



Quick answers to common problems

Microsoft System Center Reporting Cookbook

Over 40 practical recipes to help you plan, create, and manage reports efficiently for all components of Microsoft System Center

*Foreword by Kathleen Wilson,
Architect, Datacenter and Cloud Center of Excellence at Microsoft Corporation*

Samuel Erskine
Kurt Van Hoecke

Dieter Gasser
Nasira Ismail

[PACKT] enterprise 
PUBLISHING professional expertise distilled

www.allitebooks.com

Microsoft System Center Reporting Cookbook

Over 40 practical recipes to help you plan, create,
and manage reports efficiently for all components of
Microsoft System Center

Samuel Erskine

Dieter Gasser

Kurt Van Hoecke

Nasira Ismail

[PACKT] enterprise 
PUBLISHING professional expertise distilled

BIRMINGHAM - MUMBAI

Microsoft System Center Reporting Cookbook

Copyright © 2015 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: March 2015

Production reference: 1240315

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78217-180-5

www.packtpub.com

Credits

Authors

Samuel Erskine
Dieter Gasser
Kurt Van Hoecke
Nasira Ismail

Reviewers

Andreas Baumgarten
Steve Buchanan
Ronnie Isherwood
Marcello Rosenberg
Nataschia Heil

Commissioning Editor

James Jones

Acquisition Editor

James Jones

Content Development Editor

Arun Nadar

Technical Editors

Manan Badani
Chinmay Puranik

Copy Editors

Karuna Narayanan
Vikrant Phadke

Project Coordinator

Neha Bhatnagar

Proofreaders

Ting Baker
Simran Bhogal
Mario Cecere
Safis Editing
Maria Gould
Paul Hindle

Indexer

Monica Ajmera Mehta

Production Coordinator

Shantanu N. Zagade

Cover Work

Shantanu N. Zagade

Foreword

Many organizations that have deployed System Center 2012 R2 struggle with delivering the reports that their stakeholders and management expect and need to make informed business decisions. System Center has a powerful reporting platform, and possibly for this reason, you have deployed System Center 2012 R2. Yet, you might not have benefitted from the great reporting that it makes possible.

This cookbook provides you with benefits such as understanding the goals of reporting, how to plan and use reporting in System Center, and a collection of recipes from which you can create your own reports and dashboards to enable a transparent view of your service and IT performance capabilities to management.

The chefs (authors) of this cookbook Samuel Erskine, Dieter Gasser, Kurt Van Hoecke, and Nasira Ismail are experienced System Center consultants. They have taken out time to write down their favorite recipes for reporting to enable all System Center customers to cook the best reports, leveraging the power of System Center 2012 R2's reporting capabilities.

The importance of being able to extract the necessary data from reporting tools is crucial for IT organizations in order to make informed decisions, improve service delivery, and innovate. This cookbook will show you why reporting is crucial and how to create your own System Center reports.

I am honored to write this foreword as well as be the early taster of these wonderful reporting recipes. Bon appétit!

Kathleen Wilson

Architect, Datacenter and Cloud Center of Excellence at Microsoft Corporation

About the Authors

Samuel Erskine (IT Driving Fellow) is an independent IT consultant, who specializes in Service Manager and Configuration Manager. He is the content designer and one of the authors of *Microsoft System Center 2012 Service Manager Cookbook* and *Microsoft System Center 2012 Orchestrator Cookbook*, both by Packt Publishing; a contributing author to *System Center 2012 Configuration Manager Unleashed*; and a coauthor of *System Center 2012 Service Manager Unleashed*, Sams Publishing. With over 18 years of IT experience, he focuses on providing training and service management consultancy services in the United Kingdom and other locations. He also writes blogs at www.itprocessed.com.

I would like to thank my friend, Fernández-Peñaranda, for the picture on the book cover. Fernández-Peñaranda can be reached at <https://es-es.facebook.com/victorfphoto>.

This book was possible only due to a great team and a great deal of research. The biggest breakthrough was my trip to Ottawa to see my friends at Provance. This was where I discovered the course content delivered by Simon Allardice from Lynda.com. I would like to acknowledge and thank my coauthors Dieter Gasser, Kurt Van Hoecke, and Nasira Ismail for delivering my vision for this book.

A big thank you goes to my nearest and dearest who continue to let me geek out in my den.

This book is dedicated to my mother, Dora Erskine, for teaching me how to simplify and share my knowledge.

Dieter Gasser is an IT consultant and the cofounder of Syliance IT Services, headquartered in Switzerland. He has a strong focus on the delivery and customization of Service Manager.

Dieter has been working in IT for more than 13 years, and has focused on Microsoft technologies. He started his career as an application and database developer, and later became the IT manager of an international manufacturing company.

In 2010, he entered the systems management and automation market. With both his technical and managerial backgrounds, he has focused on Service Manager. Together with his colleagues, he delivers data center management and automation solutions based on Microsoft System Center to customers all across Switzerland.

Kurt Van Hoecke is an MVP and managing consultant at Inovativ Belgium. He focuses on the System Center product suite, including Service Manager, Configuration Manager, and Orchestrator. He is the coauthor of *System Center 2012 Orchestrator Unleashed*, Sams Publishing; *System Center 2012 Service Manager Unleashed*, Sams Publishing; and a contributor to *System Center Service Manager 2010 Unleashed*. He provides consultancy, development services, and blogs at www.scug.be/scsm and www.authoringFriday.com.

Nasira Ismail is an experienced project manager and service delivery consultant. She has result-oriented experience in incident, problem, change, and configuration management processes for medium and large enterprises. She has a strong grasp of the methods used to design processes through to implementation, with technologies (including System Center). She also leads training, mentoring, and coaching sessions to bring about a proactive culture and behavioral change in her organizations.

About the Reviewers

Andreas Baumgarten is a Microsoft MVP and works as an IT architect with the German IT service provider, H&D International Group. He has been working as an IT professional for more than 20 years. He has been associated with Microsoft for a long time, and has more than 14 years of experience as a Microsoft Certified Trainer.

Since 2008, Andreas has been responsible for the field of Microsoft System Center technology consulting, and has taken part in Microsoft System Center Service Manager 2010, 2012, and 2012 R2. Additionally, he has participated in the technology adoption programs of Microsoft System Center Orchestrator 2012 and 2012 R2 with H&D.

With his in-depth technical know-how and experience with the Microsoft System Center product family and IT management, he now designs and develops private and hybrid cloud solutions for customers all across Germany and Europe.

In October 2012, 2013, and 2014, Andreas was awarded the Microsoft Most Valuable Professional (MVP) title for System Center Cloud and Datacenter Management. He is a coauthor of *Microsoft System Center 2012 Service Manager Cookbook*, *Microsoft System Center 2012 Orchestrator Cookbook*, and *Microsoft System Center 2012 R2 Compliance Management Cookbook*, all by Packt Publishing.

Steve Buchanan is a regional solution lead with Concurrency, a three-time Microsoft System Center MVP, and the author of several technical books focused on the System Center platform.

Steve has been an IT Professional for over 15 years with various positions, ranging from infrastructure architect to IT manager. He focuses on transforming IT departments through DevOps, service management, systems management, and cloud technologies.

He has authored the following books: *System Center 2012 Service Manager Unleashed*, Sams Publishing; *Microsoft System Center Data Protection Manager 2012 SP1*, Packt Publishing; and *Microsoft Data Protection Manager 2010*, Packt Publishing. He holds certifications in A+, Linux+, MCSA, MCITP (Server Administrator), and MCSE (Private Cloud).

He is active in the System Center community and enjoys blogging about his adventures in the world of IT at www.buchatech.com. You can also follow him on Twitter at @buchatech for his latest blog posts.

It was great working with Sam Erskine, Dieter Gasser, and the other authors on this book. I also want to thank Sam for bringing me into this project.

Ronnie Isherwood, MCITP and MBCS, is a technology entrepreneur who has worked in the IT industry for more than 20 years, including 15 years of experience in delivering infrastructure, systems management, and virtualization technologies to the public sector and in financial and legal companies. He has worked with Microsoft Learning as a subject matter expert and technical reviewer, contributing to the several MCSE courses on the server and the cloud. In 2014, he cofounded a software development company, JEB3.com, where he designs SaaS solutions and mobile applications for the financial services industry. He is committed to the IT community and is the founder of a Microsoft Windows user group and chairman of the Chartered Institute for IT in Jersey.

I'd like to thank Samuel Erskine for this opportunity, and Mélinda Isherwood for supporting me tirelessly with all my technology endeavors.

Marcello Rosenberg is an IT infrastructure management specialist with more than 5 years of experience in IT. He has developed broad expertise in the various components of the Microsoft System Center product. Marcello is working as a System Center engineer and consultant for Syliance IT Services, headquartered in Switzerland, where he delivers professional services in the area of data center management, automation, and cloud computing to customers in Switzerland and worldwide.

Natascia Heil is a senior technical analyst, with extensive experience in server monitoring and data center automation. She has over 20 years of experience in IT and is an MCSE in Microsoft Private Cloud. Natascia focuses on implementation and maintenance of monitoring systems.

She has good knowledge of all ITIL-related processes and how they should be integrated into monitoring solutions. She focuses on System Center Operations Manager and Orchestrator. She posts blogs at <https://systemcentertips.wordpress.com/>.

www.PacktPub.com

Support files, eBooks, discount offers, and more

For support files and downloads related to your book, please visit www.PacktPub.com.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<https://www2.packtpub.com/books/subscription/packtlib>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and read Packt's entire library of books.

Why subscribe?

- ▶ Fully searchable across every book published by Packt
- ▶ Copy and paste, print, and bookmark content
- ▶ On demand and accessible via a web browser

Free access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view 9 entirely free books. Simply use your login credentials for immediate access.

