists of one or more fields. The field performs three purposes, as follows:

1. Identifies the name of an attribute, field or common in an external application that is required for data collection
2. Describes the data type and validation criteria for the field
3. Identifies a database table column (in an IBM Rational Dashboard transform database) where this field is to be stored

Each field that you create will be extracted from an external application (unless the “Do not collect?” option is selected, which is described below). The source attribute name must exactly match the name given to the attribute in the external application. For example, if the field in Microsoft Project Server 2003 is called “task\_act\_start”, then this field must be defined the same. In IBM Rational Doors for example, some object attributes have spaces in the name, such as “Object Heading” – the field in IBM Rational Dashboard should be entered with the embedded space. Do not use leading or trailing spaces or other white space. The order in which the fields are defined does not matter.

Each field also contains a table field name. This is the name of the SQL table field where the resulting data will be stored. Each field in the external application is stored in a table field whose name is specified here. The table field name allows you to use more meaningful or descriptive names (instead of the name used by the external application) In addition, the table field name insulates your queries from name changes in the external application, since all the queries are written against the table field name and not the external attribute names.

Using the “null” option controls whether the transform database table will be defined to allow a null value or not. If a null value is not allowed, then the external application must provide a value or you must select the option to provide a default value. If the value from the external application is null and no default is provided, then the entire record (that is, the data for all fields) will fail when it is inserted.

The field contains an option for “Do not collect?” – this option is used to define and validate your interface, fields and queries in manageable pieces. For example, you might define all the fields in your external application and then set all but two of them to “Do not collect?”. Once you have validated the collection of data and checked the queries, you could turn on other fields using the “Do not collect?” option. For example, you might then work on the date fields in your external application. By using this flag, you can methodically work through all the aspects of an integration.

A field set with no fields is simply a place-holder. If this field set is run by the Collector, no data will be collected. Except in initial setup, you should define one or more fields in a field set.

When the interface is saved (using the “Save” button at the bottom left of the page, IBM Rational Dashboard will make sure that the fields in the interface are contained in the transform database.

### 3.2.1.5. Interface Queries

In addition to fields, each interface contains a set of queries. There is one set of queries for all field sets and fields in the interface. An interface query defines the processing needed to generate a data point. A query is the first opportunity in the Portal to analyze data and generate a data point.

To allow the rules for generating a data point to be modified, queries use the SQL language. SQL is a fairly simple language to use and provides excellent capabilities for data analysis. Also, the Portal provides a query builder that helps develop basic SQL queries, and also an editor for writing complex SQL queries.

There are two types of queries, described in the following table.

Type	Purpose
Single series	Generates a data point in a schedule based graph
	Generates a set of records in a snapshot grid
Multi series	Generates a set of series (dynamically) based on the contents of the data, and then generates a data point for each series

During data processing, Web Services will automatically execute the queries that are associated with each interface. For example, the Collector finds a list of items from an interface and sends them to the Portal. An administrator will assign that item to a unit. In the Unit, the information needs associated with the item contain graphs whose series are queries. Once this set of assignments is made (items to a unit, and queries to an interface), IBM Rational Dashboard takes care of running the queries automatically as new item data is collected.

### 3.2.1.6. Query Builder

The query builder allows you to quickly and easily build queries that have a fairly basic definition. As shown below, the Edit Query page provides various selectable options to build a SQL query statement.

The screenshot shows the 'Edit Query' interface with the following elements:

- Title:** Requirements Total (6)
- Select Series:** Assign these queries to series in the Library ...
- Query Builder:** Query Edit Test Query
- 1 Data from:** DOORSReqs database table
- 2 Result is:** a count of (selected) a sum of CreatedOn (Default Set)
- 3 Filters are:** the current item (checked) for the current period (checked)
- 4 With terms:** obj\_type = 'requirement'  
CreatedOn <= %EndDate% (Delete)
- Add new query term:** CreatedOn (Default Set) = [ ] Start Date Insert
- Buttons:** Save Cancel

The first line, “1 Data from:”, displays the name of the table in the first field set in the interface.

The second line, “2 Result is”, allows you to specify whether the records resulting from the SQL query should be the count of the records, or the sum of a selected field.

The “count” option is the most common, where you are interested in counting the number of data records that match a set of filter criteria (in SQL these appear in the “WHERE” clause). When you select “count”, the resulting value is the number of records that match.

The “sum” option is used when the value you want is the sum of one of the fields in your data set. You create needed filter criteria (to select the appropriate records) and

the “sum” a single field to create the resulting value. For example, you could apply the “sum” option to the field “planned\_hours” to generate the total number of planned hours for a period. If you need to sum more than one field, you must use the Query Edit sub-tab and enter the SQL in by hand.

The third line, “3 Filters are:”, lets you add common filters to the SQL query.

The option “the current item” adds a WHERE clause term to the SQL statement that selects only records that were collected from one item, the item assigned to the current unit.

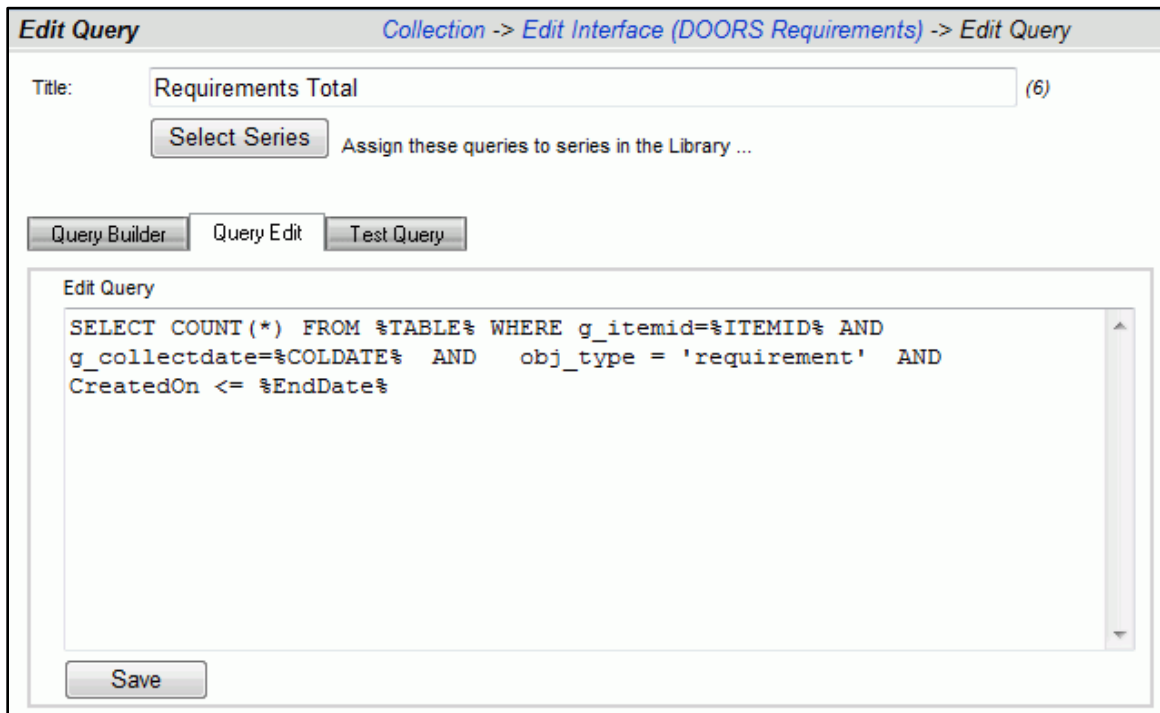
The option “the current period” adds a WHERE clause term to the SQL statement that selects only the records that were collected during one collection period.

The “4 With terms:” section allows you to add additional terms to the SQL WHERE CLAUSE using the “AND” operator. To add a term, you select a field from the drop down titled “Add a new term”. Then, you select a comparison operator and provide a value. A value can be provided by either entering a value into the edit box either or by selecting a tag and then pressing the “Insert” button. To place the new term into the query, press the “Add” button on the left.

To use an “OR” operator, you must use the Query Edit sub-tab.

### 3.2.1.7. Query Edit Page

The Query Edit page, accessible from the Query Edit sub-tab, allows you to directly enter or modify the SQL for an interface query. The figure below shows the Query Edit sub-tab for the query created in the Query Builder.



### 3.2.1.8. Query Test Page

The Query Test page allows you to test an interface query, and to explore the database table used by the interface. In order to test a query, the query should be run against a table that contains data. To get actual data, you must select a unit and item that uses the same interface that contains the selected query. As shown in the screen below, the Test Query sub-tab requires that you select a unit and item before trying to test the query.



**Edit Query** *Collection -> Edit Interface (DOORS Requirements) -> Edit Query*

Title:  (6)

Assign these queries to series in the Library ...

**Test Query**  
Units and items below which uses this query in a graph or grid. Select a unit/item and press 'run', or press 'run' .

Unit:

Item:

Once you have pressed the “Run” button, the query test page is shown, as depicted below. The buttons on the top left labeled “Show” allow you to explore the data that is in the specified table. These buttons help you see what data was collected, for which dates and items and which collectors collected the data. These buttons save you time when testing or debugging a query.

*Collection -> Edit Interface (DOORS Requirements) -> Edit Query -> Query Test (Requirements Total)*

Server: Transform Database	Contents of database table	Tables and Fields
%table%: DOORSReqs	Rows: 0 rows <input type="button" value="Show"/>	DOORSReqs
%unitid%: 158	Dates: 0 distinct dates <input type="button" value="Show"/>	• CreatedOn
%itemid%: 13090	Items: 0 distinct items <input type="button" value="Show"/>	• ModifiedOn
%startdate%: 4/1/2009	Collectors: 0 distinct collectors <input type="button" value="Show"/>	• Obj_heading
%enddate%: 4/30/2009		• Status
%coldate%: 5/1/2009		• obj_text
		• obj_type

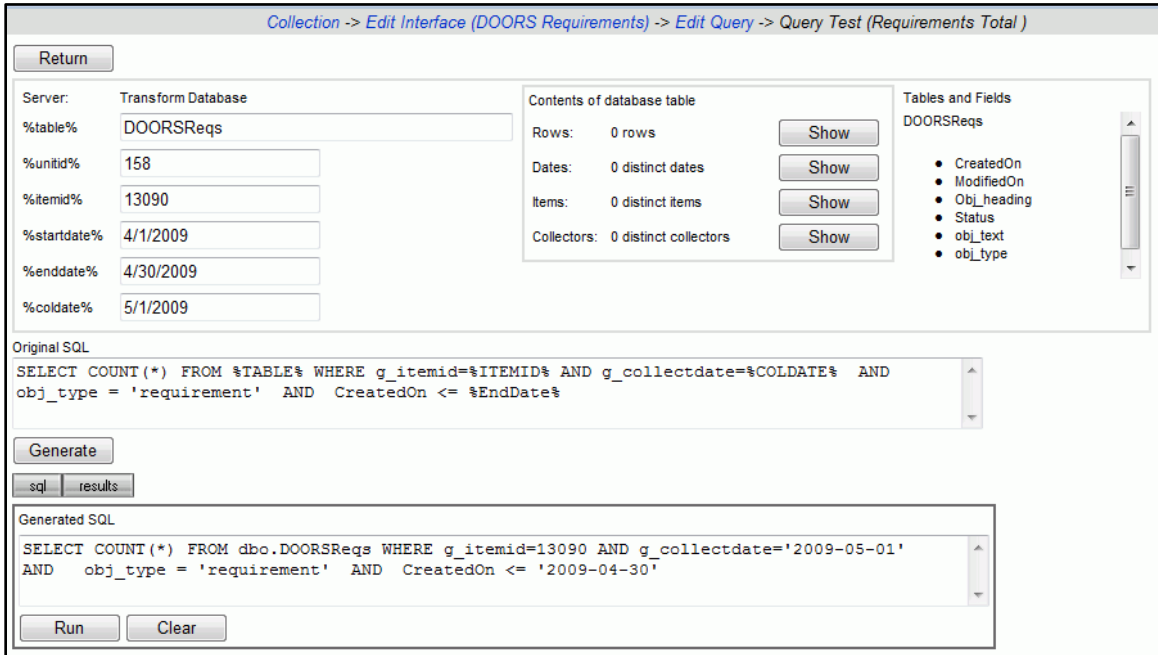
Original SQL  

```
SELECT COUNT(*) FROM %TABLE% WHERE g_itemid=%ITEMID% AND g_collectdate=%COLDATE% AND obj_type = 'requirement' AND CreatedOn <= %EndDate%
```

Generated SQL

Notice that all the common query tags are shown in two columns on the left, along with the original SQL. You may modify any of the tag values by simply changing the appropriate edit box. To see the original query with tags replaced, you must press the

“Generate” button. This creates a SQL statement where all tags have been replaced. The figure below shows the query with the tags replaced.



Then, you can press the “Run” button to actually run the query. The results are shown on the “results” sub-tab, either as a single record in the case of a query with a “count(\*)” or “sum(field)” term, or a data table in the case of a query that returns records such as a query for a snapshot grid.

### 3.2.2. Databases – Where to Store Data

Databases are used to store the data required for the Portal and Web Services. There are two default IBM Rational Dashboard databases:

1. Portal database
2. Transform database

A portal database stores the data needed to run the Portal and Web Services, such as accounts, units, security, alerts, graphs and processed graph data. There should only be one Portal database for each IBM Rational Dashboard installation. When using a Web Farm, multiple IBM Rational Dashboard Portal servers should share the same Portal database.

A transform database acts as a data mart, storing data collected from external applications by the Collector. The transform database is used by the Portal as its source of data when performing data calculations or analysis. For example, when using a Defect Growth information need, collected defect data is stored in a transform database, which is then counted to construct (for example) the number of new defects during the last month. The resulting “counted” data is stored in a portal database.

The structure of a Transform database is maintained by the Portal. As you define data in the Portal (by adding fields to interfaces for example), tables and fields are automatically added, modified or dropped from the Transform database. You can turn off this feature in the Portal user interface if you want to maintain the Transform database manually.

You may define more than one transform database to separate data based on any appropriate factors, including capacity, performance or security. Initially, you should use just one transform database. Over time, you can add transform databases. The only limitation of using multiple transform databases is that you cannot combine data which resides in different transform databases, unless you manually create Views.

### **3.2.3. Schedules - When to Collect**

Schedules are important in determining *when* data is going to be collected from a source. You may create schedules with any frequency; common options are to have collections on a daily, weekly, or monthly frequency. Each period for a schedule consists of the starting and ending dates for the period and the date that the data is actually being collected or a ‘collection date’. If a collection is run for a source, but it is not a date that any schedule has a collection date for, then no data will be collected. A schedule must be set for an item when it is created and must relate to when the data is going to be collected.

Performance Measurement data is generally gathered on a periodic basis (e.g. weekly or monthly, over the life of a project or a program). Each period has a start date and an end date. In IBM Rational Dashboard, the collection of periods and dates is called a schedule. Schedules can be defined monthly, weekly, daily, or individual dates may be added as needed.

IBM Rational Dashboard schedules are defined by administrators and available to all users to assign to a unit or an item. Some schedules may be regular (monthly or weekly) and others may be more irregular (a schedule based on project milestones or phases).

**Administration**

**Security**

- Accounts
- Roles

**Graph**

- Report Groups
- Unit Reports

**Other**

- Attributes
- Database Utilities
- Forms
- Logs
- Schedules
- Servers
- Version Info

**Schedules**

Add new schedule.

		Title	Start Date	End Date
<input type="button" value="edit"/>	<input type="button" value="delete"/>	Daily 2006 - 2007	01 Jan 06	31 Dec 07
<input type="button" value="edit"/>	<input type="button" value="delete"/>	Monthly 2006 - 2010	01 Jan 06	31 Dec 10
<input type="button" value="edit"/>	<input type="button" value="delete"/>	Monthly Schedule 2008 - 2009	01 Jan 08	31 Dec 09
<input type="button" value="edit"/>	<input type="button" value="delete"/>	Weekly Schedule	01 Sep 07	31 Dec 08

Figure 22

### 3.3. Item List, Item Details and Units

This subsection provides a high level walkthrough of the concept of an “item” and how it is used within a Unit.

#### 3.3.1. Overview of Items Concept

On the left hand side of the Status page is the Organization Tree, containing your organization structure, grouped into folders, sub-folders, and units. Your Organization Tree should represent a structure that is familiar and easily recognized by your managers. Included with the Portal, sample folders and units (Figure 1) show predefined Information Needs and utilize the “demo data” within the portal.

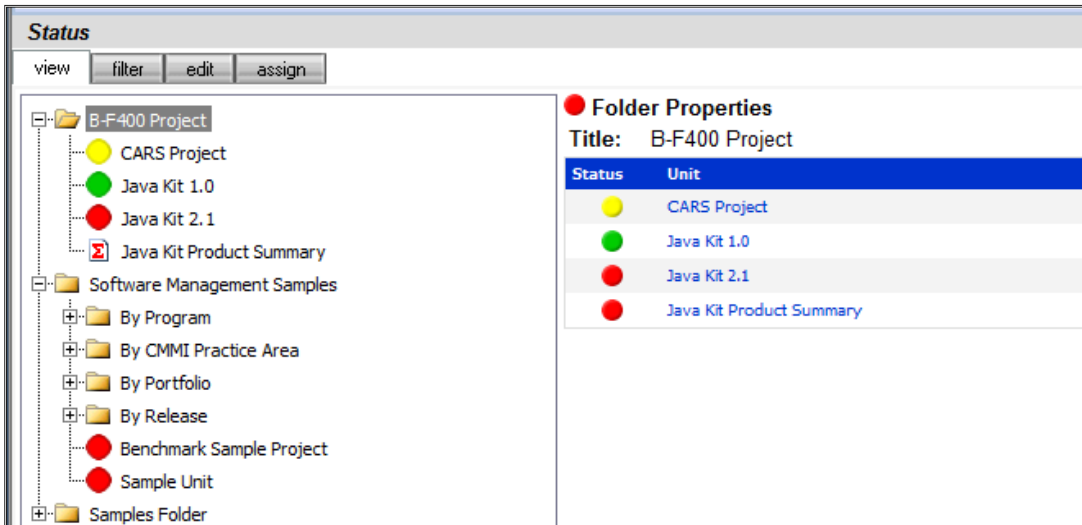


Figure 1

The sample Units are visible to users whose Accounts have been assigned the standard “Sample View” Role via the Admin.

### 3.3.2. The Item List

On the Unit Properties page, basic units are displayed in folders in the Organization tree. In the Tree, the basic unit is displayed as a colored circle while the summary unit is displayed as a colored sigma sign. In the *Software Management Samples* folder, click on the *Sample Unit* (Figure 2).

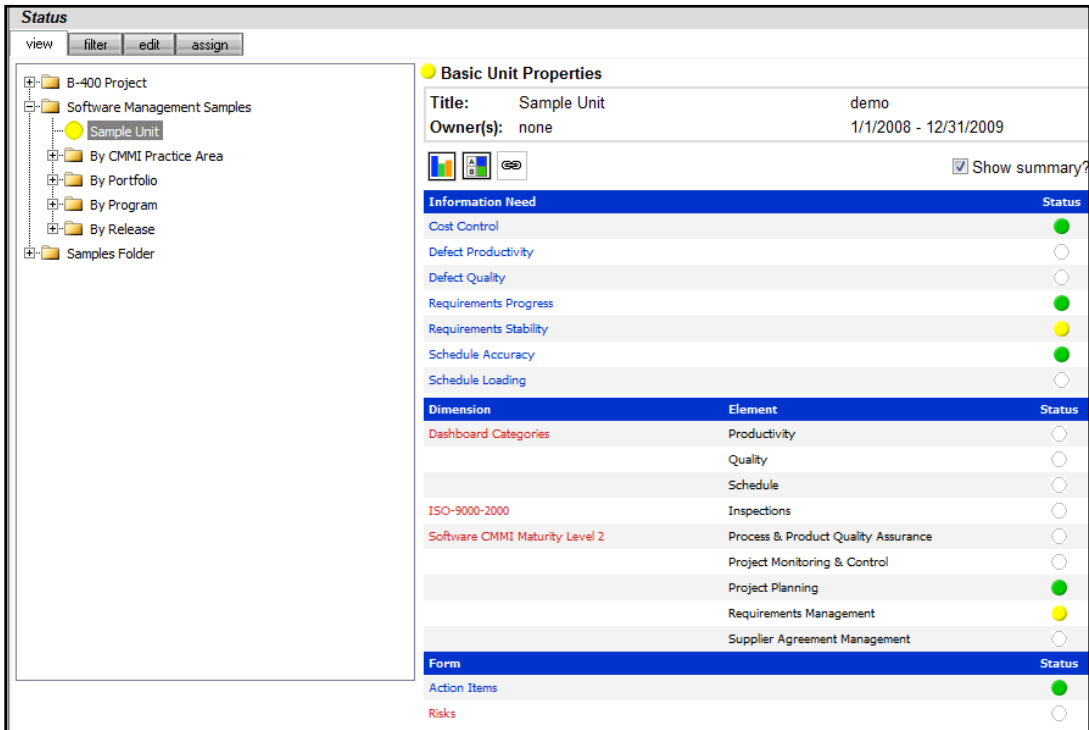


Figure 2

There are several methods to drill into more detail for the *Sample Unit*, as follows:

- The details icon (the left-most of three icons located above the text Information Need, on the right hand side)
- An information need hyperlink (in the Information need column)
- A dimension hyperlink (in the dimension column)
- A form hyperlink (in the form column)

The Unit page provides commands to navigate, configure and secure a unit, as well as enter plan, actual, analysis/notes and form data.

By pressing the Details icon, the Unit Status Page first displays the Gantt View (Figure 3). The Views panel (on the left-hand side) provides a number of other commands to review status and progress.

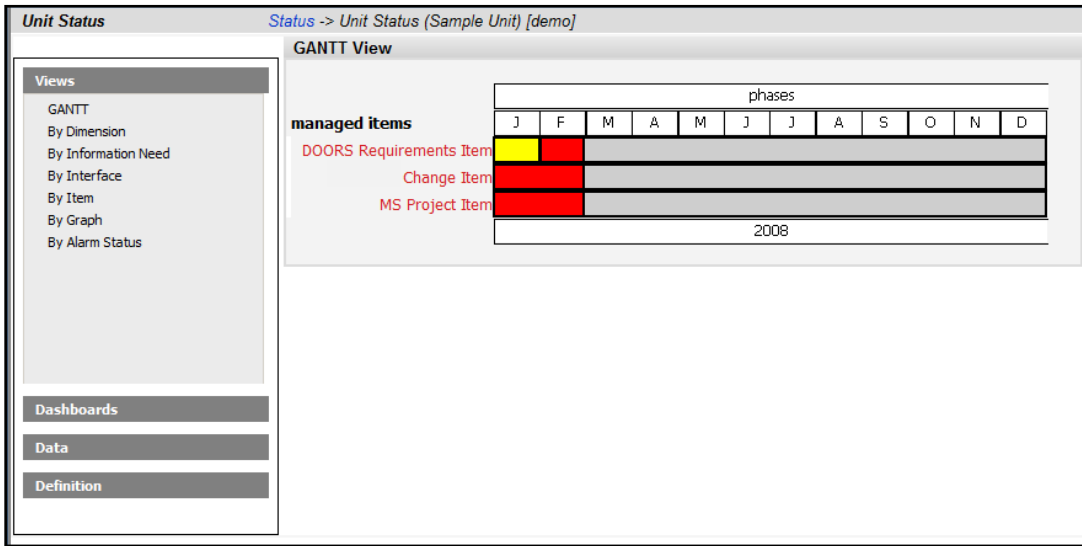


Figure 3

Unit data is presented in a variety of ways with *views*. This flexibility tailors the data presentation to the manager's style of managing instead of forcing the manager to rely on a prescribed way of receiving information. Use the navigation bar to review each of the available *views*: Gantt, By Dimension, By Information Need, By Interface, By Item, By Graph, and By Alarm Status.

Click on the different types of view on the navigation bar in the left pane to explore the different views available (Figure 4).

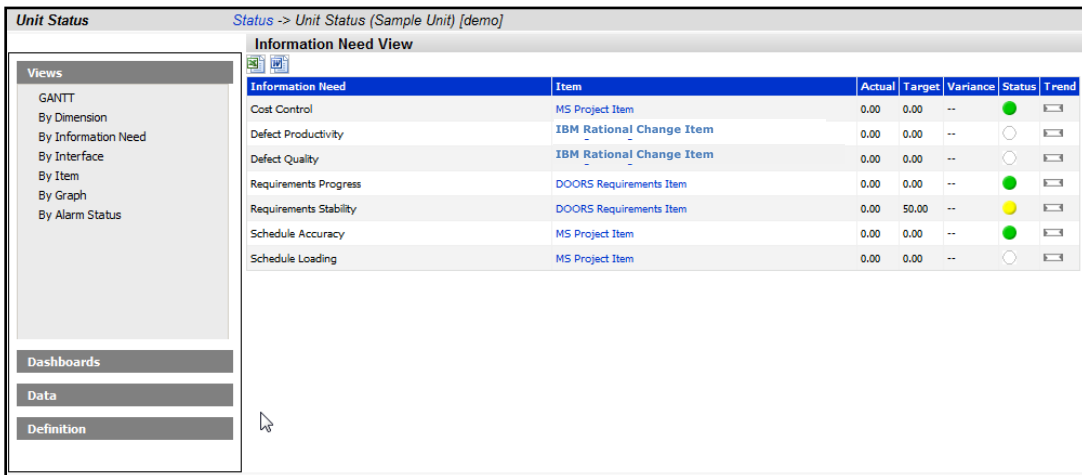


Figure 4

### 3.3.3. Item Data

Unit Dashboards are created automatically as items and information needs are assigned to a basic unit. Every item assigned to the unit has its own dashboard, with the graphs in the dashboards being defined by the information needs that were assigned.

Click the Dashboards panel on the left and then click the *IBM Rational Change Item*. All graphs associated with the relevant Information Needs will be displayed in the Dashboards panel (Figure 17).

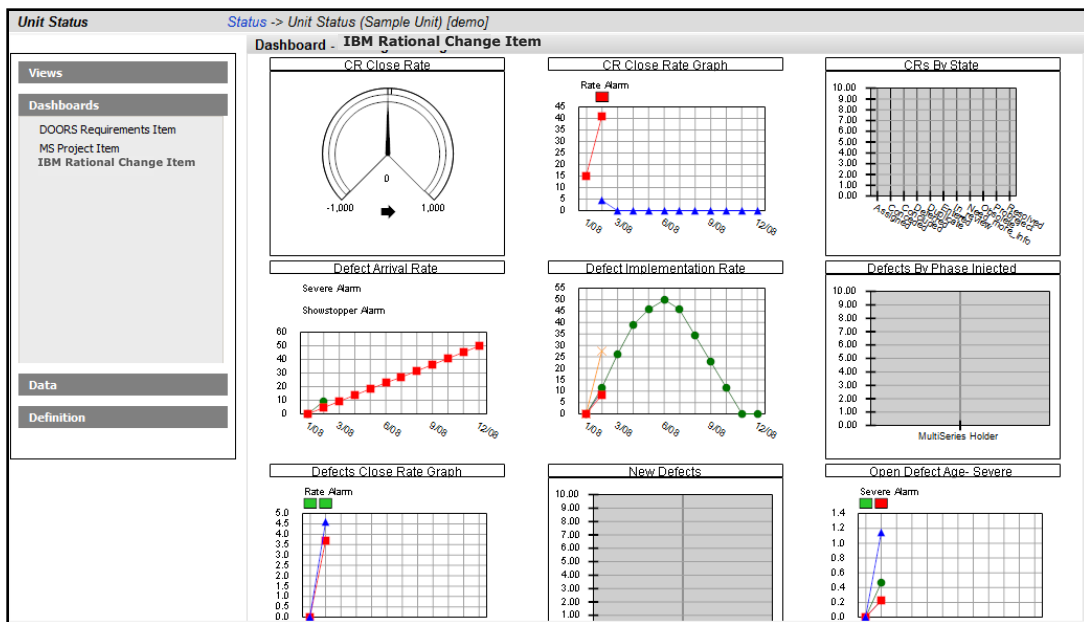


Figure 5

Take a moment to look at the graphs.

Once you're done, click on *MS Project Item* in the Dashboards panel on the left side and you will see graphs similar to those on Figure 18.



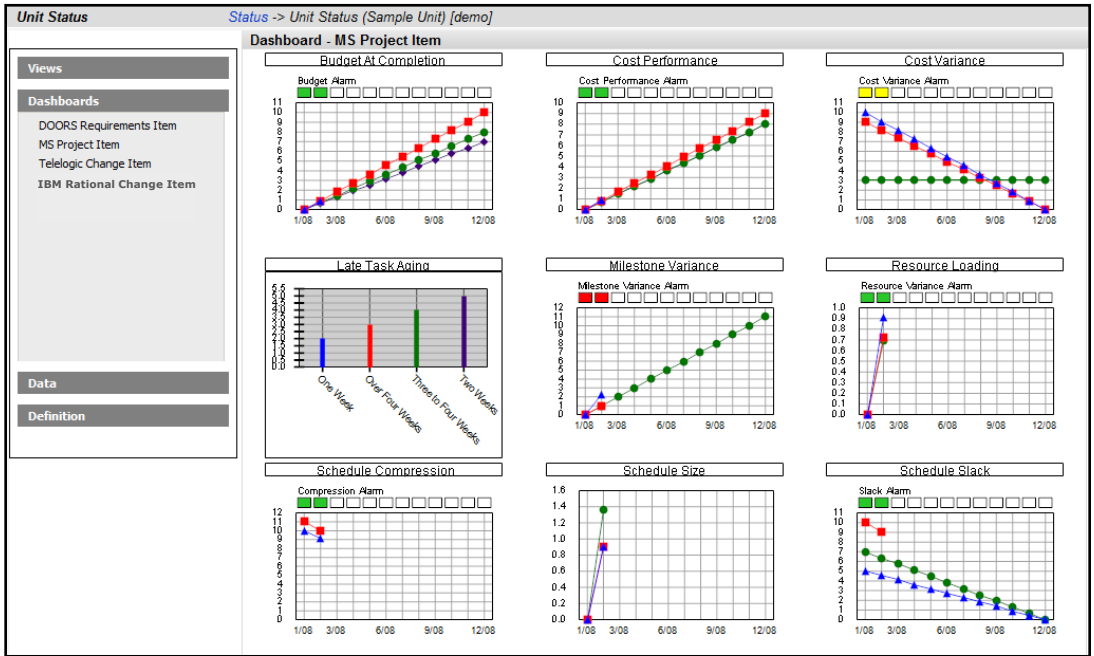


Figure 6

And finally, click on *DOORS Requirements Item* in the Dashboards panel. All Graphs associated with the relevant Information Needs will be displayed in the Managed Item Dashboard (Figure 19).

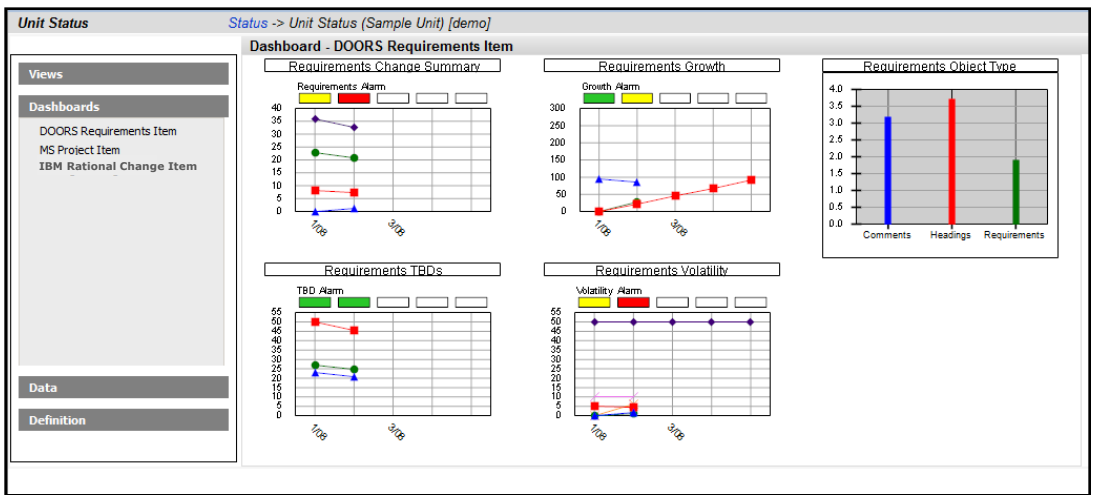


Figure 7

### 3.4. Overview of Queries and Graphs

Click on the Requirements Volatility graph in the DOORS Requirements items to view the graph in more in-depth detail on the Graph Display Page (Figure 20).

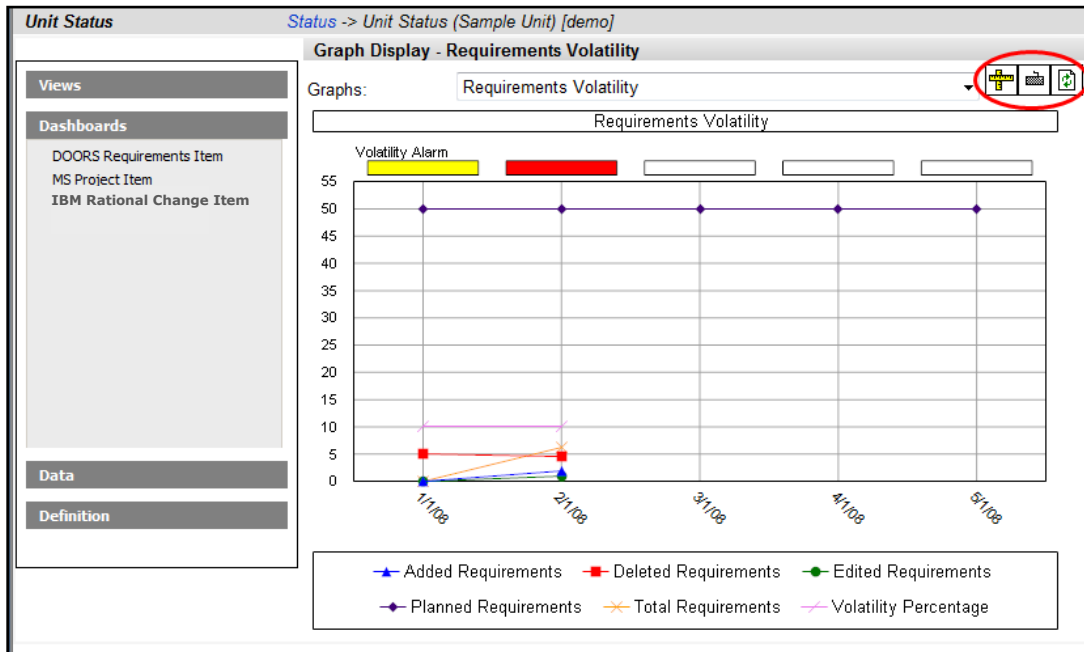


Figure 8

On the Graph Display Page you see three icons on top of the graph, representing the three options available to modify this graph.

The first icon (ruler icon) allows the user to manage the format and visual layout of the graph. The second icon (keyboard icon) allows the user to define and enter data for the graph. The third icon (refresh) allows the user to refresh the graph for the most current data and status available.

Notice in our sample graph, Requirements Volatility, we have an alarm for Volatility, a planned series, and several series based on collected data. The flexibility in graph formatting, as well as the large selection of graph types, allows a manager to view status and data in a manner which is most productive for their management style.

### 3.5. Writing Queries

This section provides technical information for developing queries defined within an interface in the IBM Rational Dashboard portal.

### **3.5.1. Sample Queries**

This subsection contains sample SQL queries that demonstrate various SQL techniques for counting data in a transform database.

The following query counts the number of records in a table:

```
SELECT count(*) FROM ReqData
```

The following query counts everything in a table where the table name is determined just before the query is run with the %TABLE% tag:

```
SELECT count(*) FROM %TABLE%
```

The following query counts all objects for a single period:

```
SELECT count(*) from %TABLE% where (g_collectdate = %COLDATE%)
```

The following query counts all objects for a single period in a single module:

```
SELECT count(*) from %TABLE% where (g_collectdate = %COLDATE%) and  
(g_itemid = %ITEMID%)
```

The following query counts all objects that were created during a single period

```
SELECT count(*) from %TABLE% where (g_collectdate = %COLDATE%) and  
(created_on >= %STARTDATE%) and (created_on <= %ENDDATE%) and  
(g_itemid = %ITEMID%)
```

The following query counts requirements that were edited in a single period

```
SELECT count(*) FROM %TABLE% WHERE (g_collectdate=%COLDATE%)  
and (ObjType='R') and (modified_date IS NOT NULL) and  
(modified_date >= %STARTDATE%) and (modified_date <= %ENDDATE%)  
and (g_itemid = %ITEMID%)
```

### 3.5.2. Query Tags

Text in the form “%ZZZ%” is called a tag, and allow a query to be written such that they can apply to as many needs as possible (without having to write a new query for every possible variation of your analysis). Each tag is evaluated at run-time and replaced with the appropriate value.

Note that a query containing a tag, such as %TABLE% is not a valid SQL command – if you run a query which contains a tag in a SQL query tool, the SQL statement will generate an error. The Portal Query Test page allows you to run SQL statements containing tags. In addition, it helps you uncover valid values for tags and also to substitute different values for the tags.

The following table describes allowable tags:

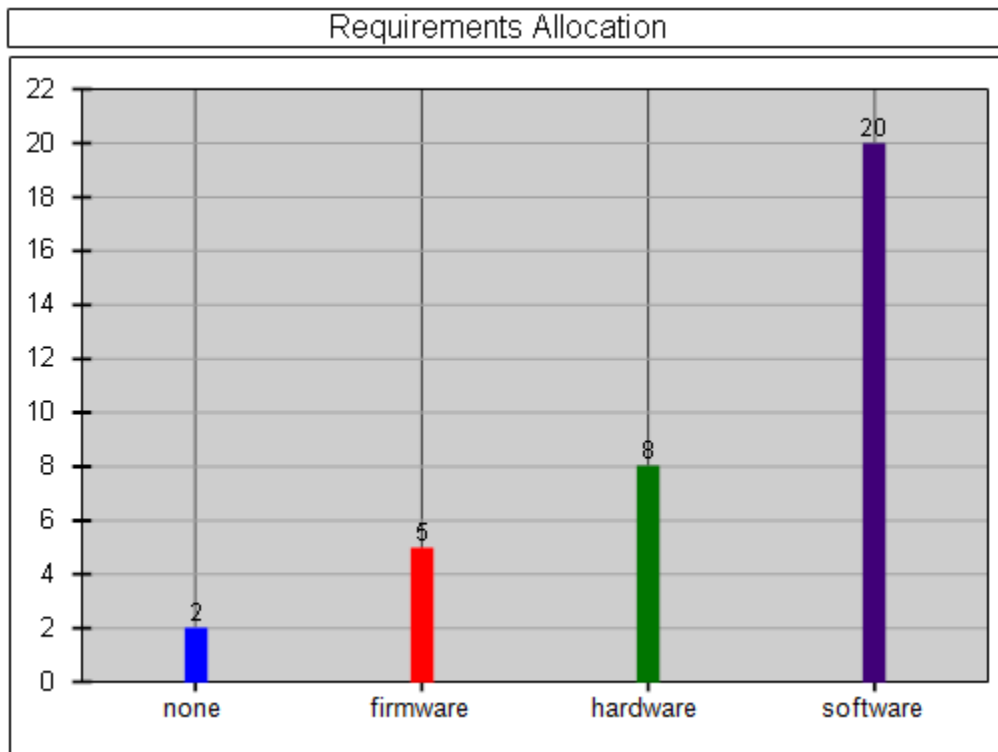
Tag	Purpose	Notes
%TABLE%	database table name	The table name of the first field set in the interface. If there is one field set, this tag can be used in all queries. If there is more than one field set (and more than one table in the interface), then this tag contains the name of the first table.  Example: SELECT ... FROM %TABLE%
%COLDATE%	collection date of schedule period	This tag allows queries to use the collection date of the schedule period being evaluated. This tag can be compared to field in the table, for example to select records before or after the current collection date. This tag is usually compared to the g_collectdate field, whose data is automatically populated by IBM Rational Dashboard when each record is stored.  Example: SELECT COUNT(*) FROM %TABLE%

		WHERE g_collectdate=%COLDATE%
%STARTDATE%	Start date of schedule period	<p>This tag allows queries to use the start date of the schedule period being evaluated. This tag can be compared to data field(s) in the table, for example to select records before or after the start of the schedule period.</p> <p>When used with the start date tag, you can select records of interest in the current period.</p> <p>Example:  SELECT COUNT(*) FROM %TABLE%  WHERE change_date BETWEEN  %STARTDATE% and %ENDDATE%</p>
%ENDDATE%	End date of schedule period	<p>This tag allows queries to use the end date of the schedule period being evaluated. This tag can be compared to data field(s) in the table, for example to select records before or after the end of the schedule period.</p> <p>When used with the start date tag, you can select records of interest in the current period.</p> <p>Example:  SELECT Count(*) FROM %TABLE% WHERE  change_date BETWEEN %STARTDATE%  AND %ENDDATE%</p>
%UNITID%	Unit id	Integer identifier of the current unit
%ITEMID%	item id	<p>Integer identifier of the current item</p> <p>This tag is used to select records applicable to one item. Commonly, a single table in a transform database is used to store data for multiple items, for example requirement objects from multiple DOORS modules, or tasks from multiple Microsoft Project schedules.</p> <p>Example:  SELECT COUNT(*) FROM MsPTasks</p>

		WHERE g_itemid=%ITEMID% AND task_actual_start > task_plan_start
%GRAPHSTARTDATE%	start date of graph	The period start date of the first schedule period in a graph Typically used to allow a SLIP chart to determine the correct value for actual series data points  Example: SELECT DATEDIFF(day, %GRAPHSTARTDATE%, schedule_end_date) AS DaysToDelivery FROM %TABLE%
%A()%	attribute value	Retrieves an attribute value from the current unit  Example: %A(Project Cost)%

### 3.5.3. Using Multi-Series Queries

A multi-series query is used to simplify setting up a graph that has more than one series which uses similar queries. For example, consider a graph of requirements by allocation type where there are four types of allocation (“none”, “software”, “hardware” and “firmware”). The graph would have four series, one series for each of the allocation type, and look like the following:



The query used for each series would be very similar (almost identical), except that each query would contain a slightly different WHERE clause such as “WHERE Allocation=’none’”, “WHERE Allocation=’software’” and so on. In this example, you can see that it would be time consuming to set up four series and then four single-series queries for the allocation types. The multi-series query simplifies this scenario.

In IBM Rational Dashboard , queries are SQL database statements that create numeric values from data stored in a transform database. Queries are run at the end of the data collection process (after data has been inserted and updated in the database) and also when a refresh is performed through the user interface. When the query is executed, it returns a numeric value which is stored in the Portal database and (usually) plotted as a series value on a graph.

There are two types of queries: single-series query and multi-series query. The most common, the single-series query, returns one value (for a *single series*) when it is run. To use a single-series query, you create a data series and assign a single-series query as the source of data. The single-series query returns one numeric value per period as the query is run for each period. For example, the “number of new defects” is a common single-series query which calculates the count of new defects created during each period.

### 3.5.3.1. How the Multi-Series Query Works

The multi-series query performs the counting operation of the single-series query but also creates other series in the graph where it is defined. In the above example, multi-series would create 4 series, one for each allocation type. A multi-series query is assigned to a series in the graph, and the “real” allocation type series are created automatically when the multi-series is run. The series created by the multi-series are called “reference series”. It is recommended that you only have one multi-series query per graph.

Compared to the single series query, the multi-series requires a little more configuration, as there must be rules for determining how to create the reference series. The edit page for the multi-series query supports two options for determining how to create the reference series:

- Query database for series
- Specify series in a list

Query Database for Series - When the “query database for series” is used, the multi-series will create as many series as there are different (in SQL query language “distinct”) values in the database field where the data is stored. In this option, you must specify a secondary SQL statement that determines the distinct values. Using the example of requirements by allocation type, the multi-query series would query the database for the allocation types that are currently in the database. For each distinct allocation type found, a new series is created in the graph. For example, consider the following collected data:

Allocation	Requirement Title
Software	Capture system events in memory log
Software	Periodically check available memory
Hardware	Provide warm restart button

With data shown above, notice that the multi-series query will find and create two distinct series, for “software” and “hardware”. Even though we know there are four possible allocation types, the multi-series query found 2, since the data only contains those 2. As additional data is collected (and requirement records are added with additional allocation types), the multi-series query will create additional series for them.

When the list of values dynamically grows and you want to see them all in a graph, this option, “query database for series” is the best choice.



Specify Series in List – This option allows you to specify the list of reference series for the multi-series query. Each time the multi-series query is run, the reference series in the list will be created in the corresponding graph. If you change the list of reference series, the graphs will be updated next time data collection is performed or the graph is refreshed.