Instant updates on new Packt books

Get notified! Find out when new books are published by following [@PacktEnterprise](https://twitter.com/PacktEnterprise) on Twitter or the *Packt Enterprise* Facebook page.

Table of Contents

Preface	v
Chapter 1: Understanding the Goals of Reporting	1
Introduction	1
Understanding the goals of reporting	2
Planning and optimizing dependent data inputs	4
Planning report outputs	7
Understanding dashboards and their dependencies	9
Targeting reports at decision stakeholders	11
Documenting report designs	14
Chapter 2: Planning System Center Report Design	17
Introduction	17
Understanding the reporting schemas of System Center components	18
Understanding how SQL Reporting Services interfaces with System Center	27
Building SQL queries the easy way	31
Chapter 3: Unpacking System Center Report Building Tools	37
Introduction	37
Planning the Report Builder installation	38
Installing Report Builder	40
Navigating through the Report Builder interface	45
Organizing the reporting environment and delegating access to reports	52
Creating data sources	60
Creating datasets and a basic report in Report Builder	66
Preparing for report subscriptions	73

Chapter 4: Creating Reports for System Center Configuration Manager	75
Introduction	75
Exploring the Configuration Manager database schema	76
Delegating access to out-of-the-box Configuration Manager reports	83
Preparing the SCCM reporting environment for reporting	88
Creating Configuration Manager custom reports	92
Applying role-based security to custom reports	108
Chapter 5: Creating Reports for SCOM and SCVMM	117
Introduction	117
Using out-of-the-box Operations Manager reports	118
Understanding the OperationsManagerDW schema	123
Preparing your environment to author reports	129
Creating alert reports	131
Creating event reports	138
Creating performance reports	144
Creating availability reports	150
Using Virtual Machine Manager reports	158
Chapter 6: Creating Reports for System Center Data Protection Manager	163
Introduction	163
Preparing the DPM reporting environment	164
Preparing a dataset query for a DPM agent status report	168
Creating a DPM agent version and a disk space report	176
Creating a backup status report with a community template	186
Chapter 7: Creating Reports for System Center Service Manager and Orchestrator	191
Introduction	192
Using out-of-the-box Service Manager reports and cubes	193
Understanding the Service Manager Data Warehouse data mart	199
Creating Work Item and Configuration Item reports	204
Creating reports using OLAP cubes	211
Creating reports using the operational database	215
Accessing data using Microsoft Excel	224
Extending the Service Manager Data Warehouse	227
Creating Orchestrator runbook reports	232

Chapter 8: Creating System Center Advanced Reports	239
Introduction	239
Creating dashboards from basic reports	240
Working with drillthrough reports	252
Working with shared datasets and parameters	258
Analyzing Chargeback reports using System Center 2012 R2 data	265
Chapter 9: Using Power BI to Analyze and Visualize System Center Data	277
Introduction	277
Configuring Microsoft Excel for System Center data analysis	278
Connecting to System Center data sources with Power Query	283
Creating the data model with PowerPivot	298
Visualizing the analysis with Power View and Power Maps	305
Using Microsoft Cloud services to share and visualize System Center data	309
Appendix: Useful Websites, Chapter Code, and Community Resources	313
Authors' community blogs	313
Useful community blogs	314
Websites for SQL and reporting	314
Online wikis and curations	315
Index	325

Preface

The System Center product provides you with the best IT systems management and reporting capabilities. The various components of this product continue to provide great value in all areas of management and monitoring. Each of the components has one or more data storing repositories. These data repositories hold valuable strategic information that can assist an organization in transforming its IT business unit into a strategic enabler.

The database (repository) technology of the System Center product is Microsoft SQL Server. There is a vast array of information on the Internet on how to query and report on Microsoft SQL databases. The skill set required for the level of querying and reporting on these databases is normally aligned to Database Administrators (DBAs) and Business Intelligence (BI) specialists. The DBAs and BI specialists use various tools to create business-valued reports based on the data stored in these Microsoft SQL databases.

The challenge DBAs and BI experts face with System Center reporting is a lack of insight into how the data is structured, and more importantly, a lack of deeper understanding of the products responsible for the input data. Conversely, in the majority of cases, the System Center product's Subject Matter Experts (SMEs) do not have the same level of competency as the DBAs and BI experts in creating and enhancing reports.

The aim of this book is to enhance the competency of System Center SMEs in the areas of reporting. There are a number of books on querying Microsoft SQL databases and generating business-value reports. However, those books do not focus on System Center. Also, all the books on System Center's components dedicate a reporting chapter to that specific component of the book's focus.

Microsoft System Center Reporting Cookbook, provides a one-stop shop for how to plan, create, and manage reports for all the components of the System Center product.

This book is written in the Packt Publishing style, which provides you with independent, task-oriented steps to achieve specific reporting objectives. We recommend that you read the first three chapters as primers for subsequent chapters. This book may be read in the order of your interest, but wherever relevant, we've referred to the dependent recipes in other chapters.

What this book covers

Chapter 1, Understanding the Goals of Reporting, discusses the building blocks of your reporting framework and how to plan for business-valued reports.

Chapter 2, Planning System Center Report Design, covers the internals of System Center reporting database structures, which you must understand in order to create credible and powerful reports. This chapter also includes an introductory recipe to SQL queries.

Chapter 3, Unpacking System Center Report Building Tools, discusses the common report-creating tools available for System Center administrators and Business Intelligence power users.

Chapter 4, Creating Reports for System Center Configuration Manager, provides tasks for System Center administrators, who can create System Center Configuration Manager reports using the Report Builder tool.

Chapter 5, Creating Reports for SCOM and SCVMM, demonstrates how to create System Center Operations and Data Protection Manager reports, with data available from the Operations Manager databases.

Chapter 6, Creating Reports for System Center Data Protection Manager, shows you how to create data-driven reports for System Center Data Protection Manager.

Chapter 7, Creating Reports for System Center Service Manager and Orchestrator, illustrates how to create service management and automation activity reports using data from System Center Service Manager and Orchestrator.

Chapter 8, Creating System Center Advanced Reports, shows you recipes that build on and enhance the reports from previous recipes in this book. These recipes delve into advanced reporting techniques and combined data source reporting.

Chapter 9, Using Power BI to Analyze and Visualize System Center Data, introduces the Power Business Intelligence (Power BI) options available for you from Microsoft. These recipes provide steps for analyzing and visualizing the System Center data using the Microsoft Excel Power BI add-ons, and introduce the cloud Power BI versions available for you.

Appendix, Useful Websites, Chapter Code, and Community Contributions, lists some of the sites that provide ready-made solutions and extensive, real-world, and dynamic content on reporting. Microsoft SQL Server-based reporting, similar to most Microsoft product areas, has an extended solutions partner community. There is an extensive and active support base on the Web. This appendix also provides you with the SQL and XML code referred to in the relevant chapters.

What you need for this book

In order to complete all the recipes in this book, you will need access to environments configured with the relevant System Center component (or components). Here is a list of technologies the recipes depend on and their relevant versions used for this book:

- ▶ Microsoft Active Directory (Windows Server 2008 R2 and above)
- ▶ System Center 2012 Configuration Manager SP1/R2
- ▶ System Center 2012 Operations Manager SP1/R2
- ▶ System Center 2012 Virtual Machine Manager SP1/R2
- ▶ System Center 2012 Service Manager SP1/R2
- ▶ System Center 2012 Orchestrator SP1/R2
- ▶ System Center 2012 Data Protection Manager SP1/R2
- ▶ Microsoft Report Builder 3.0

The required software and deployment guides of System Center 2012 can be found on the official Microsoft website, at <http://www.microsoft.com/en-us/server-cloud/system-center/default.aspx>.

We recommend using the online Microsoft resource due to the frequency of updates to the products' supported requirements. Also note that the dynamic nature of the Internet may require you to search for updated links listed in this book.

Who this book is for

This book is for IT professionals who are responsible for producing reports using the data from System Center components. Basic knowledge of Microsoft System Center technologies is assumed.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "You have the option to configure two additional databases known as `OMDataMart` and `CMDataMart`."