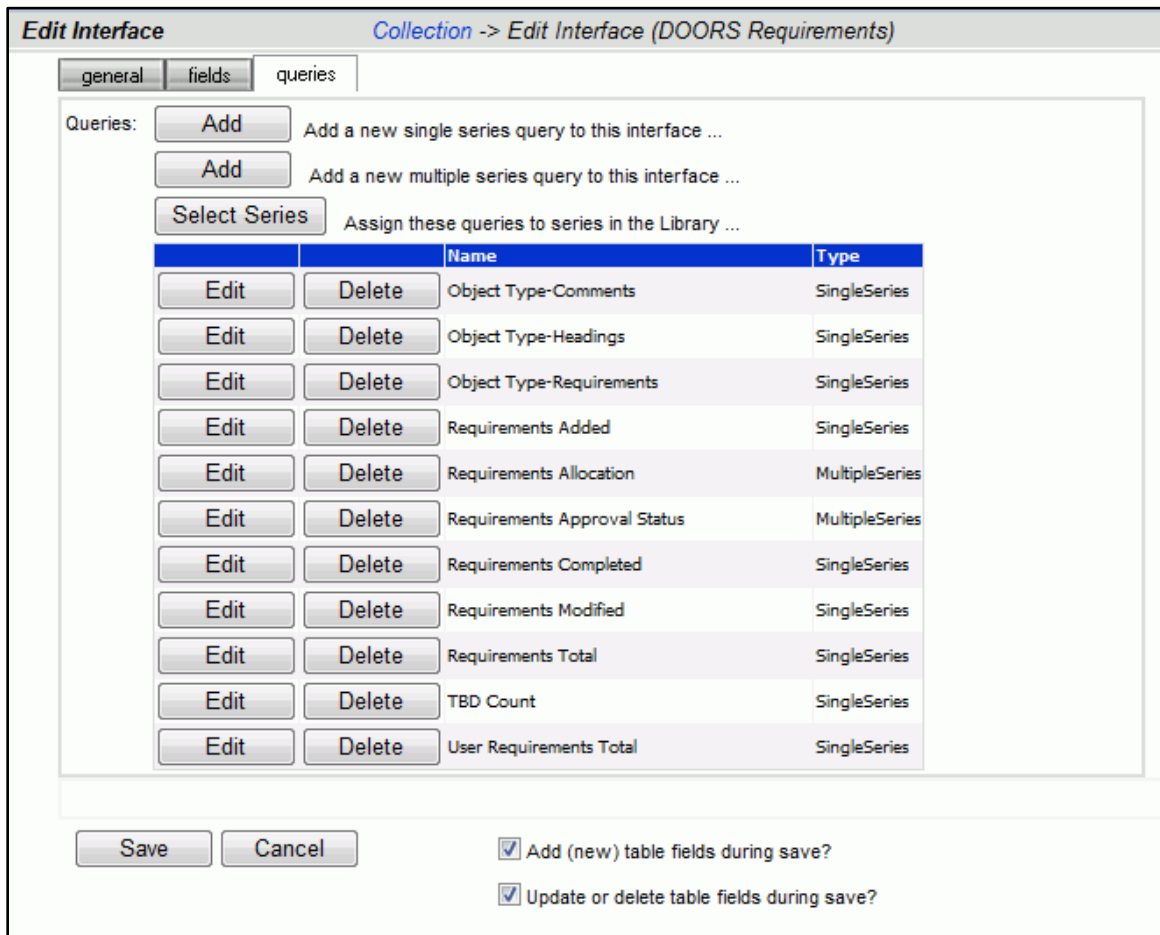
In this option, you enter a list of the reference series. Each reference series consists of a series name and a value. Each entry in the list becomes a series in the graph that contains the multi-series query.

Since the list specifies the reference series to expect, there is the chance that the data in the associated table contains more than just the values you specify. For example, if we enter the four allocation types in our example, the table could contain a data value for “operator” (which was not in our list of four in this example). To detect data such as this, the multi-series query supports something called the “other” series. When this option is enabled (using a checkbox titled “Include ‘other’ series?”), the multi-series query will create a series titled ‘other’ and include a count of data items that were not counted using the reference series. When this option is used, you must enter a SQL statement that selects and counts the records to be considered ‘other’.

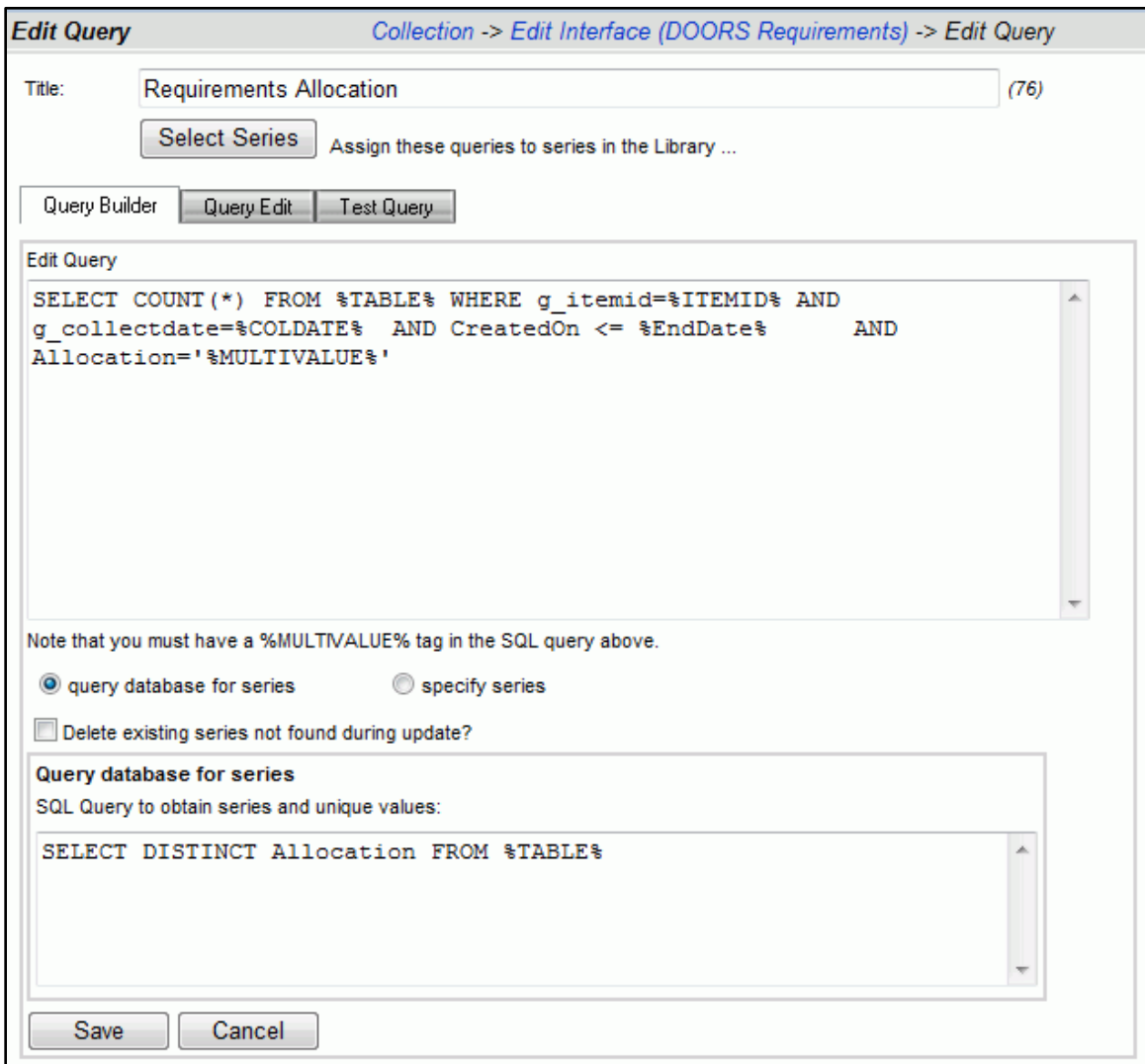
When the list of values does not change often, this option, “specify series in list” is the best choice.

### 3.5.3.2. Example Multi-Series Query Using Database Values

A multi-series query is created as a query in an interface – an interface such as the ‘IBM Rational Doors’ interface typically contains both single-series and multi-series queries as shown below.



After you press the “Add” button for adding a multi-series query, you will see the “Edit Query” page, as shown below for the “Requirements Allocation” query which is a multi-series query in the “IBM Rational Doors” interface.



Note that there is one replacement tag, %MULTIVALUE% which is not used for a single series query. Since the purpose of the multi-series is to simplify creating and counting multiple series when they differ by only one term – it is the %MULTIVALUE% replacement tag that performs this function. When the SQL query for a multi-series query is run, it will be run multiple times, one for each reference series with the value of the %MULTIVALUE% tag replaced by the corresponding value from the reference series.

To understand how this works, let's walk through using the requirements allocation example. For this example, assume the list of reference series is obtained by querying the database. The SQL statement to do this is contained in the edit box labeled "Query for list of unique values for series" and is run before the multi-series query to obtain the

list of reference series. From the image above, the SQL statement to get a list of allocation status values is as follows:

```
SELECT DISTINCT Allocation from %TABLE%
```

In this SQL statement, the “DISTINCT” keyword will return a list of unique values for the data in the Allocation field. This list becomes the reference series for our multi-series query. In this example, assume that the query returns the following list of string values for the allocation field:

```
“None”, “Software”, “Hardware”, “Firmware”
```

Once the list of reference series is obtained, the multi-series query in the wizard will be run. For this example, the “raw” SQL for the multi-series query uses the following:

```
SELECT COUNT(*) FROM %TABLE% WHERE  
Allocation='%MULTIVALUE%' AND  
CreatedOn<=%ENDDATE% AND  
g_item=%ITEMID%
```

So, using the results of the first query (which returns the list of reference series), IBM Rational Dashboard will execute the multi-series query four times, one for each of the four reference values. The resulting queries look like the following:

```
SELECT COUNT(*) FROM DoorsReq WHERE Allocation='none' AND  
CreatedOn<='20080601' AND g_item=123  
SELECT COUNT(*) FROM DoorsReq WHERE Allocation='software' AND CreatedOn  
<='20080601' AND g_item=123  
SELECT COUNT(*) FROM DoorsReq WHERE Allocation='hardware' AND CreatedOn  
<='20080601' AND g_item=123  
SELECT COUNT(*) FROM DoorsReq WHERE Allocation='firmware' AND CreatedOn  
<='20080601' AND g_item=123
```

(Other tag values, such as %ITEMID%, have sample data.)

The resulting graph will have four series automatically added, for a total of five with the multi-series query (which is also technically a series). Each of the four reference series will have a value set for the corresponding collection period. The graph may contain other series, such as equations and alarms that analyze the data in the multi-series query.

For example, the requirement by allocation type graph may contain a data series with a source of equation that uses the equation graphsum(“\”), which provides a sum of all actual data series in the graph. As reference series are added and removed, the graphsum equation adds up all available series, and does not require any maintenance.

### 3.5.3.3. Example Multi-Series Query With Specified Series List

When using the option for “specify series”, you are able to:

- Provide a series title rather than using the value which comes from the database
- Limit the series which are created to those that you want or expect
- Specify a single value to use when substituting the %MULTIVALUE% tag
- Automatically generate an ‘other’ series which counts the number of records which are not in the list

As shown below, the example from above has been modified to use the “specify series” option rather than the “query database for series” option.

The screenshot shows the Query Builder interface with the following configuration:

- 1 Data from:** DOORSReqs database table
- 2 Result is:**  a count of  a sum of CreatedOn (Default Set)
- 3 Filters are:**  the current item  for the current period
- 4 With terms:** CreatedOn <= %EndDate% Allocation=%MULTIVALUE% (with a delete button)
- Add new query term:** CreatedOn (Default Set) = [ ] Start Date [ ] (with add and insert buttons)
- 5 Multi Series:** Note that you must have a %MULTIVALUE% tag in the SQL query above.
  - query database for series  specify series
  - Delete existing series not found during update?
  - Specify series**
    - Include 'other' series?
    - add Add a series to the list ...

		Series Title	Field Value
edit	delete	none	na
edit	delete	software	software
edit	delete	hardware	hardware
edit	delete	firmware	firmware

Notice that at the bottom of the screen, there are four reference series specified for “none”, “software”, “hardware” and “firmware”. Along with each reference series, there is a value which will be used for the %MULTIVALUE% tag. The values in the “field value” column are string values but do not contain the enclosing single quote mark. (In the SQL language, when you check whether a field contains a string value, you would enter fieldname='value', with the value surrounded by single quote marks.) This is

because the term shown at item 4 (in the image next to “4. With Terms”) is defined as Allocation = ‘%MULTIVALUE%’ which provides the enclosing quote marks.

Notice that the first reference series in the list contains a title of “none” but uses a different value “na”. This allows you to substitute a value to search for different than the series title. You could also substitute a more readable or shorter title for the series if needed.

The operation of the multi-series query is much the same as the previous example, except instead of running the first query to obtain a list of distinct values, the values are simply extracted from the reference series list.

### **3.5.4. Setting Up a Snapshot Grid to Show Tabular Data**

Tabular data can be displayed through a graph type called a Snapshot Grid. The snapshot grid is a data table which displays a set of records of interest to managers. An example of an inspection grid is shown below:

Inspection Date	Document Title	Pages Inspected	Total Errors	Total Time
1/1/2008 12:00:00 AM	JTL_INSP	5	10	15
2/1/2008 12:00:00 AM	RLP_INSP	7	14	21
3/1/2008 12:00:00 AM	SSB_INSP	10	20	30
4/1/2008 12:00:00 AM	MVR_INSP	15	30	45
5/1/2008 12:00:00 AM	JTL_INSP	20	40	60
6/1/2008 12:00:00 AM	RLP_INSP	25	50	75
7/1/2008 12:00:00 AM	SSB_INSP	28	56	84
8/1/2008 12:00:00 AM	MVR_INSP	32	64	96
9/1/2008 12:00:00 AM	TSV_INSP	36	73	108
10/1/2008 12:00:00 AM	JTL_INSP	40	81	120
11/1/2008 12:00:00 AM	RLP_INSP	45	89	135
12/1/2008 12:00:00 AM	SSB_INSP	49	97	147

Any data records (in one of the data mart databases) can be displayed in a Snapshot Grid. For example, if an information need contained a run graph of software inspections held each week, you could define a snapshot grid to show the details for all completed inspections.

A snapshot grid is actually a type of graph, with the graph type set to Snapshot grid. In the Library for example, the snapshot grid is one of the available Graph types (as shown in the figure below). You can add any number of snapshot grids to an Information Need in the Library tab.

**Library Graph** *Library -> Information Need (Inspection Data) -> Graph (New graph 5/5/2008 3-29-27 PM)*

graph

Title:  (318)

Description:

Type:  ▼

To display a snapshot grid, follow three steps:

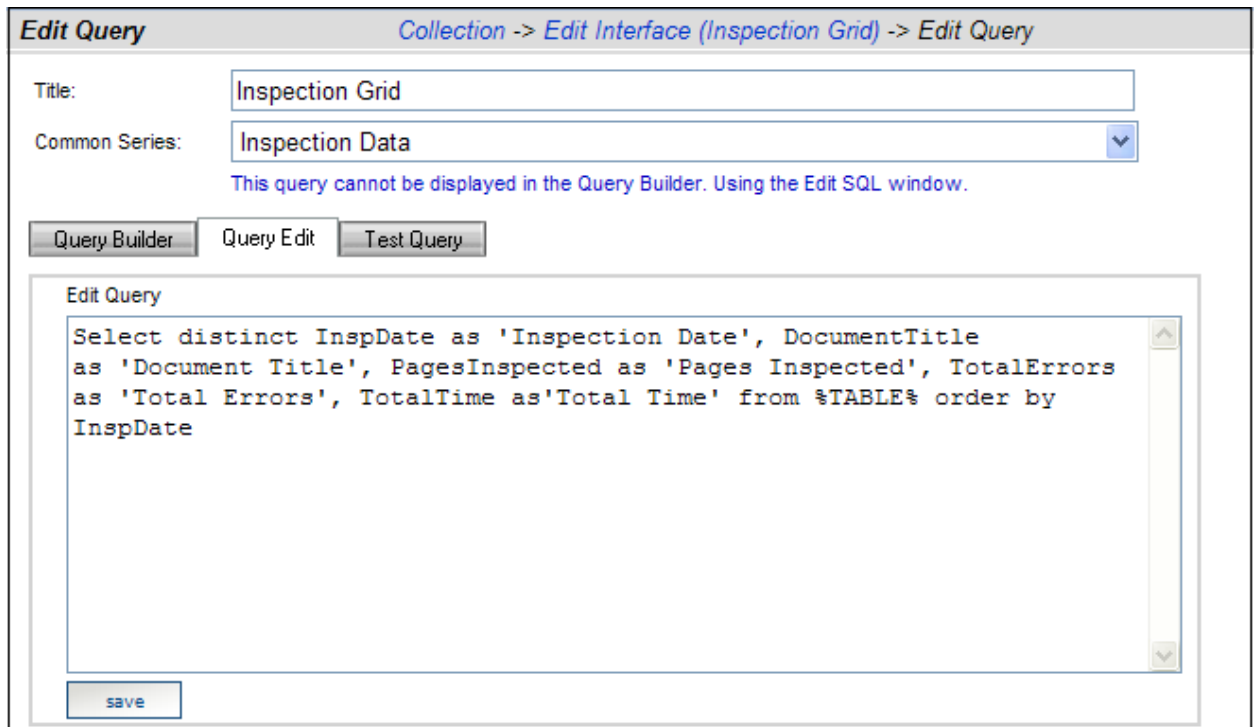
1. Create a SQL query in an interface (Collect tab)
2. Create a snapshot grid in an Information Need (Library tab)
3. Assign the Information need to your Unit (Status tab)

Once the snapshot grid and associated query is defined, you assign an information need to your unit and appropriate data (that is, data which meets the conditions of your query) will be displayed. Detailed instructions for each step are below.

#### **3.5.4.1. Step 1: Creating a SQL Query within an Existing Interface.**

To add a SQL Query to the Interface, follow these steps:

1. Click on the Collection tab.
2. In the gray panel, click on Interfaces.
3. Click on the edit button for the Interface in which you want to add a snapshot grid.
4. In the Edit Interface page, click on the queries tab.
5. Click on add (add a new single series query to this interface) button.
6. Enter a Title for the snapshot grid query.
7. Enter the query by using the query builder or the query edit tab.
8. Save your changes.



In the above diagram, notice that the fields (or columns) are defined using the “as” clause so that the headings in the snapshot grid have a more readable/understable column heading.

### 3.5.4.2. Step 2: Adding a Snapshot Grid to an Information Need.

To add a snapshot grid to your information need, follow these steps:

1. Click on the library tab. Click on the Edit button for the Information Need where you want to add the data table.
2. Click on the graphs tab in the Information Need. Click on the add button.
3. In the graph tab, enter the following information:
  - a. Title - enter the name for the snapshot grid.
  - b. Type – select snapshot grid from the drop down. (This action will cause the page to reload with only the graph tab available).
  - c. Frequency – click on the Schedule-driven option.
  - d. Measure – click on the measure that is associated with the information need.
  - e. Options – check the box to engage the option.
4. Save your changes.



**Library Graph** Library -> Information Need (Schedule Accuracy) -> Graph

graph

Title:  (237)

Description:

Type:

Frequency:  Schedule-driven

Do not propagate to unit?

Data:

Options:

- Show header as first row?
- Use alternating row colors (white/gray)?
- Insert a subtotal column on right side?
- Insert a subtotal row at bottom of grid?
- Show horizontal grid lines?
- Show vertical grid lines?

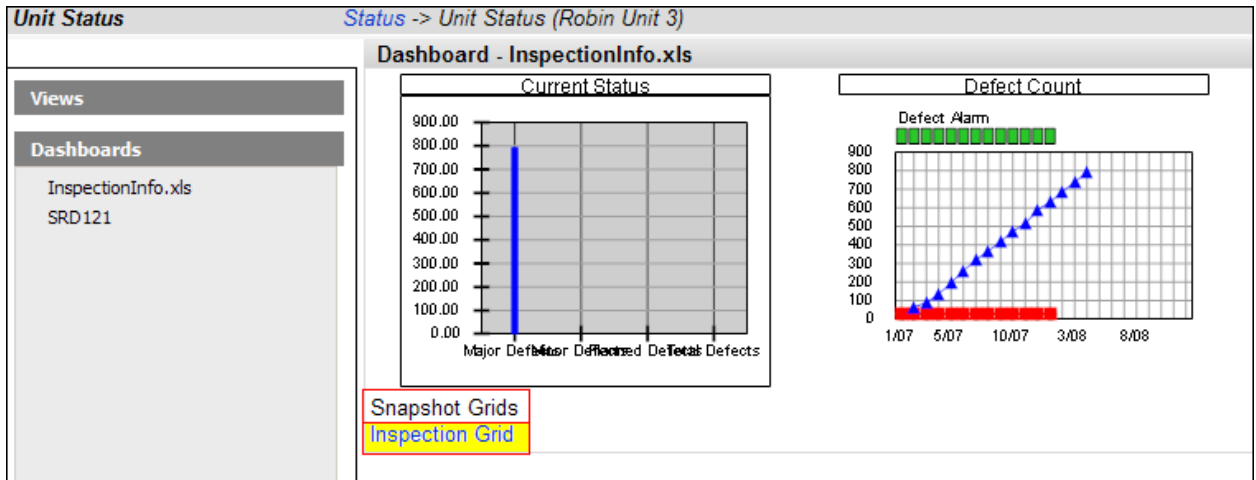
### 3.5.4.3. Step 3: Assigning the Information Need to your Unit

To see the Snapshot Grid in your Dashboard, follow these steps:

1. Click on the Status tab and click on the Unit in which you wish to add the snapshot grid.
2. Click on the Details icon to open Unit.
3. In the navigation panel on the left side, click on the Definition group and then the Information Needs command.
4. From the drop down list, select the Information Need that contains the snapshot grid.
5. Select the Interface associated that contains the query for the snapshot grid.
6. Check boxes: Assign information need to existing items and Perform refresh after assignment.
7. Click on the assign button.
8. The new Information Need will be listed in a grid directly below the Information Need Definition.

9. In the navigation bar on the left hand side, click the Dashboard group and then click the command that contains the information need.
10. In the Dashboard, under Snapshot Grids, click on the link. This will open the snapshot grid.

Each dashboard contains a set of graphs, followed by a list of snapshot grids. Each item in the list is a hyperlink that will open the selected grid. In the image below, there is one snapshot grid, called "Inspection Grid" in the "InspectionInfo" dashboard.



Once you click on the hyperlink for the grid, the snapshot grid for inspection data will be displayed as shown below (if you used the same SQL query as provided above).

Inspection Date	Document Title	Pages Inspected	Total Errors	Total Time
1/1/2008 12:00:00 AM	JTL_INSP	5	10	15
2/1/2008 12:00:00 AM	RLP_INSP	7	14	21
3/1/2008 12:00:00 AM	SSB_INSP	10	20	30
4/1/2008 12:00:00 AM	MVR_INSP	15	30	45
5/1/2008 12:00:00 AM	JTL_INSP	20	40	60
6/1/2008 12:00:00 AM	RLP_INSP	25	50	75
7/1/2008 12:00:00 AM	SSB_INSP	28	56	84
8/1/2008 12:00:00 AM	MVR_INSP	32	64	96
9/1/2008 12:00:00 AM	TSV_INSP	36	73	108
10/1/2008 12:00:00 AM	JTL_INSP	40	81	120
11/1/2008 12:00:00 AM	RLP_INSP	45	89	135
12/1/2008 12:00:00 AM	SSB_INSP	49	97	147

### 3.5.4.4. Writing Queries for Snapshot Grids

When setting up a measurement program, it is often helpful to see a table of supporting data that explains what caused the behavior of a series. For example, if the series

“number of tasks started late” goes up one month unexpectedly, then a manager may want to see which tasks are late. In IBM Rational Dashboard, a snapshot grid can display data in a tabular form, such as a list of late tasks. Snapshot grids can be added to information needs, along with graphs, to provide supporting details for managers.

A snapshot grid uses a single query as its data source. The query is added to the interface, along with other single-value queries. The snapshot grid query uses the same format and tags as other queries. The query for a snapshot grid must define the data columns (or fields) in the order that you want them to appear in the snapshot grid, and also order the resulting records in the desired fashion. Unlike a single-value query for a data series, the query for a snapshot grid does not use a SUM, COUNT or other aggregate function to return a single value.

For example, below is a snapshot query that returns the tasks which started late in the last period.

```
SELECT task_title, task_start_date as PlannedStart, task_act_start as ActualStart FROM %TABLE% WHERE g_itemid=%ITEMID% AND (task_start_date BETWEEN %STARTDATE% AND %ENDDATE%) and task_start_date < task_act_start ORDER BY task_start_date
```

The query returns all records in a schedule that were planned to start in the last period and were started late (that is, the actual start is after the planned start date). Since the column names are used as headers in the snapshot grid, the “AS” clause is used to provide more readable headers than the column names. Finally, the “ORDER BY” clause sorts the records by planned start date from the earliest (i.e. start of the period) to the latest (i.e. end of the period).

### **3.5.5. Understanding the !FIELD! Tag in the Collector**

When the Collector is run, it obtains the interface definition for each data source that is being run. The interface definition contains a list of field sets, where each field set contains a table name, a field set replacement tag and a list of fields. The Collector then builds whatever interface code/script/API is required for the data source and runs a process to collect the desired data. For example, when collecting from IBM Rational Doors, the Collector collects object attributes using DXL. When collecting from a SQL database, the Collector collects data base fields (sometimes called columns) using a SQL statement.

Consider when the Collector runs to collect data from a SQL, Oracle or ODBC database: after the Collector obtains a list of items, the Collector must collect data for each item by generating a SQL statement. This SQL statement must request the fields that were

defined in the interface. To simplify the coding of the SQL statement, the Collector provides a replacement tag that looks like !FIELDS!. This tag is replaced with a comma separated list of all the field names that were defined in the interface.

Looking at an example of the !FIELDS! tag, if an interface for a SQL database contains three fields (TaskName, StartDate and EndDate), the Collector could be configured with a “Get Details” query that looks like the following:

```
SELECT !FIELDS! FROM TaskTable WHERE ProjectId=!ITEM!
```

The two tags above, !FIELDS! and !ITEM!, are filled in at run-time by the Collector, to look like the following (though the ITEM value depends on actual data):

```
SELECT TaskName, StartDate, EndDate FROM TaskTable WHERE ProjectId=1
```

The !FIELDS! tag is very useful when the syntax of the collection allows for a comma separated field list, as is the case with a SQL statement.

### **3.5.6. Understanding the !SETTAG! in the Collector**

In an interface, the field set replacement tag is an optional field in a field set. Each field set has it's a replacement tag, a table name and a set of fields. During collection, field set information is passed to the Collector. Some parameters in a data source are replaced at run-time by the field set replacement tag. In particular, the replaceable parameters are the “getlist” and “getdetails” queries from each type of interface.

In the Collector data source, you may enter the text !SETTAG!, which signals that the Collector should replace the text !SETTAG! with the text entered into the that was entered into the Portal. You may use the !SETTAG! to replace either part of the data source parameter, as in:

```
SELECT * FROM ProjectList WHERE !SETTAG!
```

Or, you may use the !SETTAG! to replace the entire parameter, as in:

```
!SETTAG!
```

If there are errors with this tag replacement, you should consult the Collector log file for more details.

#### **3.5.6.1. Using the Field Set Replacement Tag for SQL Joins**

When you need to combine data from more than one table, you are typically performing what is called a join query. In a join query, you must use an identifier (which

is usually not human readable) to look up a readable or descriptive string from another table. The other table(s) is frequently referred to as a look-up table.

To use a JOIN statement, you use the field set replacement tag, !SETTAG!. The !SETTAG! is basically a method to write more advanced SQL statements without being encumbered by the simplicity of the !FIELDS! tag. An example, presented in the next subsections, walks through how to configure and run a SQL join using the !SETTAG!.

### 3.5.6.2. A Sample Database

In this sample, the external application is a Microsoft SQL database. The database contains a table called "IssueData", from which we would like to collect data. The IssueData table has four fields and there are four records in the table, as shown below.

	IssueTitle	CreateDate	ProjectId	PriorityCode
	Warm Restart Generates Cold Start Errors	1/1/2008 12:00:00 AM	1	4
	Disable 1553B in UPS Mode	1/1/2008 12:00:00 AM	1	3
	Altitude Assigned to Surface Track	6/6/2008 12:00:00 AM	1	2
▶	Weight Calculation Uses Wrong Alloy Coeff	2/2/2008 12:00:00 AM	1	1
*	NULL	NULL	NULL	NULL

Notice that the table has a "ProjectId" field but we cannot tell the Project name or title. Also, there is a "PriorityCode" field but we cannot tell what the priority is (such as low, medium or high).

The database also has two lookup tables, one for the project title and one for the priority title. Both tables contain an Id field and a text field. The project look-up table is shown below.

	ProjectId	ProjectTitle
	1	Project 1
	2	Project 2
▶	3	Project 3
	4	Project 4
	5	Project 5
*	NULL	NULL

Similar to the project table, a priority look-up table is shown below. The priority table is not used in the sample (described in subsequent paragraphs), but can be used as a follow-on to expand the sample from a two-table to a three-table join.

	PriorityCode	PriorityTitle
	1	low
	2	medium
	3	high
▶	4	critical
*	NULL	NULL

### 3.5.6.3. Configuring an Interface to Perform a Join

The IBM Rational Dashboard Portal provides the means to define an interface, the field sets within the interface and the fields within each field set. An interface called “Sample SQL Join” is shown below with the default field set being configured to perform a SQL join on one table.

**Edit Interface** Collection -> Edit Interface (Sample SQL Join)

general fields queries

List of Sets:  
 Default Set  
 add

Title: Default Set  
 Database Table: JoinResults

[-] Parameter Replacement Tag:  

```
SELECT i.IssueTitle, i.ProjectId,
p.ProjectTitle
from IssueData as i , luProjects as p
WHERE (i.ProjectId = p.ProjectId) AND
(i.ProjectID = !ITEM!)
```

Fields in Selected Set:  
 add  
 IssueTitle  
 ProjectId  
 ProjectTitle  
 PriorityCode  
 Priority

**Selected Field Properties**  
 Do collect from source?  
 Source Attribute: PriorityCode  
Changes below modify the associated database table. Changes of: 1) string type to non-string; 2) string to shorter string; or 3) allow null to not allow null; will result in the field being dropped and re-added causing loss of data.  
 Table Field: PriorityCode  
 Type: Integer  
 Value:  Allow null - default optional    default value:   
 Don't allow null - default required

apply delete

save cancel

Add (new) table fields during save?  
 Update or delete table fields during save?

In the interface (above), the fields defined for the field set named “Default Set” are shown in the following table.

Collect ?	Source Attribute	Table Field	Type
Yes	IssueTitle	IssueTitle	integer
Yes	ProjectId	ProjectId	integer
Yes	ProjectTitle	ProjectName	string 50
No	PriorityCode	PriorityCode	integer
No	Priority	Priority	string 10

Notice that the PriorityCode and Priority attributes are defined in the interface but marked as “Do not collect”.

In the interface above, the Parameter Replacement Field (i.e. !SETTAG!) contains a complete SQL statement, which is valid on the source (with the exception of the !ITEM! tag which is filled in by the Collector at run-time):

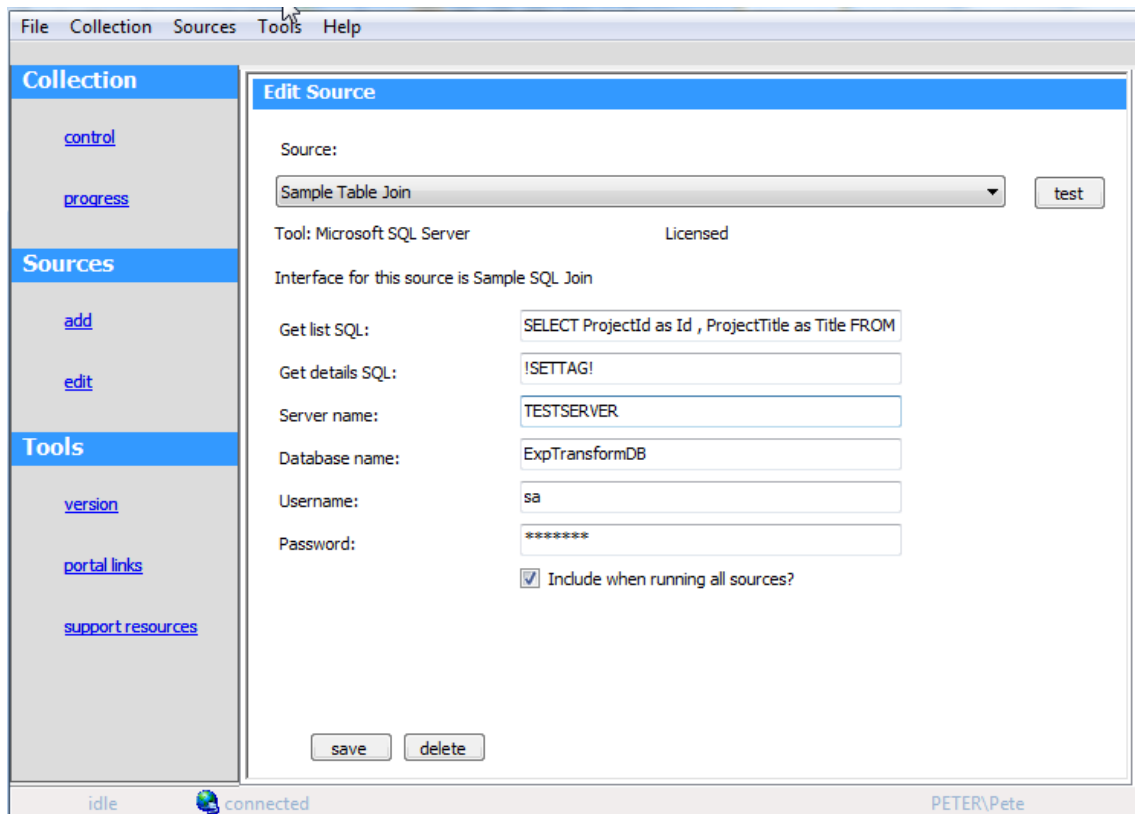
```
SELECT i.IssueTitle, i.ProjectId, p.ProjectTitle
FROM IssueData as i, luProjects as p
WHERE (i.ProjectId = p.ProjectId) AND (i.ProjectID = !ITEM!)
```

In the SQL statement above, notice that the ProjectId field from the IssueData table is joined to the ProjectId field of the luProjects table so that the query can return the ProjectTitle as a field. This type of SQL statement is called a two-table join.

To finish the sample, let’s look at how the Collector is configured.

#### 3.5.6.4. Configuring the Collector to Perform a Join

To configure the collector for a join, you must remove the !FIELDS! tag and use the !SETTAG!. As shown in the image below, the “Get details SQL” textbox is simply filled in with “!SETTAG!”



The “Get list SQL” generally does not change as a result of using either the !FIELDS! or !SETTAG! in the details SQL. For example, in the above source, the Get List SQL is: SELECT ProjectId as Id, ProjectTitle as Title FROM luProjects

### 3.5.6.5. The Resulting Joined Data

The results of the join query using the SETTAG is shown below. The resulting data is stored in a table called “JoinResults”, as specified in the Interface Definition (see previous image).

	g_collectdate	g_collectorid	g_interfaceid	g_itemid	IssueTitle	ProjectId	ProjectName
▶	7/2008 12:00:00 AM	1	21	13151	Warm Restart Generates Cold Start Errors	1	Project 1
	4/2/2008 12:00:...	1	21	13151	Disable 1553B in UPS Mode	1	Project 1
	4/2/2008 12:00:...	1	21	13151	Altitude Assigned to Surface Track	1	Project 1
	4/2/2008 12:00:...	1	21	13151	Weight Calculation Uses Wrong Alloy Coeff	1	Project 1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL



The ProjectId and IssueTitle fields were captured directly from the IssueData table. The ProjectTitle field data was captured from the luProjects table, using a JOIN clause in the SQL statement to obtain the correct value from the lookup table. Queries in this Interface can use any of these fields to filter, sort or select records of interest.

Notice that the JoinResults table contains a number of fields that start with "g\_". These global fields are created automatically for you and are used by IBM Rational Dashboard . The Collector inserts the correct data for these fields as it collects data.

## 4. Collector Reference and Commands

The first subsection describes the menu commands available in the Collector user interface. The second subsection describes the command line commands.

### 4.1. *Menu Command Reference*

#### 4.1.1. *File Menu*

##### 4.1.1.1. **Connect Command**

This command initiates a Collector attempt to connect to the web services. Once connected, the Collector will indicate with the text "connected" in the status bar at the bottom of the main window. If the web service connection cannot be made, the Collector will display a message box containing any reason and will change the status bar text to "disconnected".

You can test the connection to Web Services using the Options command.

##### 4.1.1.2. **Disconnect**

This command disconnects the Collector from any previous web service connection.

##### 4.1.1.3. **Options**

The options command provides a method for configuring the connection to the Portal and web services.

The Portal URL is the server and virtual web name of the Portal (which is an ASP.NET application installed on IIS). The format of the Portal URL is:

`http://<servername>/Portal/`

The Web Services URL is the server and virtual web of the Web Services (which are ASP.NET web services installed on IIS). The format of the web service URL is:

http://<servername>/webservice/

Available options:

- connect automatically to web services when the Collector starts
- use basic credentials other than the current Windows user when connecting
- collect data for after finding a new item during collection

**If you make changes to these options, press Apply to save them before pressing the Test button.**

The "Test" button verifies that using the last saved set of options, the Collector can communicate with web services.

#### **4.1.1.4. Logging**

This command displays the logging information which has been captured in the logfile.txt. You may empty the contents of the log file by pressing the clear button. Also, you may rename or copy the log file to another name using the Windows File Explorer. More information about Collector logging can be found in The Collection Log topic.

The Collector configuration file contains settings which control the operation of logging. The Collector uses Nlog for logging. Consult the Nlog web site for information on how to configure and control. The Nlog configuration is located in the Collector.vshost.exe.config file in the folder where the collector is installed.

#### **4.1.1.5. Exit**

This command exits the Collector. If the Collector is connected to web services, they are disconnected.

## **4.1.2. Collection Menu**

### **4.1.2.1. Control Command**

This command displays a window which allows collection to be started and stopped. When starting collection, you may select all data sources or a single data source to run. In addition, you may specify a date range to collect (selected date to present), which is useful for collecting historic data.

Once collection is started, you may press the stop button to cease collection. The Collector must complete the current operation, so it may not appear to stop immediately. Do not terminate the application or terminate the Collector process (using the Task Manager) until the Collector has stopped processing.

During collection, some commands are disabled. These commands are re-enabled after collection.

### **4.1.2.2. Progress Command**

This command displays the progress of current collection. This window displays current progress, showing the number of interfaces completed as well the number of items found. When collection is complete, totals, including the number of bytes stored in the database, are shown.

You may use the View Log File link at the bottom of the page to view informational messages, warnings or errors encountered during collection.

## **4.1.3. Sources Menu**

### **4.1.3.1. Add Source Command**

This command allows you to add a new data source. For each 'untyped' source, you must select a Portal interface which provides the fields or attributes to validate and collect from the external tool. You must have a connection to web services, otherwise you cannot select a Portal interface for your new data source.

The DOORS data source is a 'typed' source, which requires a type identifier to be specified in the Portal interface and does not require an interface to be selected when

configuring the data source in the Collector.

Finally, the associated interface must be licensed or it will not be shown in the Collector. Once you press the Add button, the edit window is displayed.

#### **4.1.3.2. Edit Source Command**

This command allows you to view or enter data for each data source. Each data source type in the Collector has a unique set of parameters.

Once you have entered data for the source, you must press the save button. If there are any errors in the entered data, a message is displayed in the status bar at the bottom of the main window.

For more information on configuration of individual data sources, please see the Source Details section for the desired data source.

#### **4.1.4. Tools Menu**

##### **4.1.4.1. Version Command**

This window displays the current application and web services version. In addition, the license status (licensed or unlicensed) of each supported data source is displayed.

##### **4.1.4.2. Portal Links Command**

This window provides quick links to pages in the Portal that are regularly used in configuring or performing collection. If the Portal URL has not been configured using the Options command, then the links on this page will not function correctly.

##### **4.1.4.3. Support Resources Command**

This window displays contact information for various support resources. Your sales and support team may also have provided you with contact information for contacting support or see the Getting Help topic.

#### **4.2. Command Line Reference**

The Collector provides a command line interface which starts collection without having to click on shortcuts and menu items. The command line interface allows the Collector to be run from a DOS command window or batch file or scheduled using the Windows Scheduler. For example, the command line can be scheduled to run each evening to collect for the current day.

When run in command line mode, the Collector does not display a user interface or popup information windows on the screen. During command line mode, the Collector writes into a log file (see section on “Collector Logging” for more information). If the Collector encounters a critical error while processing, it will log the error to the log and display the user interface.

Only one instance of the Collector should be run at a time – please exit the Collector user interface before starting the command line.

### 4.2.1. Command Line Parameters

The Collector requires at least one parameter (the “/source” parameter), and allows for two others. When the Collector finds the '/source' parameter on the command line, the Collector runs in command mode. The command line has the following format:

*Collector.exe /source=all; /date=today;*

In the example above, the Collector executable is run with two parameters (/source and /date). Each parameter has a parameter value and is terminated with a “;”.

The parameters supported by the Collector are shown in the following table.

Parameter	Description and Example
/source	specify either “all” or the name of one data source in the collector Examples: /source=all; /source=Excel Cost Data;
/date	specify the collection date or the value “today”. The value “today” indicates that the current date is used as the date of collection. See note below about the format of the date string. Examples: /date=today; /date=11/24/2008; (for mm/dd/yyyy date format) /date=24/11/2008; (for dd/mm/yyyy date format)
/fromdate	specify the starting date of a date range. The date range starts with the value specified in this option and ends with the current date. See note below about the format of the date string. Example: / fromdate =10/20/2008; (for mm/dd/yyyy date format) / fromdate =20/10/2008; (for dd/mm/yyyy date format)

Note on date format:

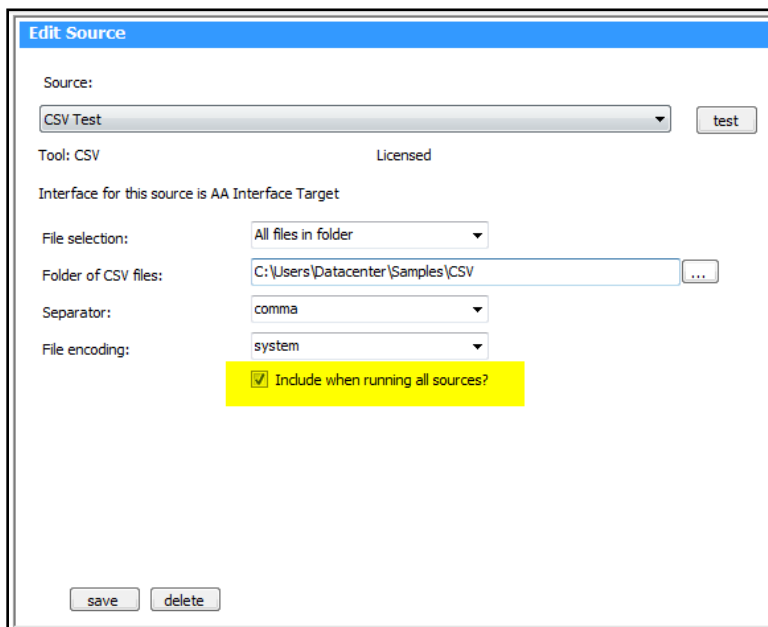
The parameter /date and /fromdate accept a date string as a parameter value. The format of the date string is interpreted using the “current culture” of Windows. For the American/United States culture, the date string will usually be interpreted as “MM/dd/YYYY”. For French and English/United Kingdom cultures, the date string is usually interpreted as “dd/MM/YYYY”.

### 4.2.2. Collecting All Sources for Current Date

To collect all data sources in the Collector using today's date as the collection date, use the following command:

```
Collector.exe /source=all; /date=today;
```

The “/source” parameter specifies that “all” sources should be collected. The Collector will run any data source that was configured with the “Include when running all sources?” option. This option is set or cleared in the Edit Source window of the collector, as shown in the following figure.



The “/date” parameter specifies that the current system date (indicated by “today”) should be used as the collection date. For more information on collection dates and schedules, please see the

### 4.2.3. Collecting One Source for a Past Date

The Collector allows you to collect for one source at a time using the “/source” parameter. In addition, you may specify a collection date on the command line, rather than using the current system, date as the collection date. An example of collecting one source for a past collection date is shown below:

```
Collector.exe /source=CSV Test; /date=30/1/2006;
```



In the example above, notice that the “/source” parameter specifies the name of a data source that has been specified in the Collector.

The “/date” parameter specifies a date in “dd/mm/yyyy” format that is used by the collector as the collection date. Essentially, the Collector queries the specified data source such as “What data do you have for the date of 30/1/2006?” Notice that some data sources are not sensitive to the collect date. This means that if you vary the collection date value, you receive the same information.