A block of code is set as follows:

```
SELECT fnRS.Name0 As [Computer Name],
       fnC.Name As [Collection Name]
FROM   fn_rbac_FullCollectionMembership (@UserSIDs) fnFCM
JOIN   fn_rbac_R_System (@UserSIDs) fnRS ON fnFCM.ResourceID
       = fnRS.ResourceID
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "Under **Report Data**, right-click on the dataset called **List_Collections**."



Warnings or important notes appear in a box like this.



Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to feedback@packtpub.com, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at <http://www.PacktPub.com>. If you purchased this book elsewhere, you can visit <http://www.PacktPub.com/support> and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/submit-errata>, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from <http://www.packtpub.com/support>.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at questions@packtpub.com if you are having a problem with any aspect of the book, and we will do our best to address it.

1

Understanding the Goals of Reporting

In this chapter, we will cover the following recipes:

- ▶ Understanding the goals of reporting
- ▶ Planning and optimizing dependent data inputs
- ▶ Planning report outputs
- ▶ Understanding dashboards and their dependencies
- ▶ Targeting reports at decision stakeholders
- ▶ Documenting Report Designs

Introduction

This chapter and the whole book builds on two guiding principles:

"You can't manage what you don't measure" and "there is no substitute for knowledge".

Reporting allows you to make strategic decisions based on the knowledge acquired from credible data sources. This chapter focuses on what steps you must take to identify and define what business specific objectives an organization aims to achieve with the output of reports. The report categories discussed in the book uses the data from the System Center product.

Understanding the goals of reporting

This recipe discusses the drivers of organizational reporting and the general requirements on how to plan for business valued reports.

Getting ready

To prepare for this recipe you need to be ready to make a difference with all the rich data available to you in the area of reporting. This may require a mindset change; be prepared.

How to do it...

The key to successfully identifying what needs to be reported is a clear understanding of what you or the report requestor is trying to measure and why.

Reporting is driven by a number of organizational needs, which may fall into one or more of these sample categories:

- ▶ Information to support a business case
- ▶ Audit and compliance driven request
- ▶ Budget planning and forecasting
- ▶ Current operational service level

These categories are examples of the business needs which you must understand. Understanding the business needs of the report increases the value of the report. For example, let us expand on and map the preceding business scenarios to the System Center Product using the following table:

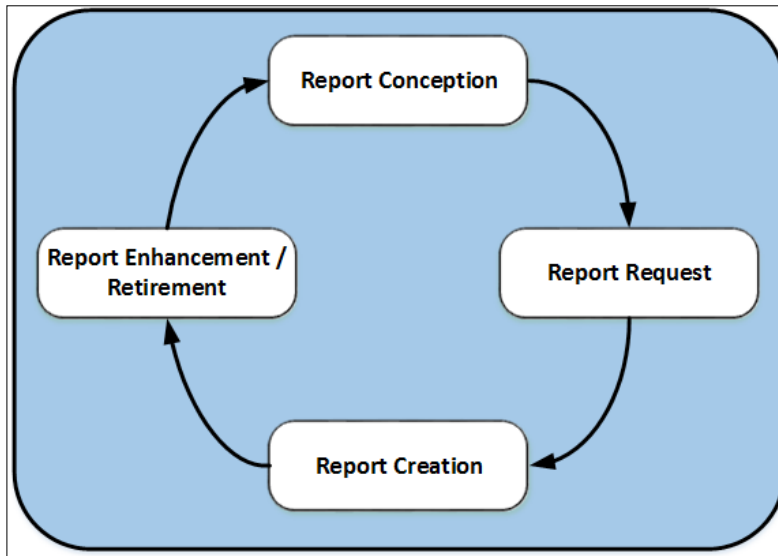
Business/organizational objective	Objective details	System Center Product
Information to support a business case	Provide a count of computers out of warranty to justify the request to buy additional computers.	System Center Configuration Manager
Audit and compliance driven request	<ul style="list-style-type: none">▶ Provide the security compliance state of all windows servers.▶ Provide a list of attempted security breaches by month.	<ul style="list-style-type: none">▶ System Center Configuration Manager▶ System Center Operations Manager

Business/organizational objective	Objective details	System Center Product
Budget planning and forecasting	How much storage should we plan to invest in next year's budget based on the last 3 years' usage data?	System Center Operations Manager
Operational Service Level	How many incidents were resolved without second tier escalation?	System Center Service Manager

In a majority of cases for System Center administrators, the requestor does not provide the business objective. Use the preceding table as an example to guide your understanding of a report request.

How it works...

Reporting is a continual life cycle that begins with a request for information and should ultimately satisfy a real need. The typical life cycle of a request is illustrated in the following figure:



The life cycle stages are:

- ▶ Report conception
- ▶ Report request
- ▶ Report creation
- ▶ Report enhancement/retirement

The recipe focuses on the report conception stage. This stage is the most important stage of the life cycle. This is due to the fact that a report with a clear business objective will deliver the following:

- ▶ **Focused activities:** A report that does have a clear objective will reduce the risk of wasted effort usually associated with unclear requirements.
- ▶ **Direct or indirect business benefit:** The reports you create, for example using System Center data, ultimately should benefit the business.

An additional benefit to this stage of report planning is knowing when a report is no longer required. This would reduce the need to manage and support a report that has no value or use.

Planning and optimizing dependent data inputs

A report is only as good as the accuracy of its data source. A data source is populated and updated by an input channel. This recipe discusses and provides steps for planning for the inputs your report data source(s) depends on.

Getting ready

Review the *Understanding the goals of reporting* recipe as a primer for this recipe.

How to do it...

The inputs of reports depend on the type of output you intend to produce and the definition of the accepted fields in the data source. An example is a report that would provide a total count of computers in a System Center Configuration Manager environment. This report will require an input field which stores a numeric value for computers in the database.

Here are the recommended steps you must take to prepare and optimize the data inputs for a report:

1. Identify the data source or sources.
2. Document the source data type properties.
3. Document the process used to populate the data sources (manual or automated process).
4. Agree the authoritative source if there is more than one source for the same data.