#### **4.2.4. Collecting One Source for Multiple Dates in the Past**

The Collector allows you to collect for one source at a time using the “/source” parameter. In addition, you may specify a collection period, rather than one date period, using the “/fromdate” parameter. An example of collecting one source for multiple dates in the past is shown below:

```
Collector.exe /source=CSV Test; /fromdate=1/1/2007;
```

In the example above, notice that the “/source” parameter specifies the name of a data source; the “/fromdate” specifies that collection should occur starting from 1 January 2006 and continuing to today’s date.

When “/fromdate” is specified on the command line, the Collector will determine which dates are valid collection dates, and then request data from the data source for each collection date. The Collector is provided the collection dates through the Web Services, which returns a list of assigned items that have been collected by the data source, along with the associated schedule.

For example, is an item called “Defect.csv” has been collected using the source “CSV Test” and assigned to a unit with a monthly schedule, then the “/fromdate” will result in every monthly collection date from 1 January 2007 to the present system date being collected.

### **4.3. Technical References**

This section describes various formats and syntax of capabilities used within the Collector.

### **4.3.1. Collector List File Format**

A list file provides a method for selecting files to be accessed during a collection. A list file is used when you do not want to use every file in a folder. A list file must be placed in a folder accessible to the Collector, and contains a list of files which themselves can be located in another folder.

The Collector opens the list file and then reads each line to determine which file to open. Lines that start with a semi-colon are treated as comments and are ignored. Non-comment lines should be comma separated with two values. Possible line formats are shown below:

; comment

file id, file\_name

'file id', 'file name'

file id, 'file name'

'file id', file name

You may use either a single or double quote to surround the value as long as you use the same one to end the value as you do to start.

The file name can be either:

- a file name (e.g. "demo.csv")
- a folder and file name (e.g. "c:\Data\Projects\Alpha\Defects.csv").

In the first case (i.e. only a file name), then the current folder (where the list file is located) is used.

A sample list file is shown below:

; three CSV files

DefectsMajor, Defects\_Major.csv

DefectsMinor1 , C:\Datacenter\Project1\Defects\_Minor\_1.csv

DefectsMinor2 , C:\Datacenter\Project2\RevA\Defects\_Minor\_2.csv

### **4.3.2. Collector Connection List Format**

A connection list file provides a method for specifying the name of multiple SQL databases to the Collector. Similar to the previous list file, the connection list allows a

data source to access multiple databases, treating each database as a collected item.

The connection list file is a comma separated value file, where each line is either a comment or a line with two values, as follows:

; comment

connection id, connect string

'connection id', 'connect string'

connection id, 'connect string'

'connection id', connect string

You may use either a single or double quote to surround the value as long as you use the same one to end the value as you do to start.

The "connection id" is a text string up to 50 characters that acts as a title.

The "connect string" is a text string consisting of the database type plus the connection syntax. The database type must be one of the following:

- "mssql" – for Microsoft Server 2005 or 2008 database
- "oracle" – for Oracle Database server
- "odbc" – for an ODBC data source

The connection syntax must be valid for the database type.

A sample connection list file is shown below:

; list file with three entries

"Project A", "mssql;server=MSSQL;uid=sa;pwd=go;Database=ProjA\_DB;"

"Project B", "odbc;dsn=ProjB\_ActionSQLDB;"

"Project C", "oracle; Data Source=Oracle8i;Integrated Security=yes"

For an odbc database type, the following are valid values:

"odbc;Driver={SQL

Server};Server=(local);Trusted\_Connection=Yes;Database=AdventureWorks;"

"odbc;Driver={Microsoft Access Driver (\*.mdb)};DBQ=c:\bin\Northwind.mdb"

"odbc;DSN=dsnname"

For additional information on connection string syntax, please consult Microsoft .NET documentation (search for the term "ConnectionString") in the following topics:

- System.Data.SqlClient
- System.Data.Odbc

- System.Data.OracleClient

### **4.3.3. Replacement Tags in GetDetails Query**

To simplify the flexibility of the Collector, the Collector provides a set of replacement tags. When the replacement tag is encountered in a getlist or getdetail query, the replacement tag is removed from the query and replaced with the appropriate value.

Below is a list of the replacement tags.

#### **!SETTAG!**

During collection, the collector replaces the reserved word !SETTAG! with the text in each interface field set's Parameter Replacement Tag field. The field set Parameter Replacement Tag is located on the 'field' subtab of the Edit Interface screen. Each field set has a unique field set tag whose value is entered into the portal and then provided to the collector, which replaces this tag with the value from the Portal field set.

For Excel data sources, this tag is configured in the portal with the worksheet name, cell range and processing flags that are provided to the Collector at run time. When an excel interface in the Portal is configured with these parameters, they are passed to the Collector at run time. If you do not want to use configuration data from the Portal for an Excel collection, you can disable (uncheck) the option to "Use configuration data from Portal" when configuring the data source in the collector.

#### **!ITEM!**

As the collector processes each item found by the GetList query, the !ITEM! reserved word is replaced in the GetDetails query by the unique item identifier of the item. The item tag is usually an integer or string value that is unique among all the items collected from a single data source. This tag allows the collector to sub-select data from a data source so that only the data for a specific item is returned.

#### **!RELEASE!**

This tag is the same as the !ITEM! tag (above). This tag is provided only for backwards compatibility with IBM Rational Synergy and IBM Rational Change sources created in version 3.0.

**It is recommended that the " !ITEM! " tag be used instead.**

#### **!PROJECT!**

This tag is the same as the !ITEM! tag (above). This tag is provided only for backwards

compatibility with Ms Project Server 2003 data sources created in version 3.0.  
**It is recommended that the “!ITEM!” tag be used instead.**

#### !FIELDS!

The !FIELDS! reserved word is replaced by a comma-separated list of the fields in the current interface field set. For example, if the interface field set contains fields for title, status, priority, the date found, and the responsible party, the !FIELDS! replacement string might look like this: 'Title,Status,Priority,DateFound,ResponPerson'. The fields are the database field names from the interface field set.

#### !THEFIELDS!

This tag is provided only for backwards compatibility with IBM Rational Synergy and IBM Rational Change sources created in version 3.0.  
**It is recommended that the “!FIELDS!” tag be used instead.**

#### !SCHEMA!

The !SCHEMA! tag is used for Oracle collections where the data is stored in multiple schemas. This allows the user to collect using the same interface from multiple projects at once from databases, such as QualityCenter, where each project has a different Schema name, but the same table names and database structure. The schema value is extracted from the user interface or from a connection list file during collection.

#### !COLLECTDATE!

This tag is replaced with current collection date, in ISO SQL standard format. For example, if the Collector was run on 20 January 2003, the !COLLECTDATE! tag would be replaced with the text '20030120'.

#### **4.3.4. Date-Stamped File Names**

Several of the data sources provide an option for selecting what is called “Date stamped files”. Date-stamped files allow you to keep historic versions of your files in the same folder by using a naming convention which includes a date. The following list of files represents a single set of data for multiple periods (from 1 January 2009 to 1 April 2009):

- Projecta\_defects\_20090101.xml
- Projecta\_defects\_20090201.xml
- Projecta\_defects\_20090301.xml
- Projecta\_defects\_20090401.xml

Date stamped files are file names which contain a base name substring and a date substring, where the date substring has the format YYYYMMDD. You may keep all historic and current files in the same folder however you must name them using this format.

When the Collector scans a folder with date-stamped files, it first removes the date stamp from the file name to obtain the item name. For example, a file called “project\_risks20080601.csv” contains an item name of “projects\_risks.csv”. The Collector will scan the folder and return all unique items to the Portal as items to be assigned and then collected.

During collection, the Collector compares the collection date specified by the user to the date part of the file name. In the previous example of “project\_risks20080601.csv”, the date stamp is “20080601” for 1 June 2008. If the Collector was run for the collection date of 1 June 2008, then this date-stamped file would be collected.

## 5. Collector Operations

This section describes operations performed using the collector.

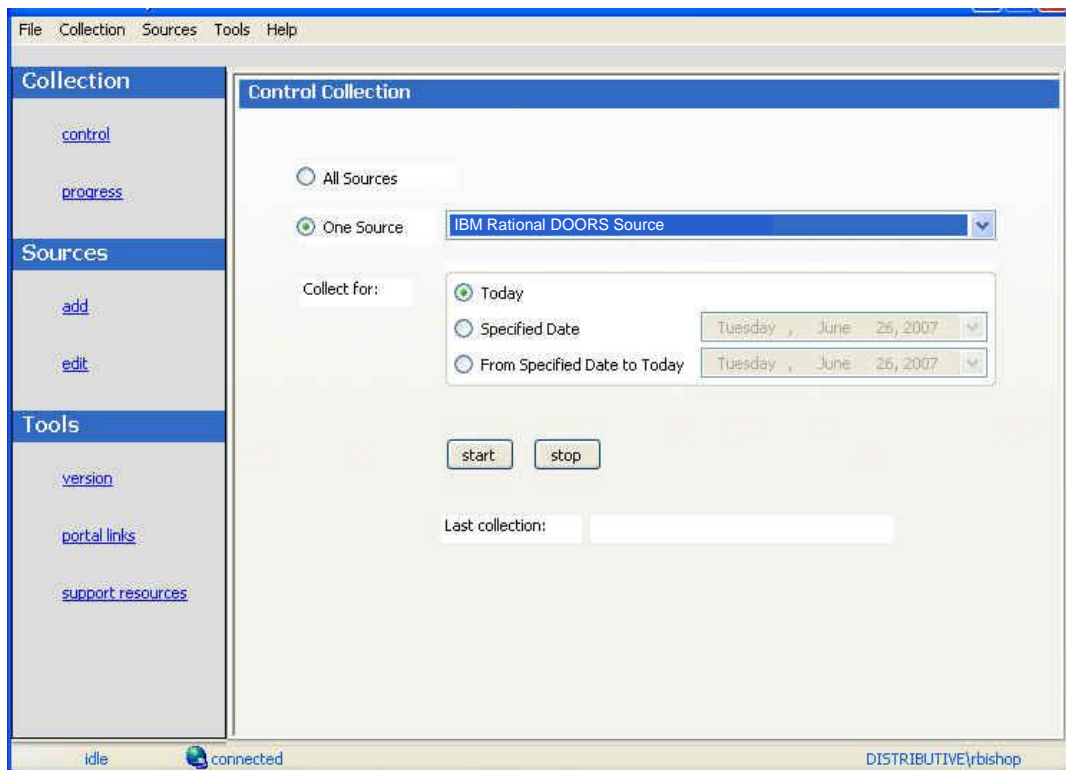
### 5.1. Configuring the Web Services

This subsection describes how to check and configure the connection between the Collector and web services.

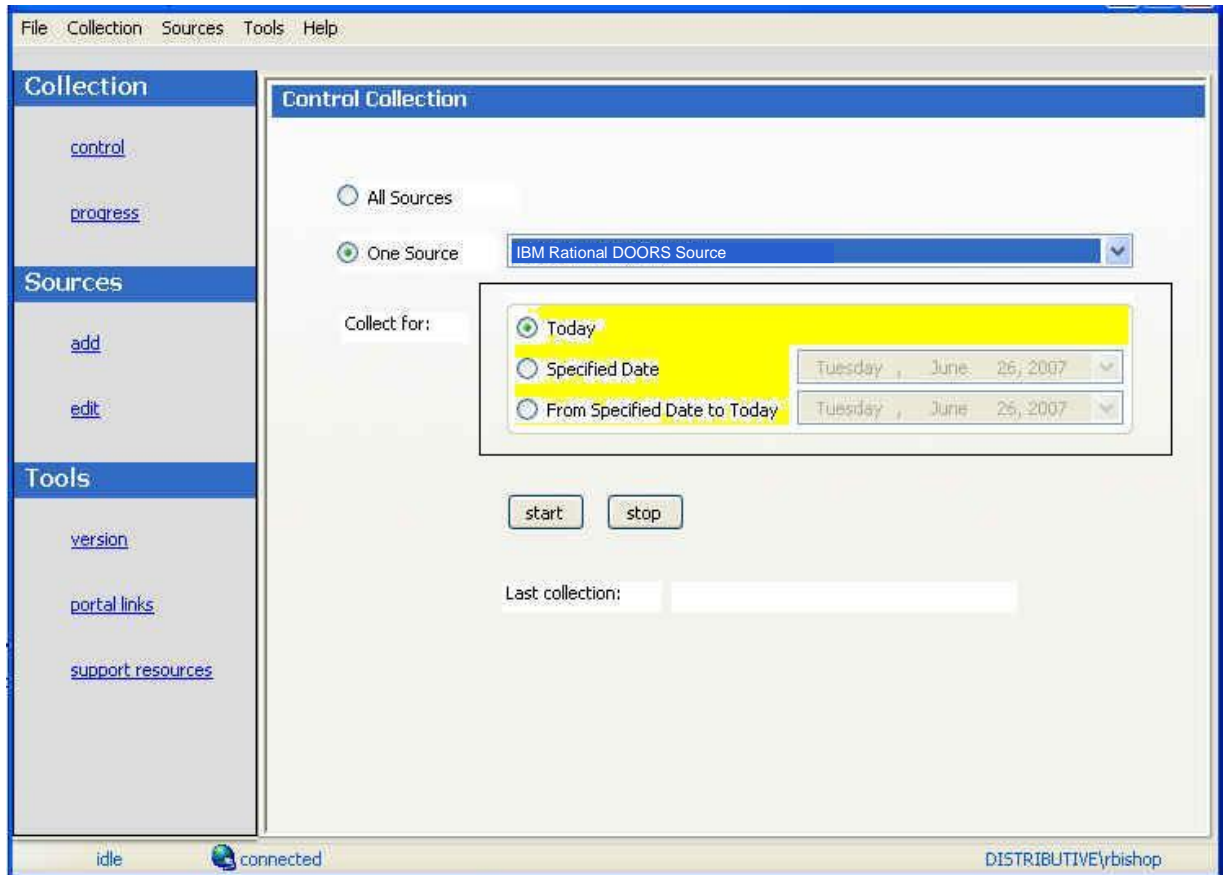
### 5.2. Running a Collection

This subsection describes the steps for collecting data from one external application, in this case IBM Rational Doors. Please note that there are detailed procedures for other interfaces available in separate Walkthrough documents.

#### 1. Open the Collector



2. Select collect "One Source"
3. Select the desired source from the drop down menu beside 'One Source'.
4. Select the date or dates you want to collect. The default is to collect for today's date.



You may also select the button for 'Specified Date' and choose a date from the calendar next to it. This will collect for the date that you specify.

You may also select the button for 'From Specified Date to Today' and choose a date from the calendar next to that button. This will collect for valid collection dates between the date you specify and today's date. Note that the date you select here must be before today's date.

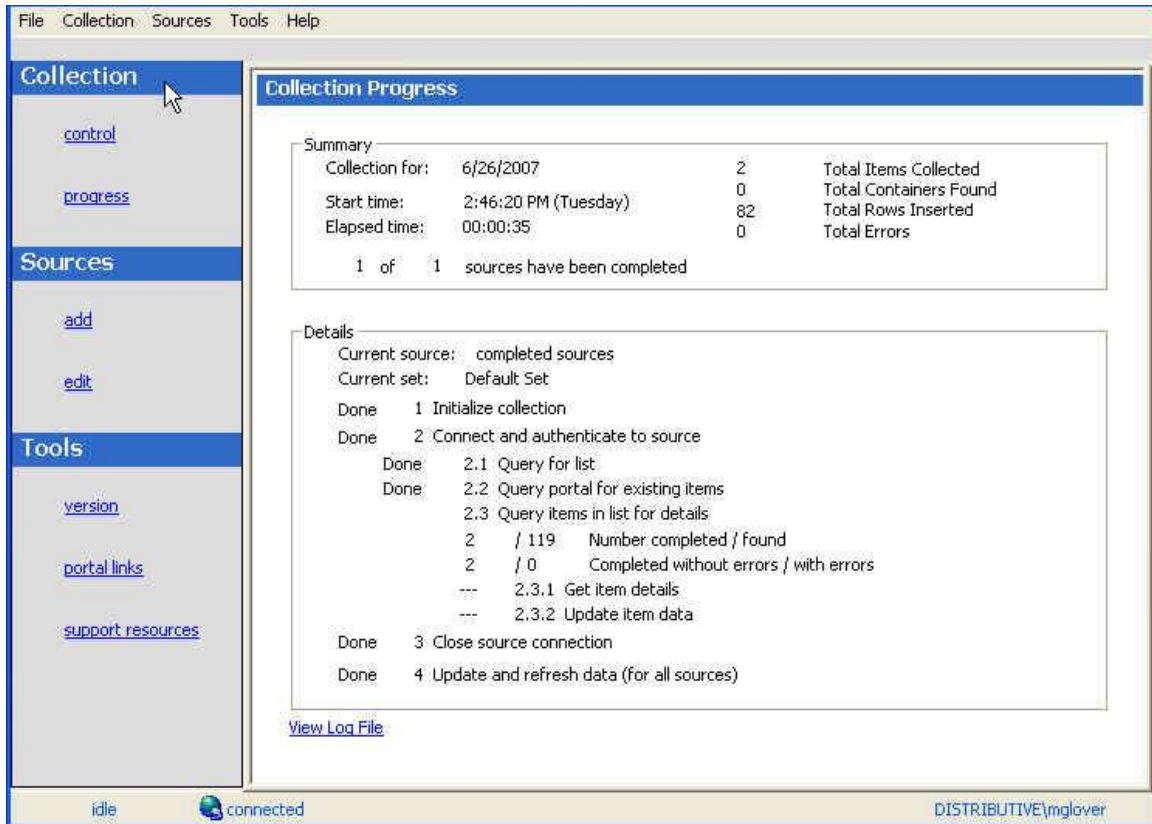
The 'From Specified Date to Today' option is designed to allow backfill of existing data to existing, assigned items. This option will run collection only for valid schedule collection dates in schedules that are in use by assigned items. This option is not intended for use in new systems, as no collection dates will be found if no items are currently assigned.



When collecting for the first time, use either the 'Today' or the 'Specified Date' option.

6. Click the **Start** button

The *Progress* page shows the progress of the current collection. There are two sections of the *Progress* page, the Summary section and the Details section.



The Summary Section has information about the full collection. It has the start and end times, the total number of items collected, the total number of lines inserted to the database and the total number of errors that occurred during the collection. If it is an "All Sources" run, it also shows information on how many sources have been completed. Aside from the start time, this information is only updated when the collection is completed.

The Details Section has information about both the full collection of the current source and information for the individual items that are being collected. This section is updated in synchronization with the collection process.

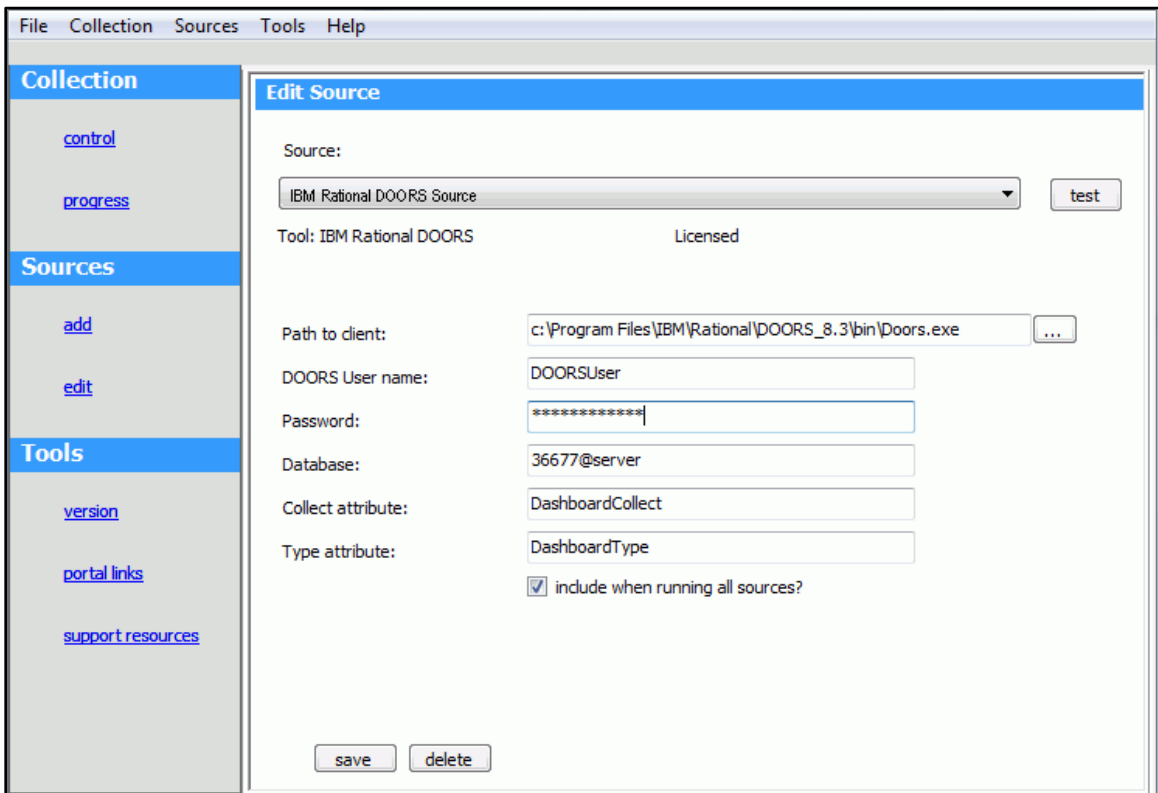
If additional information is needed, specifically about errors and collections, check the Collector Log file by pressing the link “View Log File” at the bottom of the page for more information.

### 5.3. Testing a Data Source

Within the Collector, there is an option for the user to test the items from which the Collector will be retrieving data without completely running the collection. This test button returns results that show which items will be collected from the source, and, if there are items that are not going to be collected by the Collector, it returns the reason why the items won't be collected. This is useful when sources are just being set up for collection, and the user wants to check configurations to assure that the data they're hoping to collect will be collected. It is also useful to determine whether items have been assigned and if their collect dates are correctly set up.

Using the Test Button, it is possible to see what the Collector would collect from a source before running the Collection. The steps to run the test are as follows.

1. Open the **Collector**.
2. Click on the **Edit** button to see the sources.



3. Once the source has been configured, click the **Test** button to test the collection.
4. Review the **results** to review which items would be collected when the collection is run.

Test Collection Details

Source: IBM Rational DOORS Source  
Collection for: 3:32:12 PM

ItemName	Assigned	Collect	New?	Notes
/Big Un to Test	no	no	no	container
/Big Un to Test/Prj3114118...	no	no	no	container
SRD0	yes	yes	no	
URD0	no	no	no	
Empty0	no	no	no	
TestModule0	no	no	no	
SRD1	no	no	no	
URD1	no	no	no	
Empty1	yes	no	no	
TestModule1	no	no	no	
SRD10	no	no	no	
URD10	no	no	no	

OK

There are 5 columns:

- item name reported by the data source
- whether the item is assigned to a unit in the Portal
- whether the item will be collected
- whether the item appears to be new item (i.e. not collected previously)
- notes indicate whether the item is a basic item or a container of items

There are several points about how the Collector handles items:

- If an item is new, the Collector will report the item to the Portal so that it can be assigned. See note below about data for new items.
- If an item is not new and not assigned to a unit, the Collector will not collect data for the item.
- If an item is not new and assigned, but the date for which the collection is run is not a collect date for the item, it will not be collected.
- If the item is not new and assigned and today is a/the collection date of the item (based on the schedule used within the unit where the item is assigned), then

the Collector will collect data for the item

#### Notes

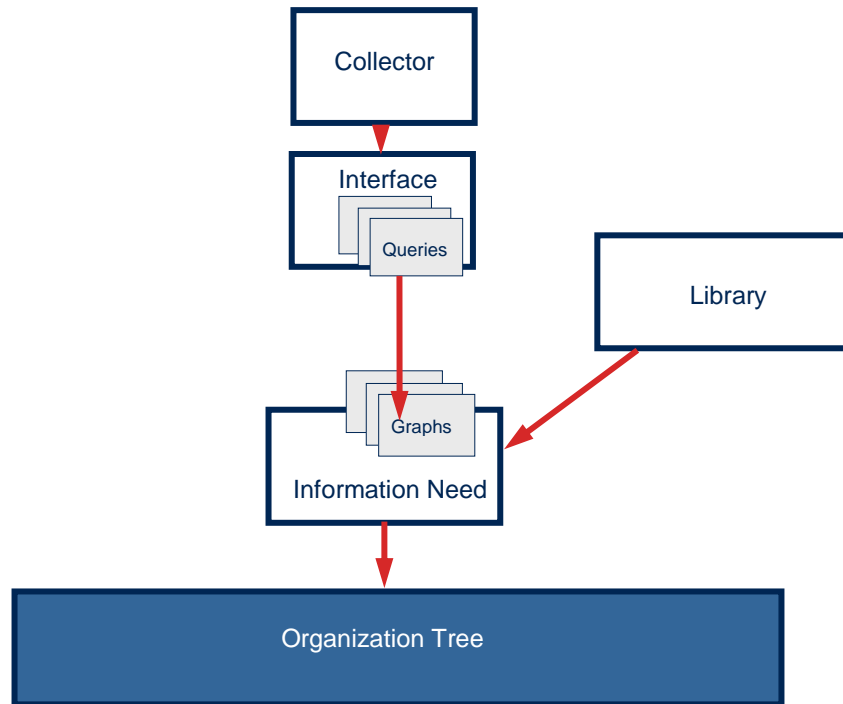
In the File -> Options form, there is an option for "Collect data for new items?". This option is normally disabled, so that when the Collector finds a new item from a data source, it does not collect data for it. When testing in development or test environments, you may want the Collector to collect data for new items so that you can immediately start developing and validating queries and equations. This option though is not generally useful in production environment as it results in data being collected the first time the item is found even though it is not a collection date which could make query values wrong because data is not actually on the collection dates.

### ***5.4. Populating a Basic Unit from The Library***

The Collector gathers data from an external tool. Once collected, the data is organized together with Information Needs and analyzed by managers in the Organization Tree.

Below is a high-level textual description of the flow:

- The Collector gathers data from outside sources and stores it in the Portal Database.
- Interfaces allow users to define the data coming in from the Collector and associated Queries
- An Interface query is assigned to a series in a Library graph to link the data obtained by the Collector and the series used in Graphs
- Information Needs are a set of Graphs, series, status and measurement guidance that can be associated with a Unit. A Unit represents a real world item (e.g. a project) and a Unit can be associated with multiple Information Needs
- The Organization Tree allows users to display and analyze data for different Units arranged hierarchically



**Figure 8**

The first time that a collection is run, the collector will retrieve all items that could possibly have data collected. These items are considered 'new'. In order to collect data, for new or existing items, the collector will check whether the item has been assigned to a unit and whether the day of collection is one of the scheduled collection dates.

If a new item is added to the tool (such as a new module or new release) after the first collection has been run, data will be collected for the new item.

### ***5.5. Configuring and Editing the Organizational Tree***

The Organization Tree defines the structure of Units. It can be edited by clicking the Edit sub-tab on top of the organization tree.

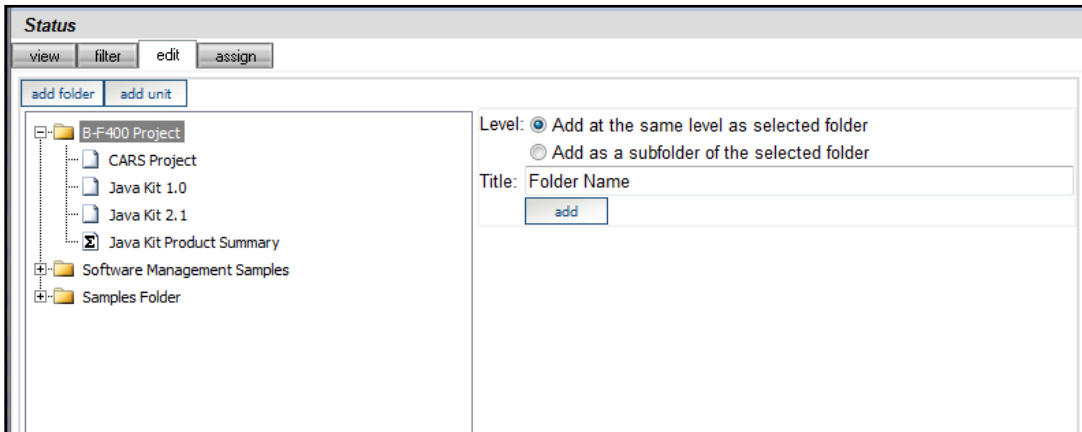


Figure 9

1. Select the folder in which you want to create a new folder. In this example, the B-F400 Project Folder.
2. Click on the add folder button on the top of the tree
3. On the right hand side of the page, select the level you want to create the folder at:
  - Same level as selected folder – this will place the folder on the same level as the folder selected in the organizational tree.
  - Subfolder of the selected folder – this will place the folder on level below the selected folder in the organizational tree.
4. Enter the Title. Enter a title name for the new folder (e.g. JIM Project)
5. Click add. This will create a new folder in the designated location

Once the folder has been added, you may (optionally) add descriptive and other information to it.

With the Edit sub-tab still showing, the folder created in the last step is displayed.

1. Enter Description. Enter a description that describes the folder (optional)
2. Enter Reference URL. Enter an URL that is tied or related to the folder (optional)
3. Click Save.

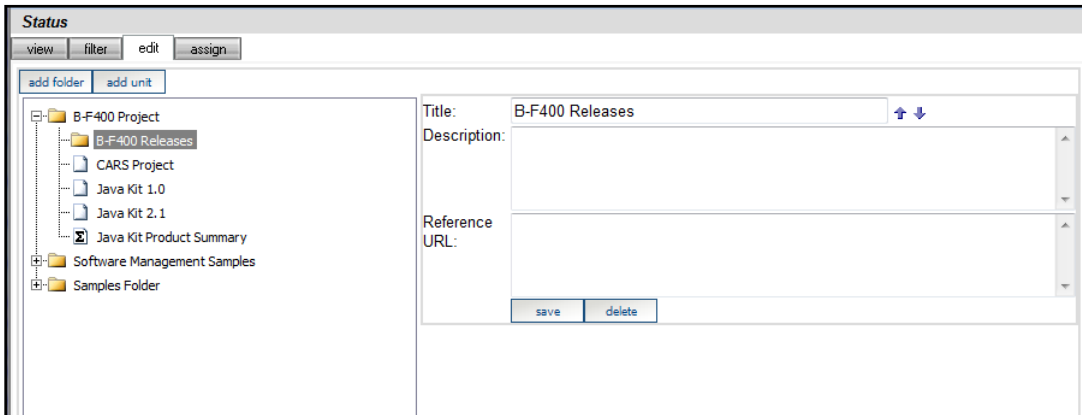


Figure 10

## 5.6. Configuring Data Sources

This section describes the options and parameters for each type of data source.

Note that you may be able to define or update a data source even though you do not have a license to run/perform collection.

### 5.6.1. Pre-Requisites for Data Sources

The table below describes the software or applications which must be required (if any) for each data source. Because the Collector itself is a .NET Windows application, the Microsoft .NET Framework is already installed.

Data Source	Connect Method	Required Components
CSV	file access	none (uses .NET)
HP Quality Center	database	for Oracle databases, the Oracle client
IBM Rational ClearQuest	database	for Oracle databases, the Oracle client
Microsoft Access	file access	none (uses .NET)
Microsoft Excel	OLE	Microsoft Excel 2003 or 2007
Microsoft Project Desktop 2003	OLE	Microsoft Project 2003 or 2007
Microsoft Project Server 2003	database	none (uses .NET)
Microsoft Project Server 2007	database	none (uses .NET)

Multi-Source SQL	database	for Oracle database, the Oracle client
ODBC	database	for Oracle databases, the Oracle client
Oracle	database	the Oracle client
IBM Rational Change	command line	IBM Rational Change client for Windows
IBM Rational Doors	command line	IBM Rational Doors client for Windows
IBM Rational Synergy	command line	IBM Rational Synergy client for Windows
XML ADO.NET	file access	none (uses .NET)

Before running a collection, you should verify that the applicable software or component is installed and functioning correctly. For example, before collecting data from IBM Rational Synergy, you should make sure that the Synergy client starts and connects to the appropriate server.

### **5.6.2. *Configuring a CSV Source***

The CSV interface collects data from text files which have values separated by a comma, tab, semicolon or colon. The CSV file must contain one logical record on each line of the file – a record is not allowed to span more than one line.

The first line in each file must be a header which contains the names of the fields (or columns) in the file. The field names in the first line are matched with the field names defined in the Portal interface. All fields in the CSV file do not have to be present in the Portal interface. However, every field defined in the Portal interface (which is set to be collected) must be present in the CSV file.

All lines in the file must have the same number of fields. Each field is an unquoted value, a single-quoted value or a double-quoted value. If a text or string value contains the separator character, then the value must be quoted.



This source requires the following parameters:

Parameter	Description
File selection	One of: <ul style="list-style-type: none"><li>• all files in folder</li><li>• a listing file</li><li>• data-stamped files in folder</li></ul>
Folder name	Name of the folder containing files OR the folder and file name of the list file. (See Notes below)
Separator	Delimiter character between fields, one of: <ul style="list-style-type: none"><li>• comma</li><li>• semi-colon</li><li>• colon</li><li>• hyphen</li><li>• tab</li></ul>
File encoding	Encoding of the file, one of: <ul style="list-style-type: none"><li>• System (use the current system default encoding)</li><li>• UTF-7</li><li>• UTF-8</li><li>• Unicode</li></ul>

Notes:

The list file capability was modified in version 3.6.1 to allow either a folder name or a folder and file name. For example, if you select just a folder for the list file parameter, then the list file is called “(selected folder)\list.txt”. Alternatively, once you select a folder name using the UI, you may type in the text “\myfiles.txt” to indicate the name of the list file is called “(selected folder)\myfiles.txt”.

Date stamped files are file names which contain a base name substring and a date substring, where the date substring has the format YYYYMMDD. When you specify date-stamped files, you may keep all historic and current files in the same folder however you must name them using this format. An example of a date-stamped file name is “defects\_projectA\_20080601.csv”. The collection date specified by the user is compared to the date substring (in this case “20080601”) to determine if the data

should be collected. So, if the Collection were run for the collection date 1 June 2008, then the sample file would be collected.

### **5.6.3. Configuring an HP Quality Center Source**

The HP Quality Center interface collects data from database used by Quality Center to store test and other project data. Because Quality Center uses a different database for each project, you must create and maintain a connection file, which identifies each database and provides connection information for each one.

This source requires the following parameters:

<b>Parameters</b>	<b>Description</b>
Connection list	The name of a file containing a list of connection strings See Notes below.
Get Details SQL	A SQL statement to get detail records for a single item from each Test Director database.  The SQL statement must contain a "!FIELDS!" tag that is replaced with a comma separated list of field names.  The SQL may contain a "!ID!" tag that is replaced by the item id.

Notes:

The format of a connection list is defined in the section titled "Connection List File Format".

### **5.6.4. Configuring an IBM Rational ClearQuest Source**

This interface provides data collection from one or more ClearQuest (CQ) database servers. Because CQ is a highly configurable tool, you should define the CQ interface in the Portal to support your CQ deployment. This data source uses one of Microsoft, Oracle or ODBC to connect to the CQ database. For Oracle, you must have the Oracle client installed.

This source requires the following parameters.

Parameter	Description
Method for list	How to obtain the list of items to collect data for. One of: <ul style="list-style-type: none"> <li>• SQL statement</li> <li>• Connection list file</li> </ul>
Database Type	One of: <ul style="list-style-type: none"> <li>• Microsoft</li> <li>• Oracle</li> <li>• ODBC</li> </ul>
Server/Data Source	The computer name where the SQL server is running or the Oracle data source name (when list method is SQL)
Database	The name of the database
Username	SQL named user
Password	SQL Password
Get List SQL	SQL statement to get a list of managed items The SQL statement must return a field called "id" and one called "title".
Folder containing list file	The folder where the list file is located (used when method for list is list file) See Notes below.
Get Details SQL	SQL statement to get detail records for a single item The SQL statement must contain a "!FIELDS!" tag that is replaced with a comma separated list of field names. The SQL may contain a "!ID!" tag that is replaced by the item id.

Notes:

The format of a connection list is defined in the section titled "Connection List File Format".

### 5.6.5. *Configuring a Microsoft Access Source*

The Microsoft Access Database interface collects data from Access files created using Office 2000, Office XP or Office 2003. This interface uses the .NET ADO.NET interface for opening an MDB file. Microsoft Access is not required in order to use this interface.

This source requires the following parameters:

Parameter	Description
Files to scan	One of: <ul style="list-style-type: none"><li>• a single MDB file</li><li>• all files in folder</li><li>• a listing file</li><li>• date-stamped files in folder</li></ul> If you select "selected file", then you must specify the "Folder and file name" parameter and the "Folder name" parameter is disabled. If you select "all files in folder" or "use a listing file", then you must specify the "Folder name" parameter.
Folder and name of Access file	name of the folder and MDB file
Folder name	name of the folder containing MDB files
Get List SQL	A SQL statement to get a list of managed items from the MDB file. The SQL statement must return a field called "id" and one called "title". Used only if file selection is "a single MDB file"
Get Details SQL	A SQL statement to get detail records for a single item from the MDB file. The SQL statement must contain a "!FIELDS!" tag that is replaced with a comma separated list of field names. The SQL may contain a "!ID!" tag that is replaced by the item id.

## Notes:

The list file capability was modified in version 3.6.1 to allow either a folder name or a folder and file name. For example, if you select just a folder for the list file parameter, then the list file is called “(selected folder)\list.txt”. Alternatively, once you select a folder name using the UI, you may type in the text “\myfiles.txt” to indicate the name of the list file is called “(selected folder)\myfiles.txt”.

Date stamped files are file names which contain a base name substring and a date substring, where the date substring has the format YYYYMMDD. When you specify date-stamped files, you may keep all historic and current files in the same folder however you must name them using this format. An example of a date-stamped file name is “defects\_projectA\_20080601.csv”. The collection date specified by the user is compared to the date substring (in this case “20080601”) to determine if the data should be collected.

The MDB cannot require login security or user name/password to open.

Currently, Access 2007 files (ie “MDBX” files) cannot be opened.

### **5.6.6. *Configuring a Microsoft Excel Source***

The Microsoft Excel interface extracts data from an Excel file. This interface supports Office 2000, XP, 2003 and 2007. This interface requires Excel 2003 or 2007 to be installed on the same machine as the collector. The interface opens each selected Excel file, and reads a cell range from each specified worksheet.

In the Collector, you may only specify one worksheet and cell range to collect from. However, if you enable the option (available in data sources created with 3.6.1 or later) to “Use Portal values?”, then you may collect data from multiple worksheets in the same file. When editing an Excel interface in the Portal, each field set defines a (potentially) different worksheet to collect data.

This source requires the following parameters:

Parameter	Description
File selection	One of: <ul style="list-style-type: none"> <li>• all files in folder</li> <li>• a listing file</li> <li>• data-stamped files in folder</li> </ul>
Folder name	name of the folder containing Excel files
Worksheet name	name of the worksheet where the data resides
Orientation of data	Eight horizontal or vertical Horizontal orientation means that data is read from left to right, with each column being a data record, and the fields (in the Portal interface) are row numbers. Vertical orientation means that data is read from top to down, with each row being a data record, and the fields (in the Portal interface) are column letters.
Cell range	A cell range like "A2:M100" which specifies the data cells to be collected. The cell range should be configured as small as is practically possible.
Stop On Blank	Collector will stop if all collected data values are blank or null for a row or column.
Use Portal values	Indicates that the instructions for a worksheet, orientation, cell range, and stop-on-blank will be provided by the Portal.

Notes:

The Collector ignores records where all the fields contain a blank or null value. So, you may use the cell range to define a larger cell range than your current data set.

The “Stop on Blank” option (entered in the Collector or Portal user interface) will cause the Collector to stop collecting data as soon as a row or column is encountered where all collected cells are blank. This means that a row or column could have data in cells

that are not being collected but since this data is not a collected cell, the Collector does not consider it.

### **5.6.7. Configuring a Microsoft Project Desktop 2003 Source**

This data source extracts task and resource data from Microsoft Office 2003 Project desktop. The interface uses the Microsoft Office Project 2003 ActiveX/COM object to open the project (i.e. MPP) files. This requires that Microsoft Project 2003 be installed on the same computer as the Collector. This source can collect task only information from Project files.

This source requires the following parameters.

<b>Parameter</b>	<b>Description</b>
File selection	One of: <ul style="list-style-type: none"><li>• all files in folder</li><li>• a listing file</li><li>• data-stamped files in folder</li></ul> See Notes below.
Folder name	Name of the folder containing files OR the folder and file name of the list file. See Notes below.

Notes:

The list file capability was modified in version 3.6.1 to allow either a folder name or a folder and file name. For example, if you select just a folder for the list file parameter, then the list file is called “(selected folder)\list.txt”. Alternatively, once you select a folder name using the UI, you may type in the text “\myfiles.txt” to indicate the name of the list file is called “(selected folder)\myfiles.txt”.

Date stamped files are file names which contain a base name substring and a date substring, where the date substring has the format YYYYMMDD. When you specify date-stamped files, you may keep all historic and current files in the same folder however you must name them using this format. An example of a date-stamped file name is “defects\_projectA\_20080601.csv”. The collection date specified by the user is compared to the date substring (in this case “20080601”) to determine if the data should be collected.

### 5.6.8. *Configuring a Microsoft Project Server 2003 Source*

This data source extracts task and resource data from Microsoft Project Server 2003. The interface uses a Microsoft SQL database connection to the Project Server 2003 database. The Project Server 2003 client tools, COM or API objects are required.

This source can collect different types of information from Project Server, for example tasks and resources. This is done by configuring the interface in the Portal with field sets that define which tables within the Project Server database to use.

This source requires the following parameters.

Parameter	Description
Get projects SQL	SQL statement to get a list of items. The SQL statement must return a field called "id" and one called "title".
Get details SQL	SQL statement to get detail records for a single item. The SQL statement must contain a "!FIELDS!" tag that is replaced with a comma separated list of field names.
Server	The computer name where the SQL Server is running
Database name	The name of the database
Username	SQL named user name
Password	Password for the named user

Notes:

The user name and password is for a SQL database, not a user name and password for a Project Server account. If needed, contact your database or Project Server administrator to obtain a user name and password the Project Server database. Unless the Collector is installed on the same machine as the SQL database, the user name must be a named SQL account.



### **5.6.9. Configuring a Microsoft Project Server 2007 Source**

Not implemented in 3.6.1.

### **5.6.10. Configuring a Multi-Source SQL Source**

The Multi-source data source collects data from multiple databases, where the list of database is contained in a special text file called a “connection list”. Each line in the connection list file contains the connection information for three different types of databases: Microsoft, Oracle and ODBC. The connection list may contain connection information for any type or number of databases.

This source requires the following parameters.

<b>Parameter</b>	<b>Description</b>
Connection List	The folder and file name of a file which contains a list of connection strings.
Get Details SQL	SQL statement which returns records for a single item. This SQL statement is run for each entry in the list file.  The SQL statement must contain a "!"FIELDS!" tag that is replaced with a comma separated list of field names.  The SQL may contain a "!"ITEMID!" tag that is replaced by the item id.

#### Notes

The format of a connection list is defined in the section titled “Connection List File Format”.

### 5.6.11. *Configuring an ODBC Database Source*

The ODBC data source provides data collection from any data source name (DSN) defined in the ODBC control panel. ODBC supports a wide variety of databases and file formats. Windows provides a control panel application to configure each ODBC DSN.

This data source requires the following parameters.

Parameter	Description
DSN Name	Data source name (DSN) defined in the ODBC control panel. A DSN is a system, user or file source.
Get List SQL	SQL statement that returns a list of items. The SQL statement must return a field called "id" and one called "title".
Get Details SQL	SQL statement which returns records for a single item. The SQL statement must contain a "!FIELDS!" tag that is replaced with a comma separated list of field names. The SQL may contain a "!ITEMID!" tag that is replaced by the item id.

To verify the DSN, you should use the test or verify function to make sure that the DSN is configured correctly. You should verify that your system has the correct pre-requisites installed.

### 5.6.12. *Configuring an Oracle SQL Database Source*

This data source provides data collection from an Oracle 9 or 10 SQL server. This data source uses the .NET 2.0 data client library to access the native Oracle Client library. You must have the Oracle client installed on the same machine as the Collector to use this data source.

This source requires the following parameters:

Parameter	Description
Get List SQL	A SQL statement to get a list of managed items. The SQL statement must return a field called "id" and one called "title".
Get Details SQL	A SQL statement to get detail records for a single item. The SQL statement must contain a "!FIELDS!" tag that is replaced with a comma separated list of field names. The SQL may contain a "!ID!" tag that is replaced by the item id.
TNS Name	An Oracle TNS name defined on the host computer
Username	Oracle user name
Password	Oracle password

### 5.6.13. *Configuring a IBM Rational Change Source*

This source requires an IBM Rational Change Windows client be installed and configured before attempting collection. To set up and run an IBM Rational Change collection, IBM Rational Synergy must be enabled to run the "ccm" Command Line Interface (CLI). Please refer to Synergy help files to enable the CLI.