5. Identify and document relationship between sources.
6. Document steps 1 to 5.

The following table provides a practical example of the steps for a report on the total count of computers by the Windows operating system. Workgroup computers and computers not in the Active Directory domain are out of scope of this report request.

Report input type	Details	Notes
Data source	Asset Database	Populated manually by the purchase order team
Data source	Active Directory	Automatically populated. Orchestrator runbook performs a scheduled clean-up of disabled objects
Data source	System Center Configuration Manager	Requires an agent and currently not used to manage servers
Authoritative source	Active Directory	Based on the report scope
Data source relationship	Microsoft System Center Configuration Manager is configured to discover all systems in the Active directory domain	Alternative source for the report using the All systems collection

Plan to document the specific fields you need from the authoritative data source. For example, use a table similar to the following.

Required data	Description
Computer name	The Fully Qualified domain name of the computer
Operating system	Friendly operating system name
Operating system environment	Server or workstation
Date created in data source	Date the computer joined the domain
Last logon date	Date the computer last updated the attributes in Active Directory

The steps provided discusses an example of identifying input sources and the fields you plan to use in a requested report.

Optimizing Report Inputs

Once the required data for your reports have been identified and documented, you must test for validity and consistency. Data sources which are populated by automated processes tend to be less prone to consistency errors. Conversely data sources based on manual entry are prone to errors (for example, correct spelling when typing text into forms used to populate the data source). Here are typical recommended practices for improving consistency in manual and automated system populated data sources:

- ▶ Automated (for example, agent based):
 1. Implement agent health check and remediation.
 2. Include last agent update information in reports.
- ▶ Manual entry:
 1. Avoid free text fields, except description or notes.
 2. Use a list picker.
 3. Implement mandatory constraints on required fields (for example, a request for e-mail address should only accept the right format for e-mail addresses.

How it works...

The reports you create and manage are only as accurate as the original data source. There may be one or more sources available for a report. The process discussed in this recipe provides steps on how to narrow down the list of requirements. The list must include the data source and the specific data fields which contain the data for the proposed report(s). These input fields are populated by manual, automated processes or a combination of both.

The final part of the recipe discussed an example of how to optimize the inputs you select.

These steps will assist in answering one of the typical questions often raised about reports: "Can we trust this information?" The answer, if you have performed these steps will be "Yes, and this is why and how."

See also

- ▶ The *Planning report outputs* recipe

Planning report outputs

The preceding recipe, *Planning and optimizing dependent data inputs*, discussed what you need for a report. This recipe builds on the preceding recipes with a focus on how you plan to view a report (output).

Getting ready

Plan to review the *Understanding the goals of reporting* and *Planning and optimizing dependent data inputs* recipes.

How to do it...

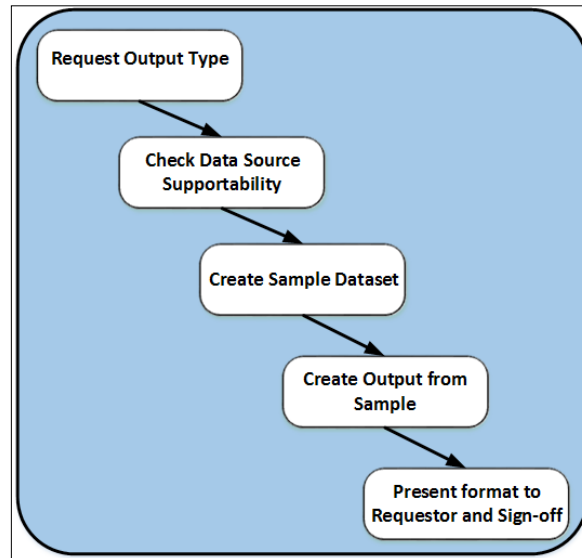
The type of report output depends on the input you query from the target data source(s). Typically, the output type is defined by the requestor of the report and may be in one or more of these formats:

- ▶ List of items (tables)
- ▶ Charts (2D, 3D, and formats supported by the reporting program)
- ▶ Geographic representation
- ▶ Dials and gauges
- ▶ A combination of all the listed formats

Here is an example of the steps you must perform to plan and agree the reporting output (s):

1. Request the target format from the initiator of the report.
2. Check the data source supports the requested output.
3. Create a sample dataset from the source.
4. Create a sample output in the requestor's format(s).
5. Agree a final format or combination of formats with the requestor.

The steps to plan the output of reports are illustrated in the following figure:



These are the basic minimal steps you must perform to plan for outputs.

How it works...

The steps in this recipe are focused on scoping the output of the report. The scope provides you with the following:

- ▶ Ensuring the output is defined before working on a large set of data
- ▶ Validating that the data source can support the requested output
- ▶ Avoids scope creep. The output is agreed and signed off

The objective is to ensure that the request can be satisfied based on what is available and not what is desired. The process also provides an additional benefit of identifying any gaps in data before embarking on the actual report creation.

There's more...

When planning report outputs, you may not always have access to the actual source data. The recommend practice is not to work directly with the original source even if this is possible to avoid negatively impacting the source during the planning stage. In either case, there are other options available to you. One of these options is using a spreadsheet program such as Microsoft Excel.

Mock up using Excel

An approach to testing and validating report outputs is the use of Microsoft Excel. You can create a representation of the input source data including the data type (numbers, text, and formula). The data can either be a sample you create yourself or an extract from the original source of the data.

The added benefit is that the spreadsheet can serve as a part of the portfolio of documentation for the report.

See also

- ▶ The *Understanding dashboards and their dependencies* recipe

Understanding dashboards and their dependencies

The *Planning report outputs* recipe discussed what you need to do to plan for report outputs.

This recipe builds on the preceding recipe highlighting the requirements and dependencies for creating report dashboards. An understanding of dashboards and their dependences is required to ensure that your intended dashboards are fit for purpose and fit for use.

Getting ready

Plan to review the *Understanding the goals of reporting*, *Planning and optimizing dependent inputs*, and *Planning report outputs* recipes.

How to do it...

The term **dashboards** usually denotes the use of a graphical image to represent data. There are typically two types of dashboards:

- ▶ Static
- ▶ Interactive

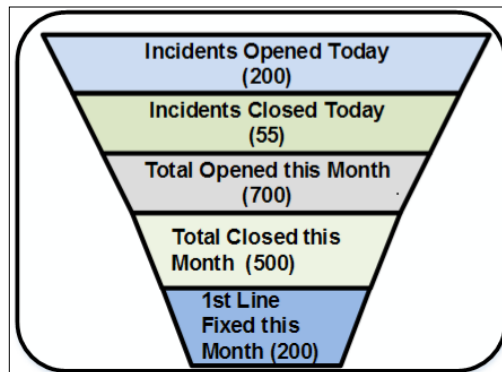
Static dashboards

Static dashboards present an image of the report but does not provide an interactive mechanism. These types of dashboards are similar to charts you create with data in a spreadsheet program (for example, Microsoft Excel).

Interactive dashboards

Interactive dashboards are similar to static dashboards but provide the ability to traverse from the initial dashboard to other dashboards or data tables. This process is commonly described as a drilldown.

In either case the dashboard is a final output of the report creation process. The following figure provides an example of an incident management process dashboard:



The incident management dashboard presented in the preceding figure has the following dashboard sections:

- ▶ **Incidents Opened Today**
- ▶ **Incidents Closed Today**
- ▶ **Total Open this Month**
- ▶ **Total Closed this Month**
- ▶ **1st Level Fixed this Month**

Ask the right questions to get an understanding on how to create this type of dashboard. We'll use the first section as an example: Incidents Opened Today:

- ▶ Is this related to only incidents created today or will it also include incidents reactivated today?
- ▶ How can the difference highlighted in the preceding question be validated?
- ▶ Is today related to working days or every day of the week?
- ▶ What is the smallest and largest display area this dashboard will be viewed on?
- ▶ Who is accountable for the data the dashboard depends on?

Build on the example questions using the scope of the specific report request.

How it works...

Dashboards share similar attributes to films (movies). A film requires a script, the actors, directors, the set, and the equipment. In addition the film requires a rehearsal and a number of processes which are best described by experts in that field. The final output which can be viewed at the cinema or digital media is the films' dashboard.

The focus of this recipe is to highlight the fact that dashboards are only as good as the processes and inputs they depend on. Often the dashboard requestor does not have the same background, or in-depth understanding of the data the report creator requires.

Use the questioning technique in this recipe to bridge the gap between the requestor and the creator. The process will ensure that the expectations of the requestor are aligned with the actual effort required to create the dashboard.

An additional factor is the validity and ownership of the information the dashboard will depend on. A dashboard regardless of type, which presents the wrong information will only lead to chaos. Identifying the owner of the information will provide the report creator with an escalation path as well as a policy enforcement option.

See also

- ▶ *Chapter 8, Creating System Center Advanced Reports*, provides recipes on dashboards using System Center data inputs

Targeting reports at decision stakeholders

This recipe discusses the recipients of reports and the objectives the recipients are typically aiming to achieve.

Getting ready

The prerequisite to this recipe is a basic understanding of the term "stakeholders" as it relates to reports.

How to do it...

The process of targeting stakeholders for reports is based on the following three categories of reporting:

- ▶ Proactive reporting
- ▶ Reactive reporting
- ▶ Highlight reporting

Proactive reporting

This category of reporting is based on highlighting potential issues before it impacts the business. The stakeholders you will target for this type are decision makers with influence. Examples of the stakeholders in this category are:

- ▶ Business unit directors
- ▶ IT security teams
- ▶ Chief executives

Proactive reporting may require forward thinking and advance expenditure, so it is critical to win over the right stakeholders.

Reactive reporting

Reactive reporting usually has many stakeholders. Examples of the stakeholders in this category are:

- ▶ Data center managers
- ▶ IT security teams
- ▶ Asset and problem management teams
- ▶ Business application owners

Examples of reports in this category are security non-compliance, license non-compliance, capacity issues, and hardware audits.

Highlight reporting

This is the "showing off" type of report and the stakeholders are typically trying to sell their value to the business. *If no one knows then you are not doing it.* The stakeholders for these type of reports are those identified in the first two categories but also includes ad hoc requests.

An approach to targeting reports to stakeholders is to first categorize reports. Allocate primary and secondary recipients/or owners for each category. The following table is an example of such a categorization exercise:

Report category	Report(s)	Primary owner	Secondary owner
Capacity management	<ul style="list-style-type: none">▶ Annual disk usage▶ Network bandwidth utilization by location	Data Center Manager	Executive Budgetary board

Report category	Report(s)	Primary owner	Secondary owner
Asset management	<ul style="list-style-type: none"> ▶ List of all client operating systems in use ▶ Count of users issued out of hours door access 	Procurement Manager	Facilities and Clients Services
Monitoring and availability	The Line of Business (LOB) application outages this month	LOB department manager	IT operations director
Security and policy compliance	List of clients with required security updates older than 60 days	Head of global security	Client Services manager
Incident and problem management	<ul style="list-style-type: none"> ▶ Number of incidents resolved without escalation ▶ List of problem records by date 	Service Delivery manager	Service Desk team lead