This source requires the following parameters.

Parameter	Description
CCM query client path	The location of the ccm.exe file on the current computer. Click the <b>button</b> to the side to browse for the file on the computer. The default value in the collector is the default value for IBM Rational Synergy 6.4 installations.
Database path	The location of the database that is being collected.
User name	A user name for a database administrator who would have

	access to all of the releases and CRs that need to be collected.
Password	Password for the user supplied
Get List Query	CLI command to retrieve list of items
Get Details Query	CLI command to get details for each item returned by the Get List Query

Notes:

The "Get List Query" and "Get Details Query" are CLI commands to retrieve items and item details. The default values select all of the releases in the database, and then collect all problems (defects or enhancements) from each release.

The Get List query can be written to return a list of objects found in the IBM Rational Change database. The Get Details query is run for each object found by the Get List query. When the Get Details query executes, the Get List object replaces the '!ITEM!' tag.

The Get Details query usually includes the '!FIELDS!' tag to allow the list of fields to be dynamically created based on the fields entered by the user in the Portal. The '!FIELDS!' tag is replaced at run-time with a comma separated list of fields taken from the Portal interface.

Any query-able object and details may be collected from IBM Rational Change.

#### **5.6.14. Configuring a IBM Rational DOORS Source**

This source allows the Collector to scan projects and modules, as well as collect module data from IBM Rational DOORS. This source requires that the DOORS Windows Client be installed and configured on the same machine as the Collector before attempting collection.

This source requires the following parameters.

Parameter	Description
Path to client	The location of the DOORS client on the current machine. Click the <b>button</b> to the side to browse for the file on the computer.
DOORS User name	The name of a DOORS Administrator, someone who has access to all of the projects and modules from which data needs to be collected
Password	The DOORS user password
Database	The port number and the server name in the format <port number>@<server>.
Collect attribute	A boolean value that indicates if a module should be collected. See Notes below.
Type attribute	A DOORS attribute used to delineate between Portal Interfaces and the different attributes that the user would want to collect for different modules.  See Notes below.

Notes:

This source uses a set of DXL scripts located in Scripts\DOORS subfolder of the Collector. You may tailor these scripts if needed. It is recommended that you configure collection with the standard scripts before attempting to tailor them.

The "Collect Attribute" and "Type Attribute" must be added to the DOORS database as module level attributes in order for modules and module data to be collected by the default DXL scripts included with the Collector.

The "Collect Attribute" should be a boolean value. The Collector will only collect those modules that have the Collect Attribute set to true.

The "Type Attribute" is used to select an interface from the Portal to use for each different module type. For example, you may want requirements metrics from

requirements modules and test metrics from DOORS test modules. This attribute value is used to select the associated DOORS interface in the Portal.

### 5.6.15. *Configuring a IBM Rational Synergy Source*

This source requires a IBM Rational Synergy Windows client be installed and configured before attempting collection. To set up and run an IBM Rational Synergy collection, IBM Rational Synergy must be enabled to run the “ccm” Command Line Interface (CLI). Please refer to Synergy help files to enable the CLI.

This source requires the following parameters.

Parameter	Description
CCM query client path	The location of the ccm.exe file on the current computer. Click the <b>button</b> to the side to browse for the file on the computer. The default value in the collector is the default value for IBM Rational Synergy 6.4 installations.
Database path	The location of the database that is being collected.
User name	A user name for a database administrator who would have access to all of the releases and CRs that need to be collected.
Password	Password for the user supplied
Get List Query	A ccm CLI command to retrieve list of items See Notes below.
Get Details Query	A ccm CLI command to get details for each item returned by the Get List Query See Notes below

Notes:

The "Get List Query" and "Get Details Query" are CLI commands to retrieve items and item details. The default values select all of the releases in the database, and then collect all problems (defects or enhancements) from each release.

The Get List query can be written to return a list of objects found in the IBM Rational Change database. The Get Details query is run for each object found by the Get List query. When the Get Details query executes, the Get List object replaces the '!ITEM!' tag.

The Get Details query usually includes the '!FIELDS!' tag to allow the list of fields to be dynamically created based on the fields entered by the user in the Portal. The '!FIELDS!' tag is replaced at run-time with a comma separated list of fields taken from the Portal interface.

Any query-able object and details may be collected from IBM Rational Change.

### **5.6.16. Configuring an XML ADO.NET Source**

This data source allows XML files which use the .NET 2.0 ADO.NET xml schema to be collected. Each XML file should have been produced by an application that produces XML files using the .NET XML schema (also called the “recordset schema”) At present, other XML formats are not accepted (though we do plan on implementing a general-purpose XML with XSLT capability)

This source requires the following parameters:

<b>Parameter</b>	<b>Description</b>
File selection	One of: <ul style="list-style-type: none"> <li>• all files in folder</li> <li>• a listing file</li> <li>• data-stamped files in folder</li> </ul> See Notes below.
Folder name	Name of the folder containing files OR the folder and file name of the list file. See Notes below.
File encoding	Encoding of the file, one of: <ul style="list-style-type: none"> <li>• System (use the current system default encoding)</li> <li>• UTF-7</li> <li>• UTF-8</li> <li>• Unicode</li> </ul>

## Notes:

The list file capability was modified in version 3.6.1 to allow either a folder name or a folder and file name. For example, if you select just a folder for the list file parameter, then the list file is called “(selected folder)\list.txt”. Alternatively, once you select a folder name using the UI, you may type in the text “\myfiles.txt” to indicate the name of the list file is called “(selected folder)\myfiles.txt”.

Date stamped files are file names which contain a base name substring and a date substring, where the date substring has the format YYYYMMDD. When you specify date-stamped files, you may keep all historic and current files in the same folder however you must name them using this format. An example of a date-stamped file name is “defects\_projectA\_20080601.csv”. The collection date specified by the user is compared to the date substring (in this case “20080601”) to determine if the data should be collected. So, if the Collection were run for the collection date 1 June 2008, then the sample file would be collected.



## 6. Portal Operations

Once IBM Rational Dashboard has been installed and your Administrator has established Interfaces for the Collector(s), security and default settings, managers can create and track their own project in IBM Rational Dashboard. This chapter shows you how to setup a basic Project (Unit), assign collected Items and Information Needs to it and display the resulting Graphs.

It is quite simple to get started. Let's follow these steps to create a new project:

1. Create a Folder
2. Create a new Unit within the Folder
3. Set a default Schedule for the Unit
4. Assign collected Items
5. Assign Information Needs to your Unit
6. Refresh the data within the Unit
7. See your Graphs

## 6.1. Define a Unit

Ensure that you select the folder in which you want to create a new basic unit (Figure 11).

1. Click on the Edit sub-tab.
2. Click on the add Unit button.

The screenshot shows the 'Status' window in the IBM Rational Dashboard. The window has a menu bar with 'view', 'filter', 'edit', and 'assign'. Below the menu bar are two buttons: 'add folder' and 'add unit'. The left pane displays a tree view of the project structure, including folders like 'B-F400 Project', 'B-F400 Releases', 'CARS Project', 'Java Kit 1.0', 'Java Kit 2.1', 'Java Kit Product Summary', 'Software Management Samples', and 'Samples Folder'. The right pane is the 'add unit' dialog, which is titled 'New Unit 2/9/2008 12:47:50 PM'. It contains the following fields and options: 'Owner' is set to 'no selection'; there are radio buttons for 'basic unit' (which is selected) and 'summary unit'; a note states 'A basic unit contains manual, calculated and collected progress data.'; 'Template' is set to 'no selection'; 'Dates' are set to 'Start date: January 1 2008' and 'End date: December 31 2008'; and a checkbox for 'For demonstration only?' is unchecked. An 'add' button is located at the bottom of the dialog.

Figure 11

3. Enter a Title for the Unit (e.g. Project A1)
4. Select the Owner of the Unit (e.g. your login account)
5. Select "basic unit" in this example.
  - A basic unit uses manual and collected data to update the graphs and status of its information needs.
  - A summary unit uses data from basic units to update the status of its information needs.
6. Select a Unit Template (if any) to be applied (e.g. DOORS Template). If you don't specify a Unit template at Unit creation then Information Needs will need to be manually assigned to the Unit later.
7. Set the Start Date and End Date for the Unit (e.g. Jan 1, 2008 and Dec 31, 2008)
8. Set the "For demonstration only?" checkbox
  - The purpose of the "For demonstration only?" option is to create the Unit with the sample data built into the IBM Rational Dashboard. In the event that you do not have any collected data in the repository and you would simply like to browse the IBM Rational Dashboard to become familiar with its interface check this box.
  - If you do have collected data and would like to use that data, leave this box unchecked.

9. Click add

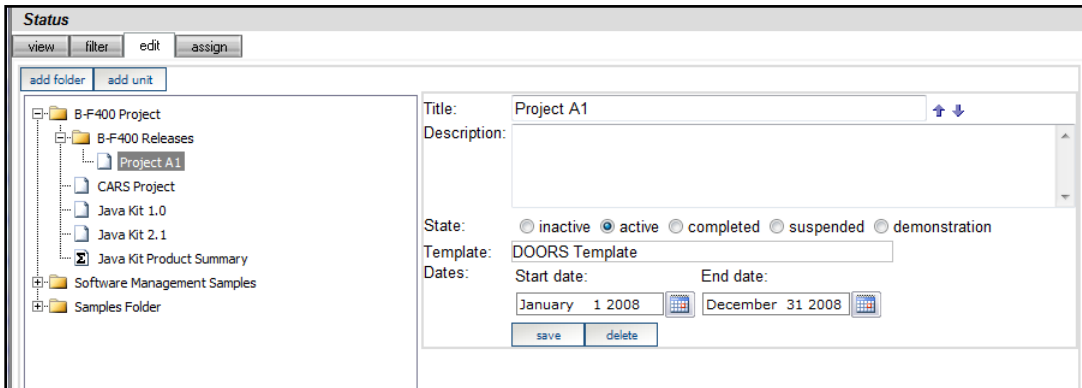


Figure 12

Once the Unit has been created:

1. Enter Description. Enter a description for the Unit (optional)
2. Enter State of Unit. Select one of these states:
  - inactive – Unit is being configured for use but data will not be collected
  - active – Unit currently active and data is collected and updated for the Unit (default)
  - completed – Unit has completed, no further action
  - suspended - Unit collection and refresh processing will not be performed
  - demonstration - Unit is not for actual project management - sample data is generated for all graphs and no collection is performed
3. Click Save

## 6.2. Set the Default Schedule for the Unit

You may need to specify a default Schedule for the Unit. The default Schedule is applied to all items assigned to the Unit. If you specified a Unit Template that contains a default Schedule, then you can skip this step. The standard “DOORS Unit Template” does not have a default schedule. The standard “Sample Unit Template” does have a default schedule.

1. Select the *view* sub-tab on the Status tab
2. Ensure the desired Unit is selected (Figure 13)

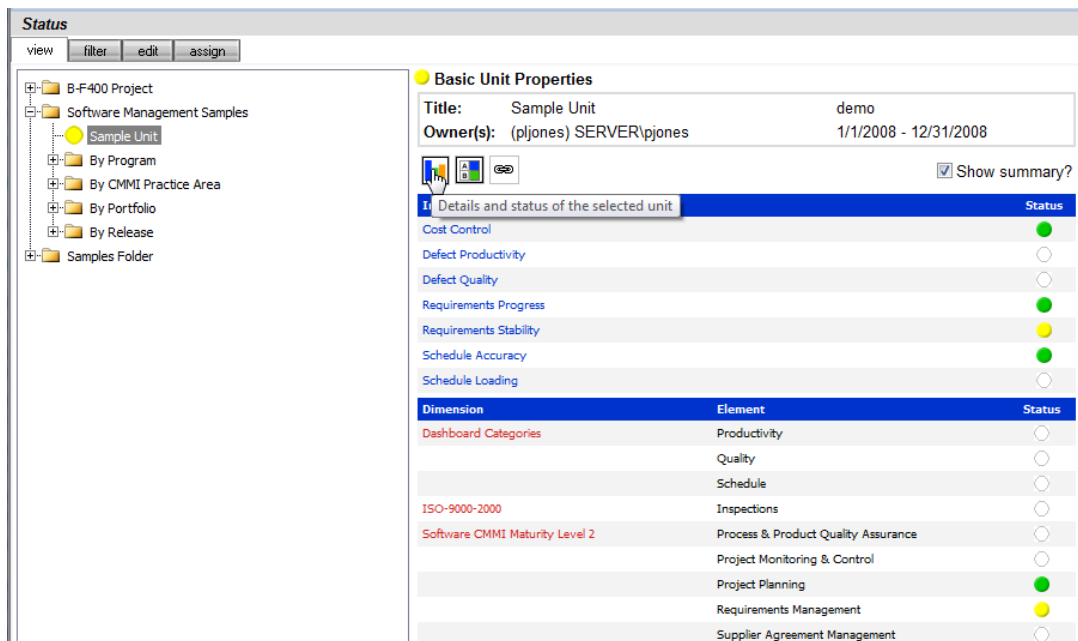


Figure 13

3. Click the details button (the vertical bar icon)
4. Click on the Definition Panel on the left-hand side (Figure 14)
5. Click on Unit Properties
6. Choose the *basic schedule mode* radio button and *Monthly Schedule 2008 – 2009* from the drop-down list
7. Click save

**Unit Status** Status -> Unit Status (Sample Unit) [demo]

---

**Views**

---

**Dashboards**

---

**Data**

---

**Definition**

- Attributes
- Information Needs
- Item Properties
- Phases
- Security
- Unit Properties

**Unit Definition**

Title:

Owner:

State:  inactive  active  completed  suspended  demonstration

Refresh Order:

Schedule:  basic schedule mode  advanced schedule mode

Progress Report:

Office Template:

Dates:

Start Date:

End Date:

Description:

URL:

Figure 14

### 6.3. Assign Collected Items

The assign sub-tab is used to assign collected items to the Unit. Once you click on assign, you will see two sub-tabs on the right hand side: by date and by source each containing an Item List. The Item List (on the right side) will be empty until a collection is performed. If there are no items shown, then you must run a collection prior to this step.

Please refer to the IBM Rational Dashboard online help for further information on how to run a collection from a source application most relevant to your project.

You can either assign items using the by date sub-tab or the by source sub-tab. Both are explained in the following subsections.

#### 6.3.1. Assign Sub-Tab with By Date Options

The “By Date” options (Figure 15) allow you to select items that have been collected by using their collect date.

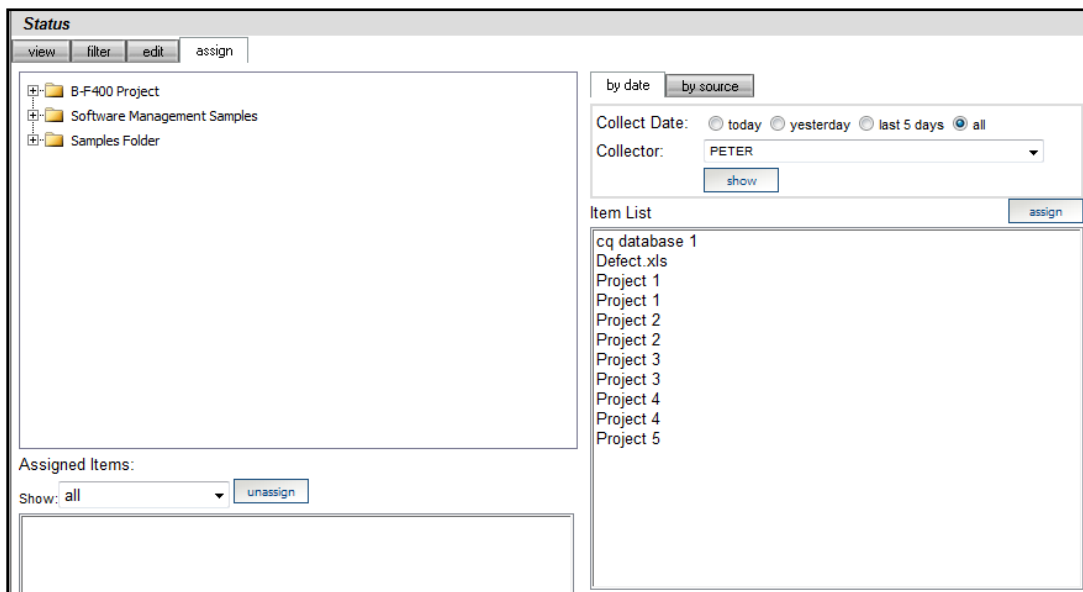


Figure 15

Collect Date: Select one of the following options:

- today – show items collected today
- yesterday – show items that were collected yesterday

- last 5 days – show all items collected in the last 5 days
- all – show all collected items

Collector: Select the collector from the drop down list

Show: Once the Collect Date and Collector are selected, click the show button to display the corresponding items in the Item List.

For some Interfaces, e.g. DOORS, “containers” could also be listed. Containers are collections of items, e.g. DOORS projects, and, if selected, may assign multiple items to the Unit.

Ensure that you select a basic unit on the left (below, the “Project A1” unit) and an item on the right (below, the “Defects.xls” item), and then press the Assign button from the right side.

Assign: Click on the item(s) you wish to assign to the Unit in the Item List and click the assign button. The item(s) will be added to the Assigned Items list below the Organization Tree

### **6.3.2. Assign Sub-Tab with By Source Options**

The operation of the “by source” assignment is similar to “by date” assignment, but differs in the method for locating collecting items. The “By Source” options allow you to review and optionally assign items that were collected from a selected data source.

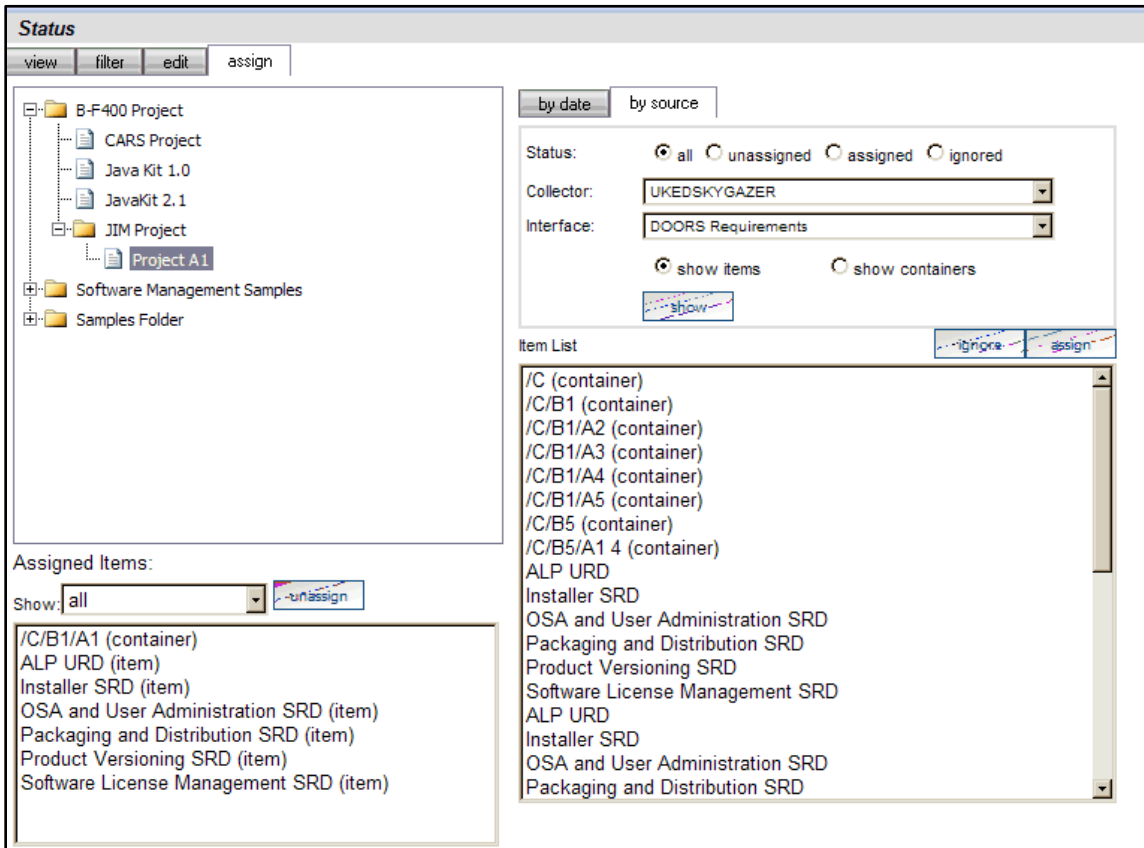


Figure 16

Status: Click on a radio button to choose the Status of those items to be displayed:

- all – Displays all of the items for the specified Collector and Interface.
- unassigned – Displays all unassigned items for the specified Collector and Interface
- assigned – Displays all of the assigned items for the specified Collector and Interface
- ignored – Displays all of the items that have been marked as ignored for the specified Collector and Interface

Collector: choose the Collector from the drop down list

Interface: choose the Interface from the drop down list

Select either “show items” or “show containers” radio button, to show the list of individual collected Items (for example DOORS modules) or a list of collected Containers.



Depending on the storage structure of data within the source, containers may not be present in the collected data. Assigning a Container to a Unit results in the assignment of all Items within the container. For example, if you assign a DOORS Project (which is a container) to a unit, then all modules within the DOORS project are assigned to the unit.

Show: Click on the show button to display the Item List

Ensure the Unit you wish to assign items to is selected in the organization tree

Select item(s) in the Item List and click the assign button to add them to the Unit. To remove an item from the list of items, select the item from the Item List and click the ignore button. Once the item has been marked as “ignored,” it will no longer display in the Item List.

Assigned items will display in the Assigned Items list below the Organization Tree. To remove an assigned item, select the item from the Assigned Items list and click on the un-assign button. The item will be removed from the list on the left side.