Change and release management	Percentage of successful changes completed in the allocated change window	Change Manager	Change advisory board

How it works...

The process of targeting reports may be proactive or reactive. The recipe provides you with the process you can follow to ensure the majority of your reports fall in the proactive category. The reactive category is unavoidable but the categorization and ownership of reports is invaluable even in this category.

Plan to adopt and adapt the information provided to your specific requirements and environment.

Documenting report designs

This recipe is a summation of all the recipes in this chapter. This recipe completes the loop with a discussion and example on documenting a report design.

Getting ready

You must plan to review and perform the steps in the preceding recipes in this chapter:

- ▶ Understanding goals of reporting
- ▶ Planning and optimizing dependent data inputs
- ▶ Planning report outputs and their dependencies
- ▶ Targeting reports at decision stakeholders

How to do it...

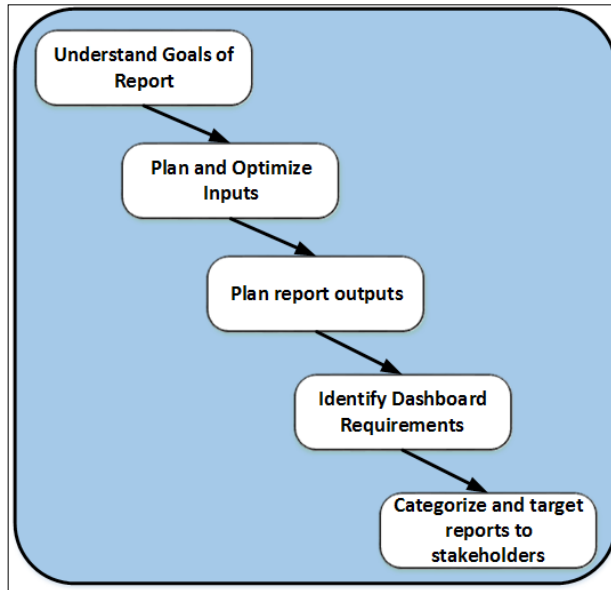
Here is the scenario used in this recipe to provide the steps for documenting a report design.

Scenario: A service request has been received by the service desk team to create a report which shows how many incidents are raised each month and each quarter.

Here are the steps you can follow to document the requirements of this scenario:

1. Create a document to capture the requirements using your favorite editor.
2. Plan to have the following sections in the document:
 - Report Owner
 - Data source and owner
 - Required output and format
 - Frequency of report
 - Review timeline for the report

3. Follow the steps discussed in the preceding recipes in this chapter. Include the tables in the document or create a spreadsheet where appropriate. The following figure is an illustration of the areas to focus on based on the recipes in this chapter:



How it works...

The documentation process approach discussed serves as a guideline and a high-level template. Reporting requirements tend to change as the business or environment evolves. There are parts of the reporting process that tend to be less prone to change (for example, owners of the report and the data sources).

The documentation process should be part of the change management process for the report. Create a baseline document for new reports and continually update the document as the requirements are updated.

One of the sections with high benefit is the review timeline for the document. Set a review date with a reminder for the document.

Documenting the report design saves time and keeps the effort focused.

See also

- ▶ *Appendix A* lists additional resources you can use to enhance your knowledge on understanding the goals of reporting

2

Planning System Center Report Design

In this chapter, we will cover the following recipes:

- ▶ Understanding the reporting schemas of System Center components
- ▶ Understanding how SQL Reporting Services interface with System Center
- ▶ Building SQL queries the easy way

Introduction

The recipes in *Chapter 1, Understanding the Goals of Reporting*, discuss the objectives of reports and the prerequisite steps you must take before delving into the creation process. This chapter builds on the concepts in *Chapter 1, Understanding the Goals of Reporting*, covering the internals of System Center 2012 R2 reporting database structures, which you must understand to create credible and powerful reports. The chapter also includes an introduction recipe to SQL queries. This information is applicable to the previous versions of the System Center components that now make up the unified product.

Understanding the reporting schemas of System Center components

The reporting schema of the System Center product (previously known as the System Center suite) is specific to each component. The components of the System Center product are listed in the following table:

System Center component	Description
Configuration Manager	This is configuration life cycle management. It is primarily targeted at client management; however, this is not a technical limitation, and can be used and is also used to manage servers. This component provides configuration management capabilities, which include but are not limited to deploying operating systems, performing hardware and software inventory, and performing application life cycle management.
Data Protection Manager	This component delivers the capabilities to provide continual protection (backup and recovery) services for servers and clients.
Orchestrator	This is the automation component of the product. It is a platform to connect the different vendor products in a heterogeneous environment in order to provide task automation and business-process automation.
Operations Manager	This component provides data center and client monitoring. Monitoring and remediation is performed at the component and deep application levels.
Service Manager	This provides IT service management capabilities. The capabilities are aligned with the Information Technology Infrastructure Library (ITIL) and the Microsoft Operations Framework (MOF) .
Virtual Machine Manager	This is the component to manage virtualization. The capabilities span the management of private, public, and hybrid clouds.

This recipe discusses the reporting capabilities of each of these components.

Getting ready

You must have a fully deployed configuration of one or more of the System Center product components. Your deployment must include the reporting option provided for the specific component.

How to do it...

The reporting capability for all the System Center components is rooted in their use of Microsoft SQL databases. The reporting databases for each of the components are listed in the following table:

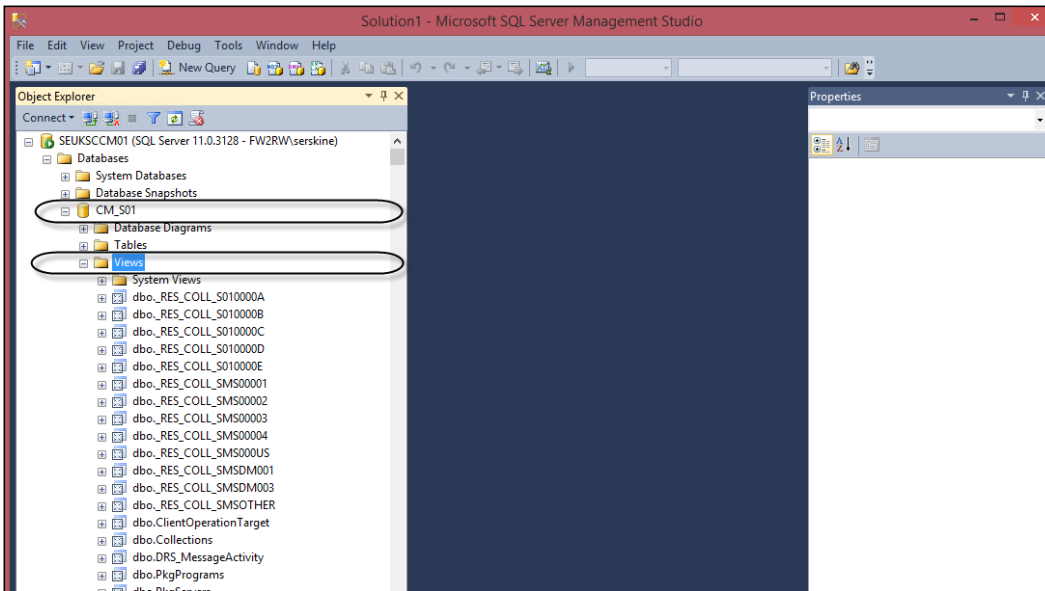
System Center component	Default installation reporting database	Additional information
Configuration Manager	CM_<Site Code>	There is one database for each Configuration Manager site.
Data Protection Manager	DPMDB_<DPM Server Name>	This is the default database for the DPM server. Additional information is written to the Operations Manager database if this optional integration is configured.
Orchestrator	Orchestrator	This is the default name when you install Orchestrator.
Operations Manager	OperationsManagerDW	You must install the reporting components to create and populate this database.
Service Manager	DWDataMart	This is the default reporting database. You have the option to configure two additional databases known as OMDataMart and CMDataMart. Additionally, SQL Analysis Services creates a database called DWASDataBase that uses DWDataMart as a source.
Virtual Machine Manager	VirtualManagerDB	This is the default database for the VMM server. Additional information is written to the Operations Manager database if this optional integration is configured.

Use the steps in the following sections to view the schema of the reporting database of each of the System Center components.

Configuration Manager

Use the following steps:

1. Identify the database server and instance of the Configuration Manager site.
2. Use **Microsoft SQL Server Management Studio (MSSMS)** to connect to the database server. You must connect with a user account with the appropriate permission to view the Configuration Manager database.
3. Navigate to **Databases | CM_<site code> | Views**, as shown in the following screenshot:

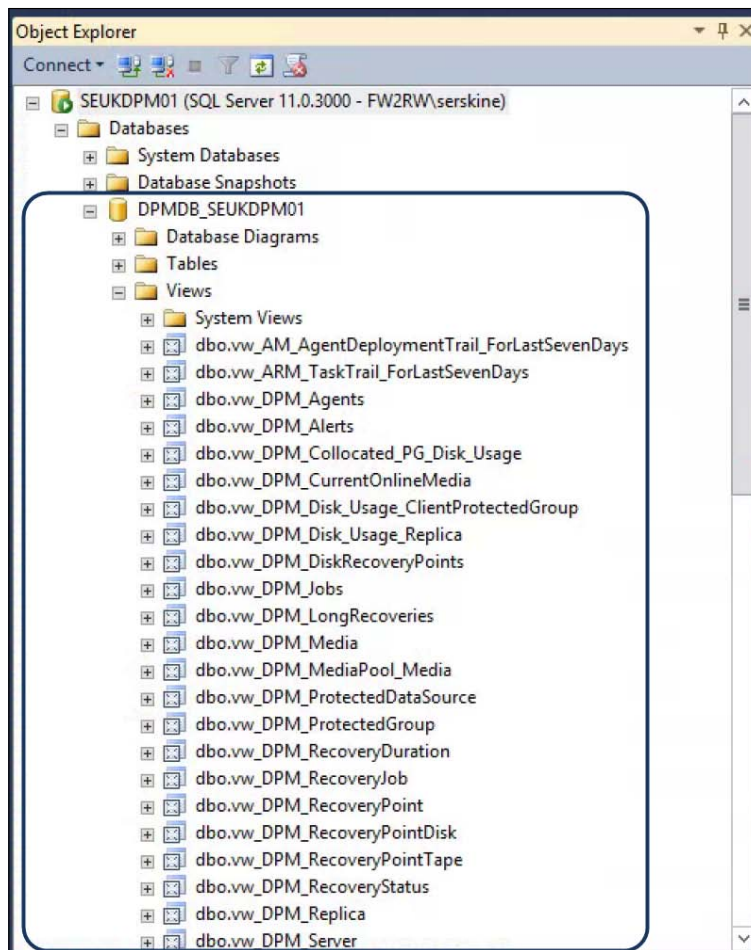


4. The views listed form the reporting schema for the System Center Configuration Manager component. Note that not all the views are listed in the screenshot.

Data Protection Manager

Use the following steps:

1. Identify the database server and SQL instance of the Data Protection Manager environment.
2. Use MSSMS to connect to the database server. You must connect with a user account with the appropriate permission to view the Configuration Manager database.
3. Navigate to **Databases | DPMDDB_<Server Name> | Views**, as shown in the following screenshot:



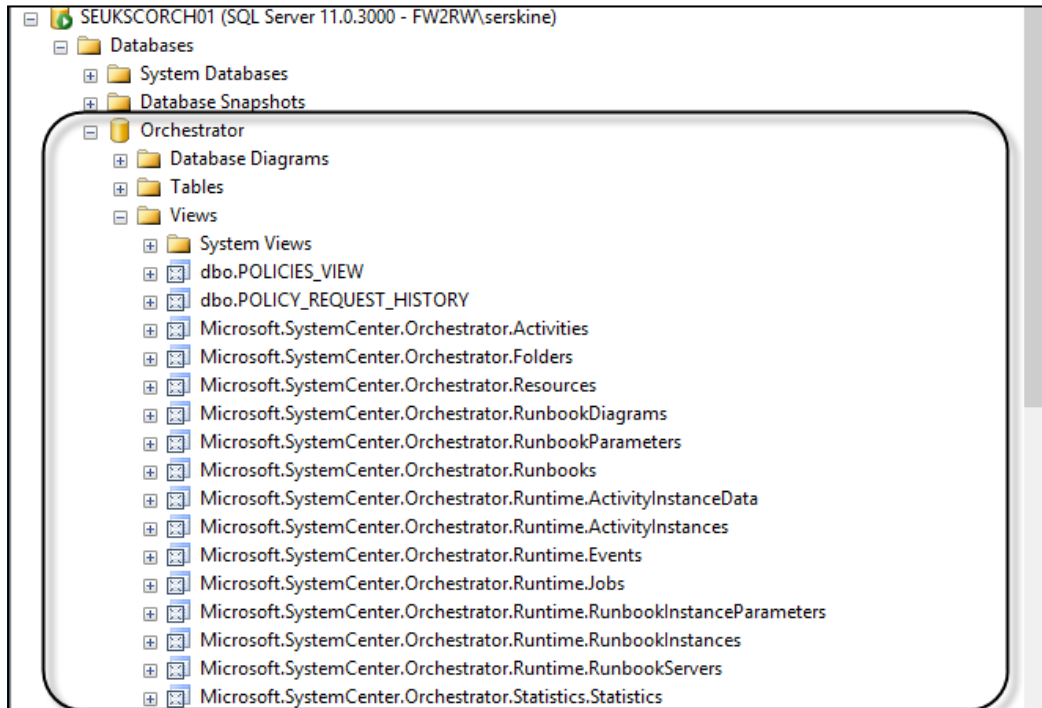
4. The views listed form the reporting schema for the System Center Data Protection Manager component. Note that not all the views are shown in the screenshot.

Orchestrator

Use the following steps:

1. Identify the database server and instance of the Orchestrator instance server.
2. Use MSSMS to connect to the database server. You must connect with a user account with the appropriate permission to view the Orchestrator database.

3. Navigate to **Databases | Orchestrator | Views**, as shown in the following screenshot:



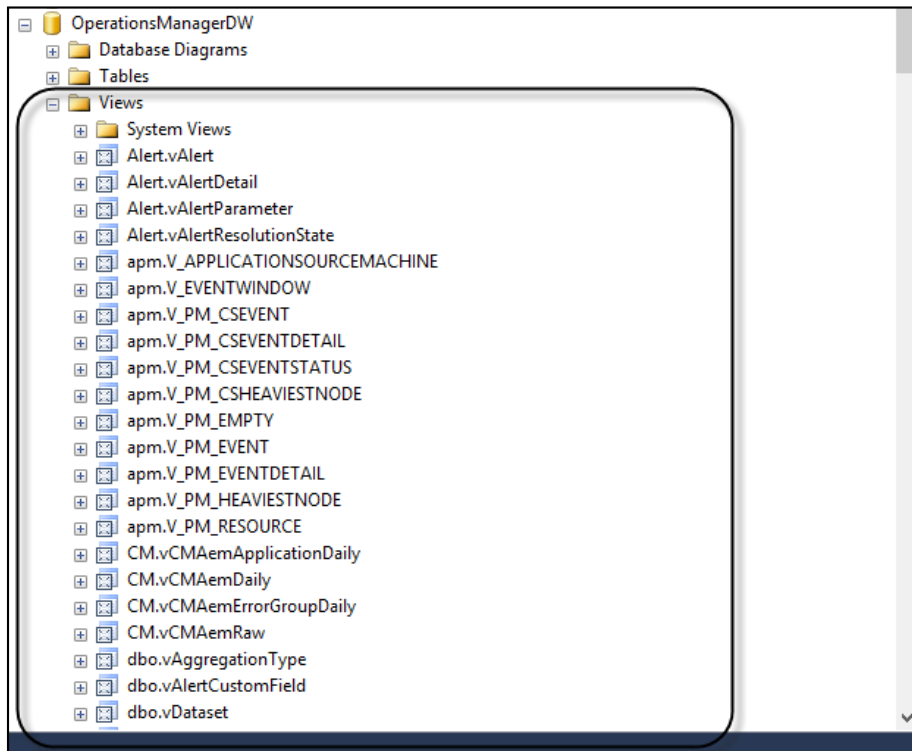
4. The views listed form the reporting schema for the System Center Orchestrator component.

Operations Manager

Use the following steps:

1. Identify the database server and instance of the Operations Manager management group.
2. Use MSSMS to connect to the database server. You must connect with a user account with the appropriate permission to view the Operations Manager data warehouse reporting database.

3. Navigate to **Databases | OperationsManagerDW | Views**, as shown in the following screenshot:



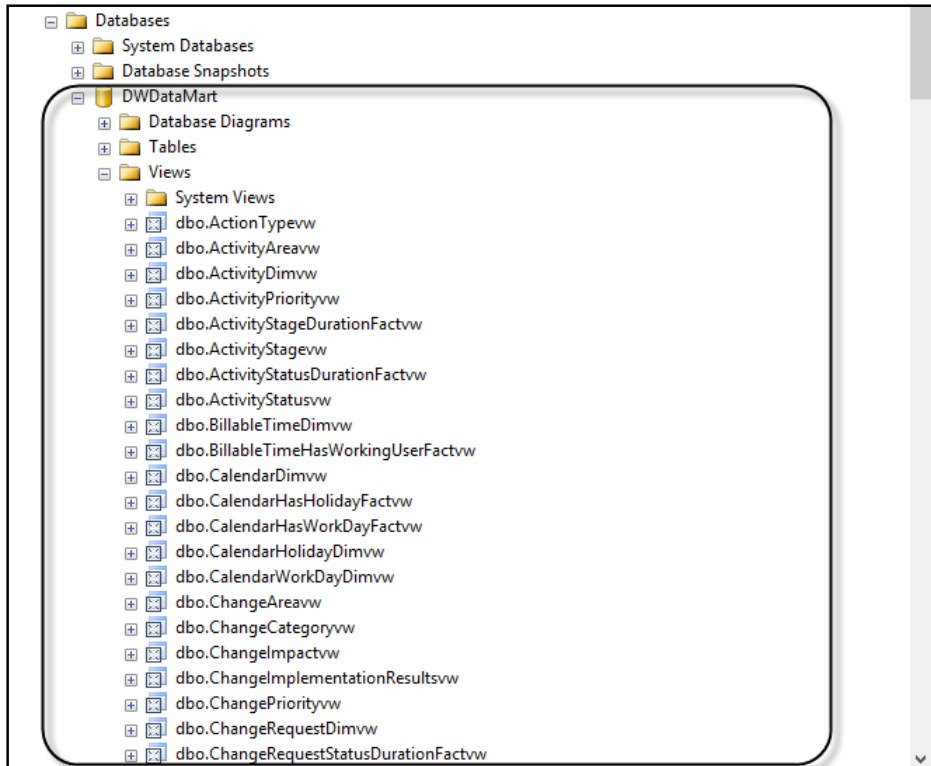
4. The views listed form the reporting schema for the System Center Operations Manager component. Note that not all the views are listed in the screenshot.

Service Manager

Use the following steps:

1. Identify the database server and instance of the Service Manager data warehouse management group.
2. Use MSSMS to connect to the database server. You must connect with a user account with the appropriate permission to view the Service Manager data warehouse database.

3. Navigate to **Databases** | **DWDDataMart** | **Views**, as shown in the following screenshot:

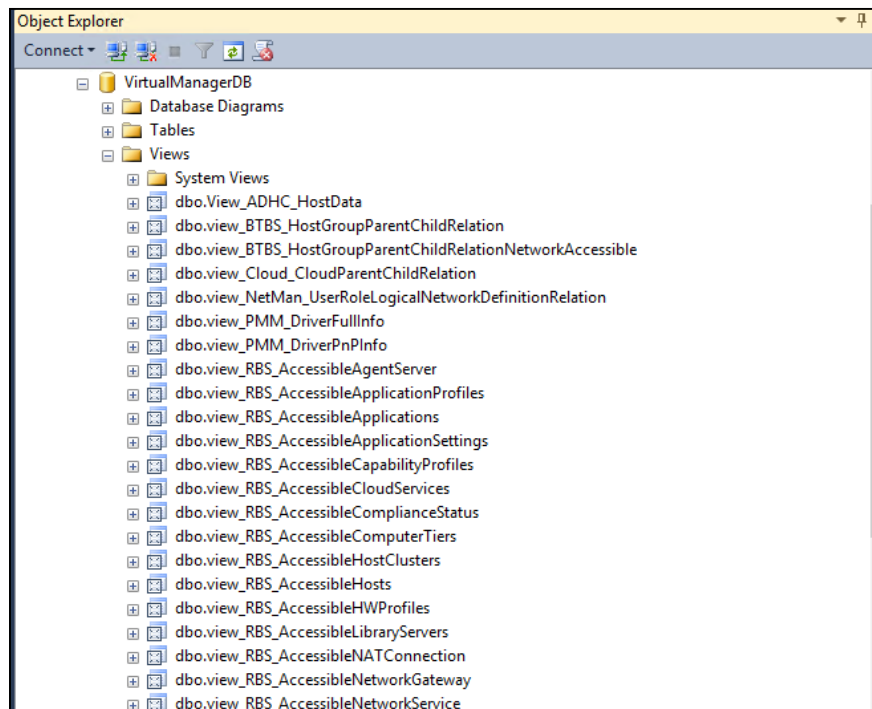


4. The views listed form the reporting schema for the System Center Configuration Manager component. Note that not all the views are listed in the screenshot.

Virtual Machine Manager

Perform the following steps:

1. Identify the database server and instance of the Virtual Machine Manager server.
2. Use MSSMS to connect to the database server. You must connect with a user account with the appropriate permission to view the Virtual Machine Manager database.
3. Go to **Databases** | **VirtualManagerDB** | **Views**, as shown in the following screenshot:



4. The views listed form the reporting schema for the System Center Configuration Manager component. Note that not all the views are listed in the screenshot.

How it works...

The procedure provided is a simplified approach to gain a baseline of what may seem rather complicated if you are new to or have limited experiences with SQL databases. The view for each respective component is a consistent representation of the data that you can retrieve by writing reports. Each view is created from one or more tables in the database. The recommended practice is to target report construction at the views, as Microsoft ensures that these views remain consistent even when the underlying tables change.

An example of how to understand the schema is as follows. Imagine the task of preparing a meal for dinner. The meal will require ingredients and a process to prepare it. Then, you will need to present the output on a plate. The following table provides a comparison of this scenario to the respective schema:

Attributes of the meal	Attributes of the schema
Raw ingredients	Database tables
Packed single or combined ingredients available from a supermarket shelf	SQL Server views that retrieve data from one or a combination of tables
Preparing the meal	Writing SQL queries using the views; use one or a combination (join) views
Presenting the meal on a plate	The report(s) in various formats

In addition to using MSSMS, as described earlier, Microsoft supplies schema information for the components in the online documentation. This information is specific for each product and varies in the depth of the content. The *See also* section of this recipe provides useful links to the available information published for the schemas.

There's more...

It is important to understand the schema for the System Center components, but equally important are the processes that populate the databases. The data population process differs by component, but the results are the same (data is automatically inserted into the respective reporting database). The schema is a map to find the data, but the information available is provided by the agents and processes that transfer the information into the databases.