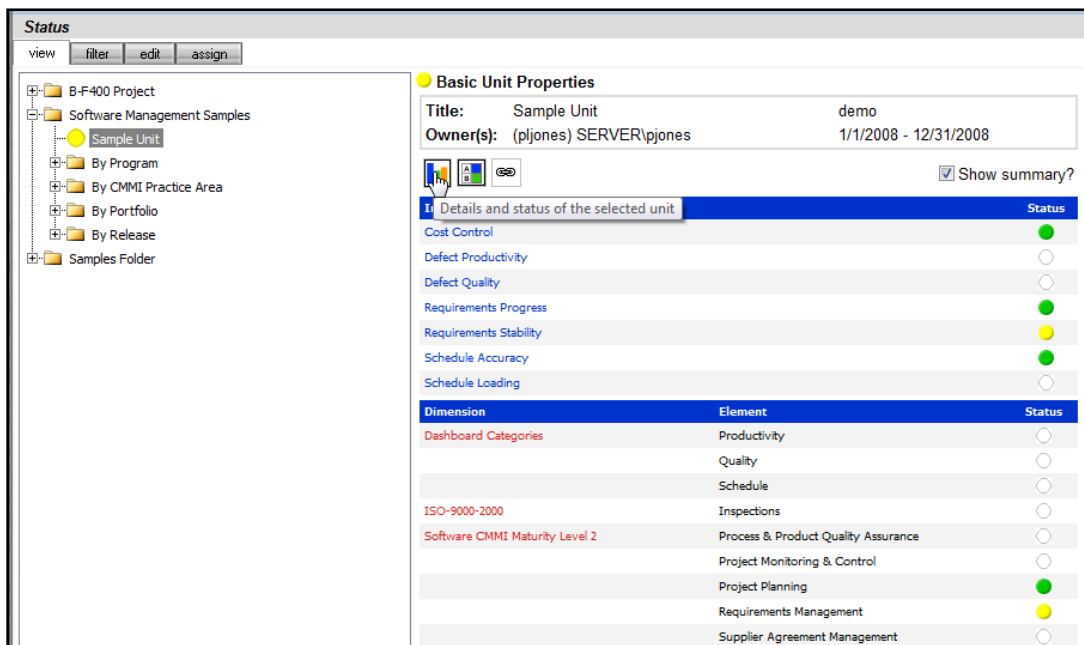
If your Item List is empty, then either the collection has not run or hasn't run successfully.

## 6.4. Assign Information Needs to your Unit

If a Unit Template was applied when creating the Unit then Information Needs may have already been automatically added to the Unit. The standard DOORS Template contains two Information Needs (Requirements Progress and Requirements Stability). If you specified this template (or another one which contains Information Needs) when the Unit was created then you may skip this step.

To add Information Need(s) to a Unit they must already exist, either by default (shipped with the Portal) or have been created in the Library tab.

1. Click on the Status tab (view sub-tab)
2. Click on the Unit in the Organization Tree. Unit Properties will appear on the right pane



The screenshot shows the 'Status' tab interface. On the left is an organization tree with 'Sample Unit' selected. The right pane shows 'Basic Unit Properties' for 'Sample Unit' with the following details:

- Title: Sample Unit
- Owner(s): (pljones) SERVER\pljones
- demo
- 1/1/2008 - 12/31/2008
- Show summary? (checked)

Below the properties is a table titled 'Details and status of the selected unit' with columns for Dimension, Element, and Status.

Dimension	Element	Status
Cost Control		●
Defect Productivity		○
Defect Quality		○
Requirements Progress		●
Requirements Stability		●
Schedule Accuracy		●
Schedule Loading		○
Dashboard Categories	Productivity	○
	Quality	○
	Schedule	○
ISO-9000-2000	Inspections	○
Software CMMI Maturity Level 2	Process & Product Quality Assurance	○
	Project Monitoring & Control	○
	Project Planning	●
	Requirements Management	●
	Supplier Agreement Management	○

Figure 17

3. Click on the details button to view the Unit Status page
4. On the Unit Status page, click the Definition panel on the left navigation pane

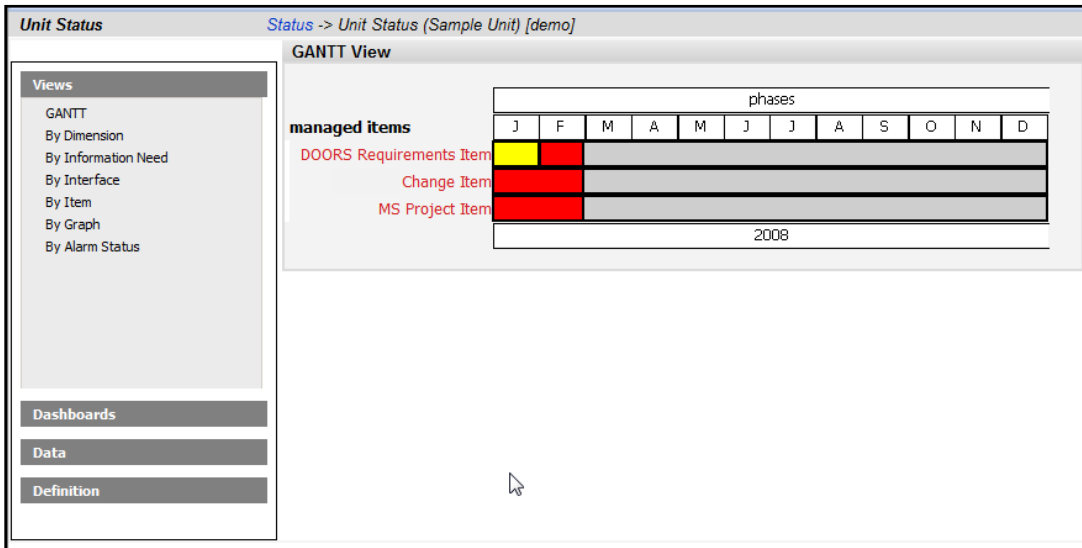


Figure 18

5. Click the Information Needs command (Figure 19)

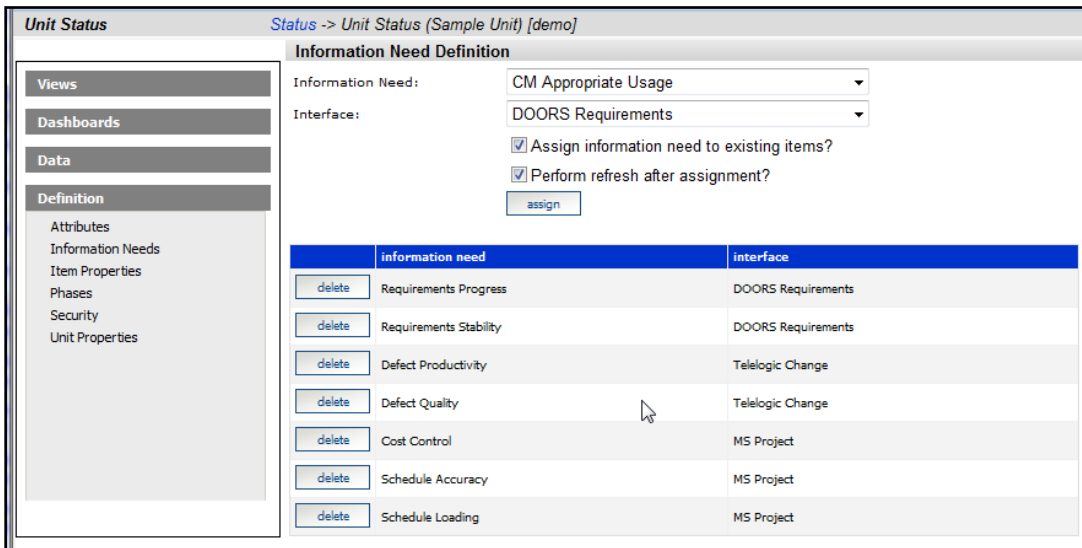


Figure 19

From the Information Need Definition page on the right-hand side:

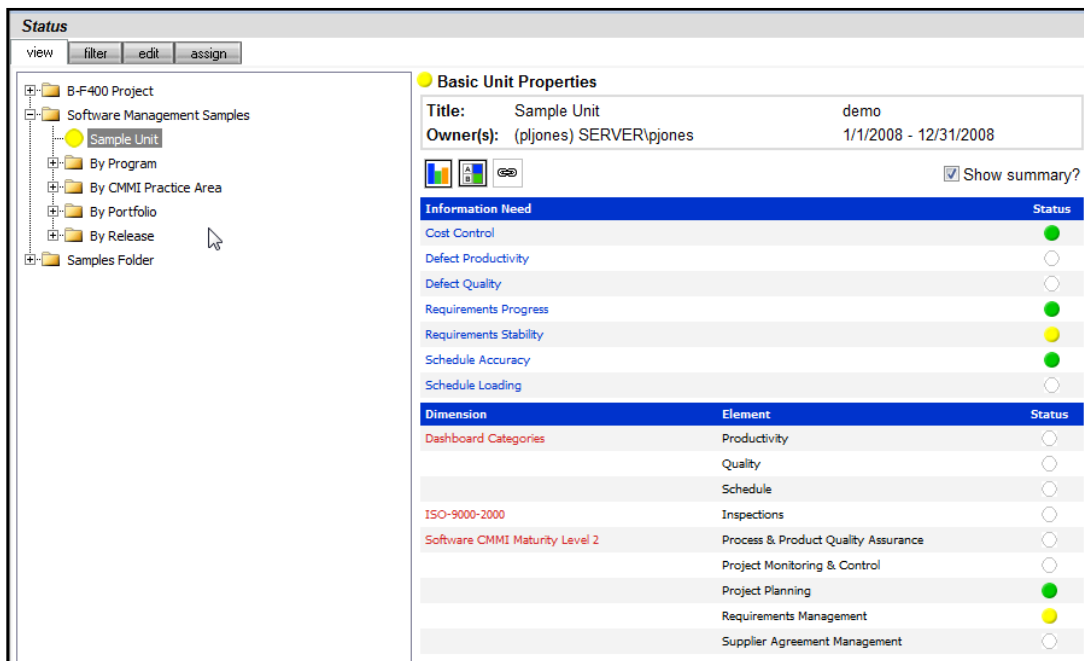
6. Select the Information Need (choose one such as Requirements Progress) from the drop down list
7. Select the Interface from the drop down list
8. Check the 'Assign information need to existing items?' box
9. Check the 'Perform refresh after assignment?' box

10. Click the assign button to add the Information Need to the Unit

## 6.5. Refresh Data within the Unit

The next step is to ensure that the data and graphs contained within the Unit are refreshed and up-to-date.

1. Select the view sub-tab on the Status tab
2. Ensure the Unit is selected (Figure 20)



The screenshot shows the 'Status' tab interface. On the left is a tree view with the following structure:

- B-F400 Project
  - Software Management Samples
    - Sample Unit** (highlighted)
    - By Program
    - By CMMI Practice Area
    - By Portfolio
    - By Release
    - Samples Folder

On the right, the 'Basic Unit Properties' section displays:

- Title:** Sample Unit
- Owner(s):** (pljones) SERVER\pljones
- demo**
- 1/1/2008 - 12/31/2008**
- Show summary?

Below this is the 'Information Need' table:

Information Need	Status
Cost Control	●
Defect Productivity	○
Defect Quality	○
Requirements Progress	●
Requirements Stability	●
Schedule Accuracy	●
Schedule Loading	○

Below that is the 'Dimension' table:

Dimension	Element	Status
Dashboard Categories	Productivity	○
	Quality	○
	Schedule	○
ISO-9000-2000	Inspections	○
	Process & Product Quality Assurance	○
Software CMMI Maturity Level 2	Project Monitoring & Control	○
	Project Planning	●
	Requirements Management	●
	Supplier Agreement Management	○

Figure 20

3. Click the details button
4. Click the Definition panel to expand the commands in that panel
5. Click the Unit Properties command (Figure 21)

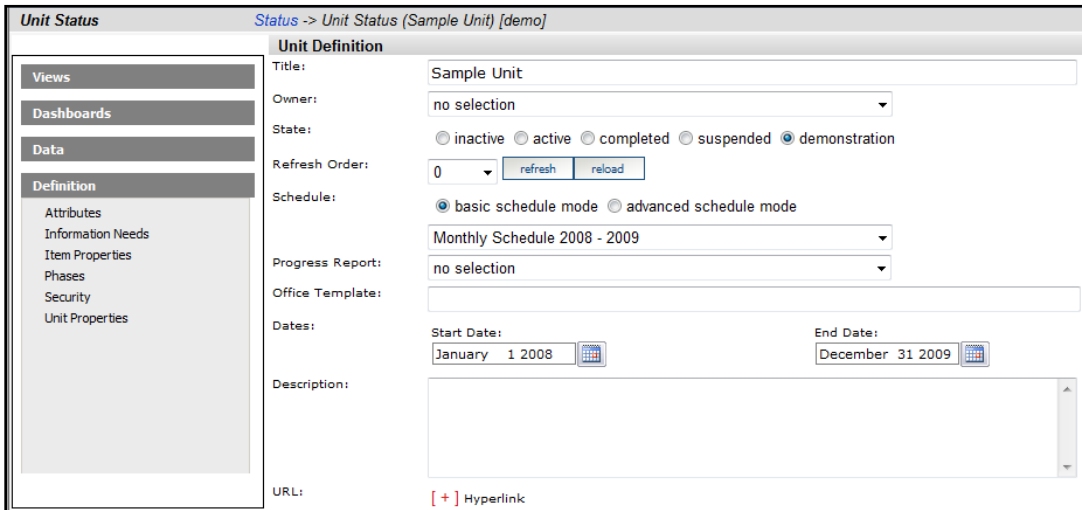


Figure 21

6. Click the refresh button to display the Refresh page (Figure 22).

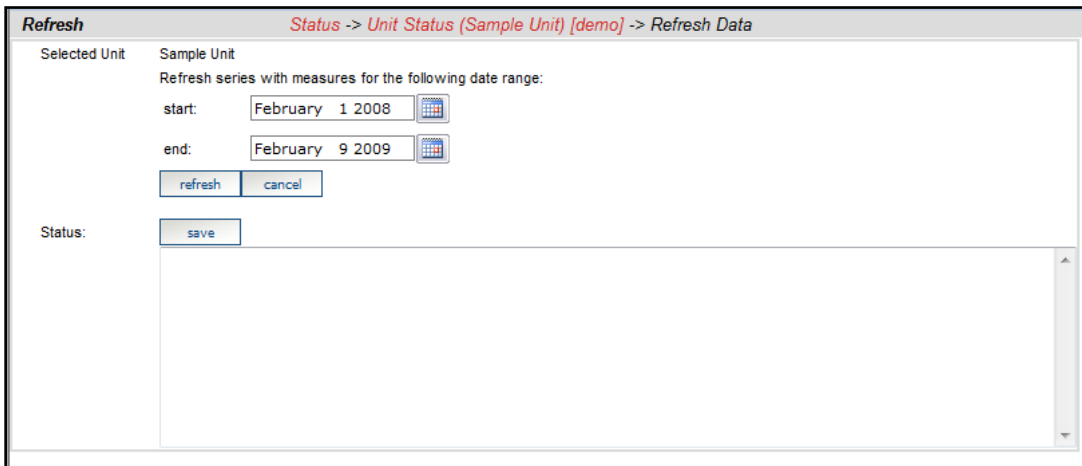


Figure 22

7. Select the start and end date for the refresh
8. Click the refresh button

## 6.6. See Your Graphs

Unless the Unit has been created with demonstration data, an Information Need and one or more collected Items must be assigned to the Unit before any Graphs will display.

If a Unit Template was applied when creating the Unit, then Information Needs may have already been automatically added to the Unit.

From the Unit Status page, select on your Item in the Dashboards panel in the left hand navigation bar. Providing that the data collection has successfully completed, the data should now show in the Graphs. If data is not displaying, please consult your administrator.

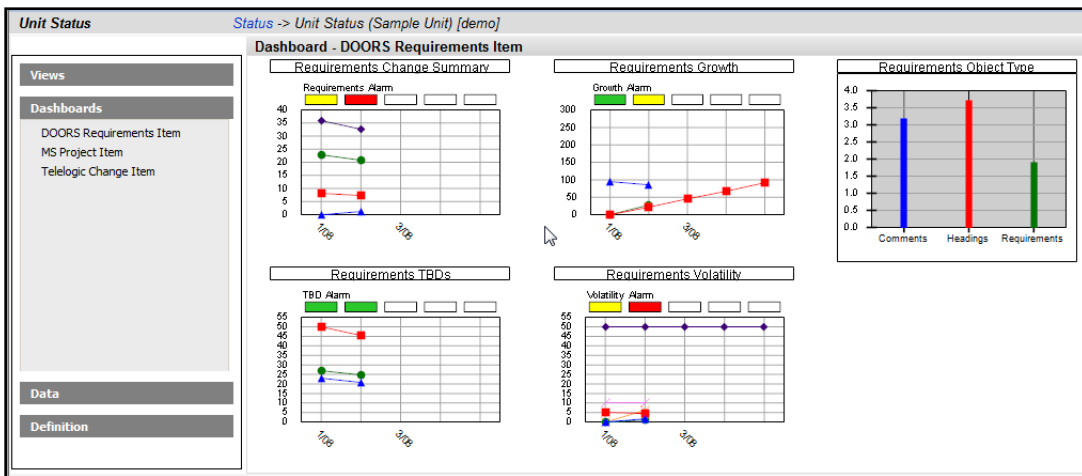


Figure 23

## 7. Other Resources

IBM Rational Dashboard provides a framework for performance measurement and compliance tracking during normal project execution. A common compliance model is the Maturity Model framework from SEI. A built-in Software CMMI Level 2 Dimension tracks the Information Needs against CMMI Level 2 KPAs. The Dimension reports CMMI Level 2 compliance using the Information Needs and status generated during normal project execution. Using the built-in Information Needs and Dimensions, project management compliance becomes a by-product of effective ongoing management using Dashboard.

In IBM Rational Dashboard, a Dimension corresponds to one of the organization's written sources for compliance. Managers may create as many Dimensions as needed based on their organizational strategy. Within a single compliance source, typically number of sub areas exists. For example, in a maturity model, several process areas exist such as requirements management or software quality. Dimensions were created in IBM Rational Dashboard to reflect each of these process areas.

The Library contains a set of software project management best practice Information Needs for common engineering process, such as requirements engineering and configuration management. These Information Needs can become a part of templates used to quick-start project performance measurement for a customer organization.

For instructions on how to create and add Information Needs and Dimensions, please refer to the online help.



## 8. Contact Information

This chapter contains the following topics:

- Contacting IBM Rational Software Support
- Prerequisites
- Submitting problems
- Other information

---

### Contacting IBM Rational Software Support

If the self-help resources have not provided a resolution to your problem, you can contact IBM Rational Software Support for assistance in resolving product issues.

**Note:** If you are a heritage Telelogic customer, you can go to <http://support.telelogic.com/toolbar> and download the IBM Rational Telelogic Software Support browser toolbar. This toolbar helps simplify the transition to the IBM Rational Telelogic product online resources. Also, a single reference site for all IBM Rational Telelogic support resources is located at <http://www.ibm.com/software/rational/support/telelogic/>

---

### Prerequisites

To submit your problem to IBM Rational Software Support, you must have an active Passport Advantage® software maintenance agreement. Passport Advantage is the IBM comprehensive software licensing and software maintenance (product upgrades and technical support) offering. You can enroll online in Passport Advantage from <http://www.ibm.com/software/lotus/passportadvantage/howtoenroll.html>.

- To learn more about Passport Advantage, visit the Passport Advantage FAQs at [http://www.ibm.com/software/lotus/passportadvantage/brochures\\_faqs\\_quickguides.html](http://www.ibm.com/software/lotus/passportadvantage/brochures_faqs_quickguides.html).
- For further assistance, contact your IBM representative.

To submit your problem online (from the IBM Web site) to IBM Rational Software Support, you must additionally:

- Be a registered user on the IBM Rational Software Support Web site. For details about registering, go to <http://www-01.ibm.com/software/support/>.
- Be listed as an authorized caller in the service request tool.

## Submitting Problems

### To submit your problem to IBM Rational Software Support:

1. Determine the business impact of your problem. When you report a problem to IBM, you are asked to supply a severity level. Therefore, you need to understand and assess the business impact of the problem that you are reporting.

Use the following table to determine the severity level.

Severity	Description
1	The problem has a <i>critical</i> business impact: You are unable to use the program, resulting in a critical impact on operations. This condition requires an immediate solution.
2	This problem has a <i>significant</i> business impact: The program is usable, but it is severely limited.
3	The problem has <i>some</i> business impact: The program is usable, but less significant features (not critical to operations) are unavailable.
4	The problem has <i>minimal</i> business impact: The problem causes little impact on operations or a reasonable circumvention to the problem was implemented.

2. Describe your problem and gather background information, When describing a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Rational Software Support specialists can help you solve the problem efficiently. To save time, know the answers to these questions:
  - What software versions were you running when the problem occurred?

To determine the exact product name and version, use the option applicable to you:

  - Start the IBM Installation Manager and select **File > View Installed Packages**. Expand a package group and select a package to see the package name and version number.

- Start your product, and click **Help > About** to see the offering name and version number.
  - What is your operating system and version number (including any service packs or patches)?
  - Do you have logs, traces, and messages that are related to the problem symptoms?
    - Can you recreate the problem? If so, what steps do you perform to recreate the problem?
    - Did you make any changes to the system? For example, did you make changes to the hardware, operating system, networking software, or other system components?
    - Are you currently using a workaround for the problem? If so, be prepared to describe the workaround when you report the problem.
3. Submit your problem to IBM Rational Software Support. You can submit your problem to IBM Rational Software Support in the following ways:
- **Online:** Go to the IBM Rational Software Support Web site at <https://www.ibm.com/software/rational/support/> and in the Rational support task navigator, click **Open Service Request**. Select the electronic problem reporting tool, and open a Problem Management Record (PMR), describing the problem accurately in your own words. For more information about opening a service request, go to <http://www.ibm.com/software/support/help.html>
  - You can also open an online service request using the IBM Support Assistant. For more information, go to <http://www-01.ibm.com/software/support/isa/faq.html>.
  - **By phone:** For the phone number to call in your country or region, go to the IBM directory of worldwide contacts at <http://www.ibm.com/planetwide/> and click the name of your country or geographic region.
  - **Through your IBM Representative:** If you cannot access IBM Rational Software Support online or by phone, contact your IBM Representative. If necessary, your IBM Representative can open a service request for you. You can find complete contact information for each country at <http://www.ibm.com/planetwide/>.

If the problem you submit is for a software defect or for missing or inaccurate documentation, IBM Rational Software Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Rational Software Support provides a workaround that you can implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM Rational Software Support Web site daily, so that other users who experience the same problem can benefit from the same resolution.

Other  
Information

For Rational software product news, events, and other information, visit the IBM Rational Software Web site on <http://www.ibm.com/software/rational/>.

## Notices

This information was developed for products and services offered in the U.S.A. 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 written license inquiries to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

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

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106-0032, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**

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 states 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk. IBM may use or distribute any of the information you supply 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:

Intellectual Property Dept. for Rational Software  
IBM Corporation  
1 Rogers Street  
Cambridge, Massachusetts 02142  
U.S.A.

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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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 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 the names and addresses used by an actual business enterprise is entirely coincidental. If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## **Trademarks**

IBM, the IBM logo, ibm.com, DOORS, Passport Advantage, and Rational 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 [www.ibm.com/legal/copytrade.html](http://www.ibm.com/legal/copytrade.html).

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

Other company, product or service names may be trademarks or service marks of others.