Components with multiple databases

System Center Service Manager and Operations Manager have a similar architecture. The data is initially written to the operational database and then transferred to the data warehouse. The operational database information is typically what is available to view in the console. The operational information is, however, not the best candidate for reporting, as this is constantly changing. Additionally, performing queries against the operational database can result in performance issues. You may view the schema of these databases using a process similar to the one described earlier, but this is not recommended for reporting purposes.

See also

- ▶ The official online documentation for the schema is updated when Microsoft makes changes to the product, and it should be a point for reference at <http://technet.microsoft.com/en-US/systemcenter/bb980621>.
- ▶ The chapters dedicated to creating reports for each component of the System Center product expands on the information provided in this recipe

Understanding how SQL Reporting Services interfaces with System Center

This recipe provides information on how the Microsoft System Center product leverages SQL Reporting Services to provide a standard reporting interface.

Getting ready

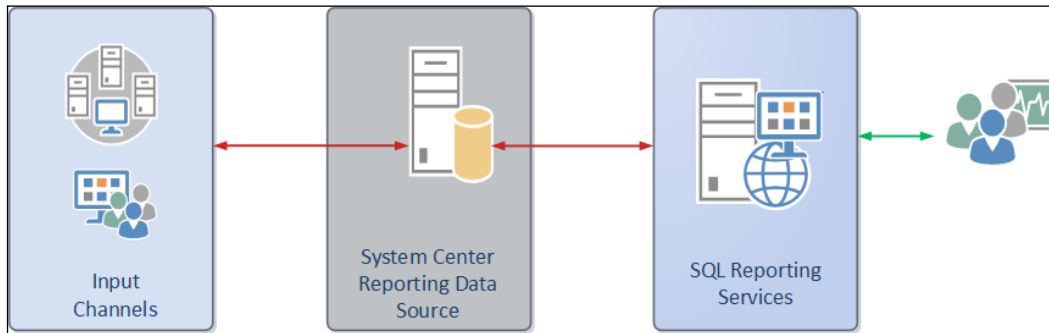
This recipe requires you to have basic understanding of the **Microsoft SQL Reporting Services (SSRS)** component. Additionally, you must have an understanding of one or more of the System Center product components.

How to do it...

The reporting capabilities of the System Center product consists of three logical parts:

- ▶ The input channels for the data
- ▶ The storage component
- ▶ The reporting interface

The first part, **Input channel for the data**, is based on either agents or console entry tasks in the respective System Center component. The second part is implemented with a Microsoft SQL database engine. The third logical part, **The reporting interface**, is provided using Microsoft SQL Reporting Services. The shared logical architecture is illustrated in the following figure:

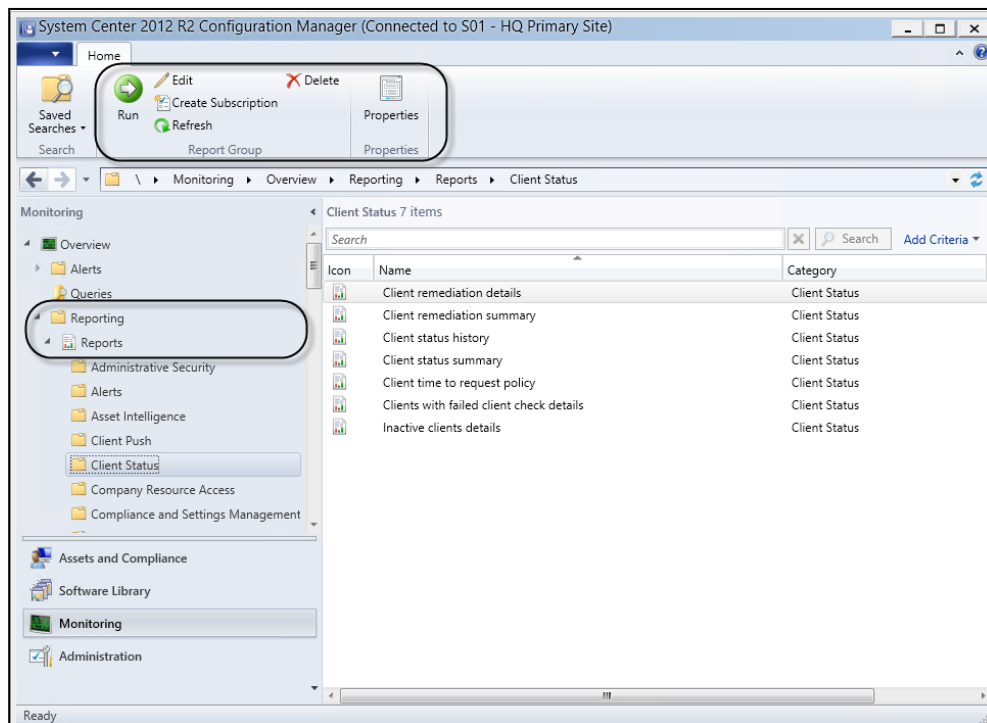


Microsoft SQL Reporting Services provides the following reporting features for the System Center product:

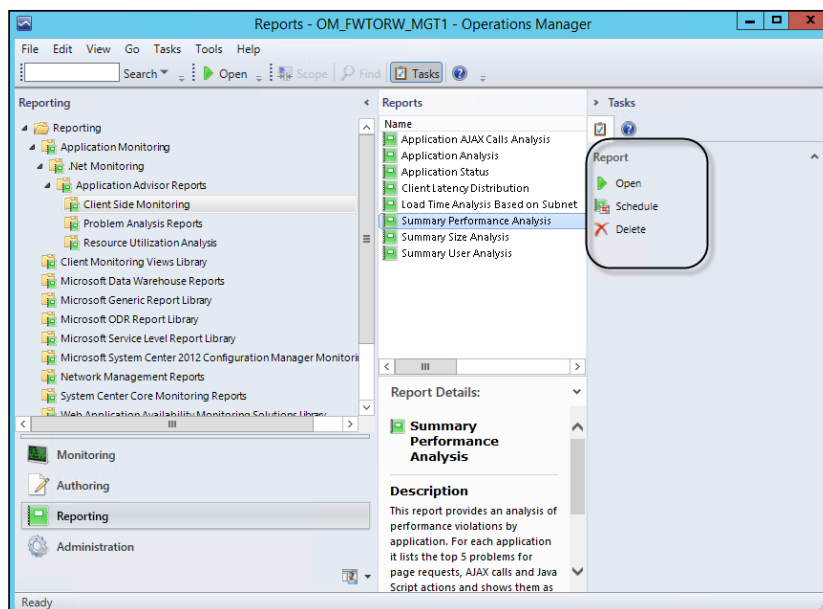
- ▶ Standardized reporting interface
- ▶ Reports are rendered in all SSRS-supported formats
- ▶ Scheduled e-mail subscription to reports
- ▶ Inherited role-based security model

The integration into the System Center product is visible in the consoles of the following components:

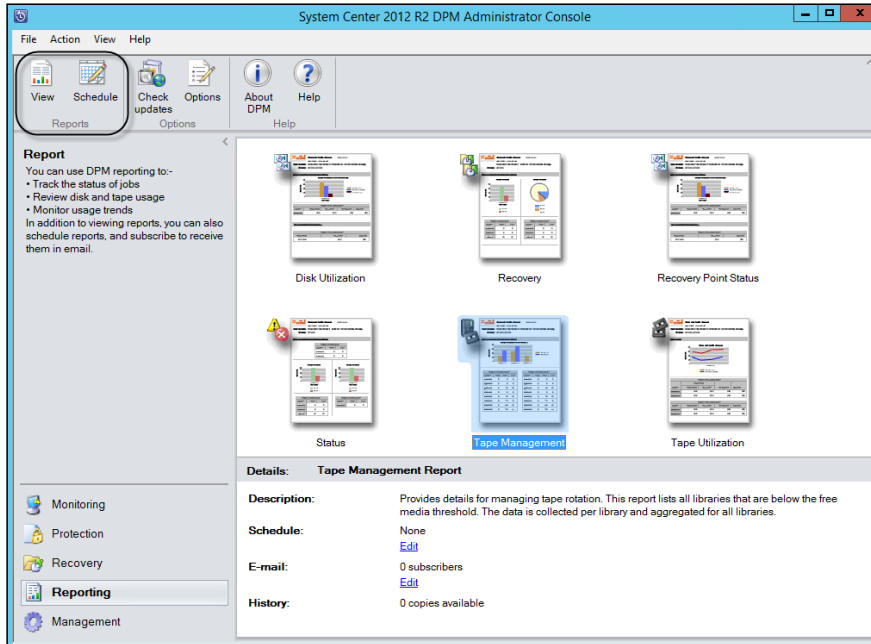
- ▶ **System Center Configuration Manager:** This is shown in the following screenshot:



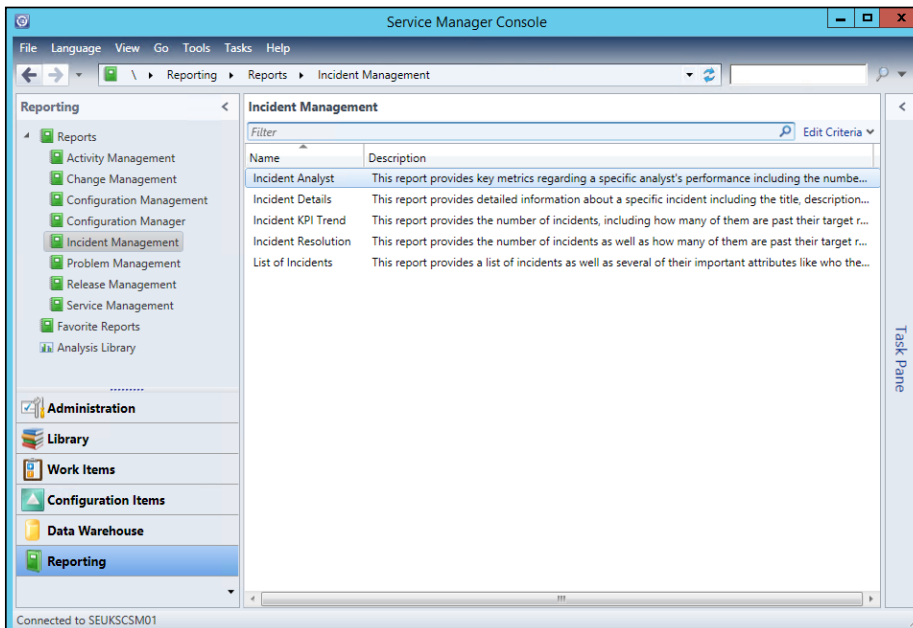
- ▶ **System Center Operations Manager:** This is shown in the following screenshot:



- ▶ **System Center Data Protection Manager:** This is shown in the following screenshot:



- ▶ **System Center Service Manager:** This is shown in the following screenshot:



The Orchestrator and Virtual Machine Manager consoles do not have a direct integration, but you can create and publish reports based on data from both components using a reporting services instance. You are able to report on VMM from Operations Manager and Service Manager when you configure the integration between VMM and Service Manager or VMM and Operations Manager.

How it works...

The use of Microsoft SQL Reporting Services ensures uniformity across the product components and reusable skills in creating reports. This recipe discussed how the features and capabilities of Microsoft SQL Reporting Services are leveraged by the System Center product.

Building SQL queries the easy way

This recipe is a simple primer on how to create very simple queries to retrieve information from the Microsoft SQL database of a System Center component. The recipe uses the System Center Configuration Manager component as the target.

Getting ready

You must have deployed a System Center 2012 Configuration Manager environment. The environment must have the Reporting Services Point role enabled and functional. Additionally, you must have system discovery and inventory enabled.

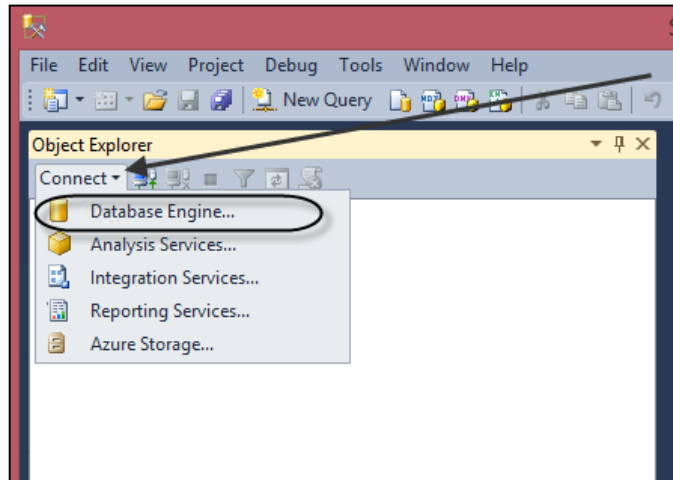
The account you use to perform the steps in this recipe must have access to the SQL Server with read access to the Configuration Manager database. The steps use the Microsoft SQL Server Management Studio (MSSMS) tool.

The recipe's goal is to create a query that returns the NetBIOS name of all the computers and the operating system name. The views in the database that contain this information are listed in the following table:

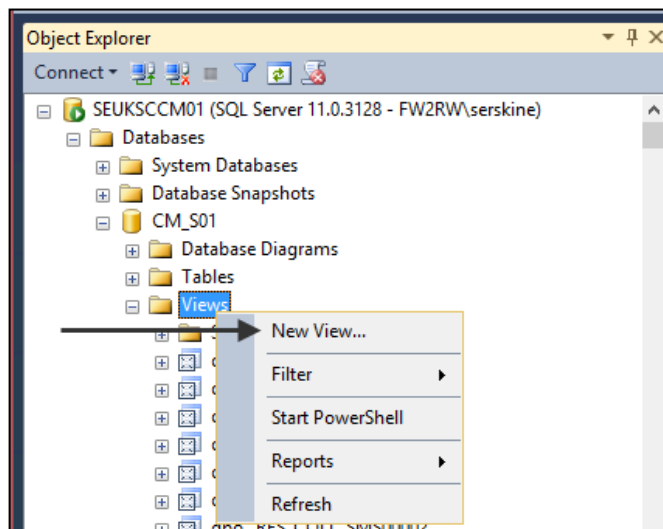
View name	Column	Additional information
V_R_System	Netbios_Name0	This view contains the details for all computers discovered. The view has a unique identifier column called ResourceID.
V_GS_OPERATING_SYSTEM	Caption0	This view contains a list of all operating systems discovered by the hardware inventory process. The name of the computer is not listed in this view, instead the ResourceID identifier is used.

How to do it...

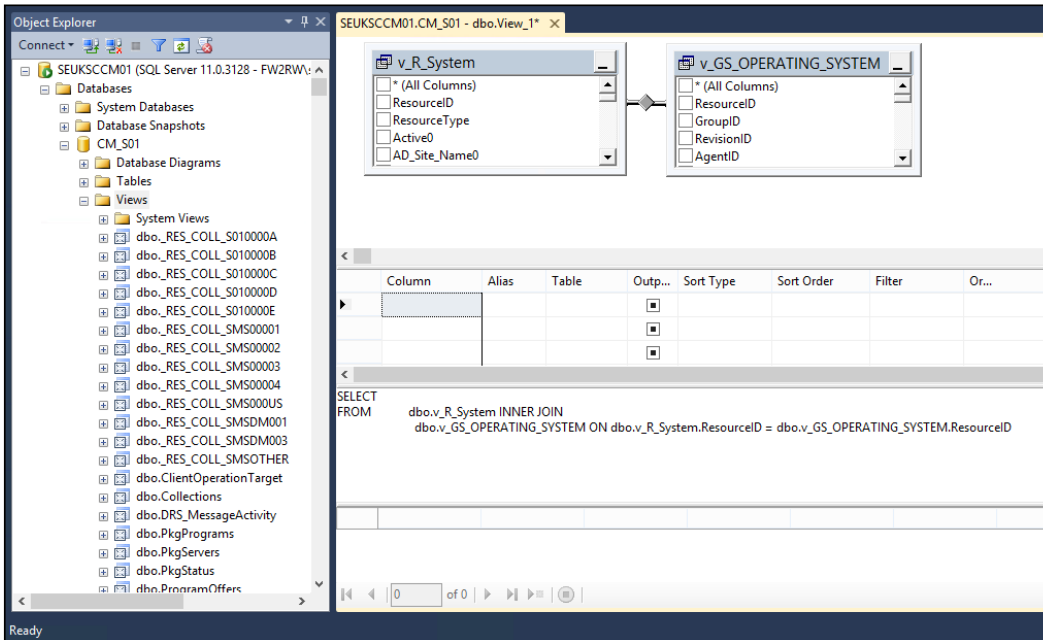
1. Launch MSSMS using an account with access to the Configuration Manager database. Under **Object Explorer**, click on **Connect** and select **Database Engine...**, as shown in the following screenshot:



2. Type the SCCM database server name in the **Server name:** field and click on **Connect**.
3. Expand **Databases**, expand the **CM_<Site Code>** database (in this case **CM_S01**), right-click on **Views**, and click on **New View**, as shown in the following screenshot:

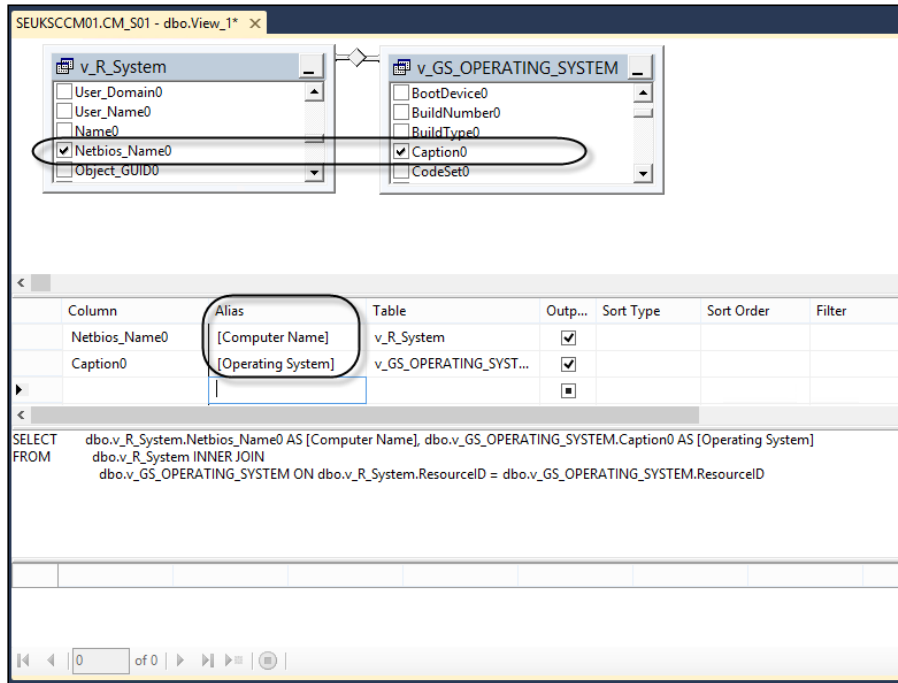


4. Select the **Views** tab in the **Add Table** dialogue box. Scroll and select **v_R_System**, and click on **Add**.
5. Go to **v_GS_OPERATING_SYSTEM** | **Add** | **Close**.
6. Left-click and hold the **ResourceID** field of the **V_R_System** view. Drag the mouse to the **ResourceID** field of the **V_GS_OPERATING_SYSTEM** view. The result of this action should be as illustrated in the following screenshot:

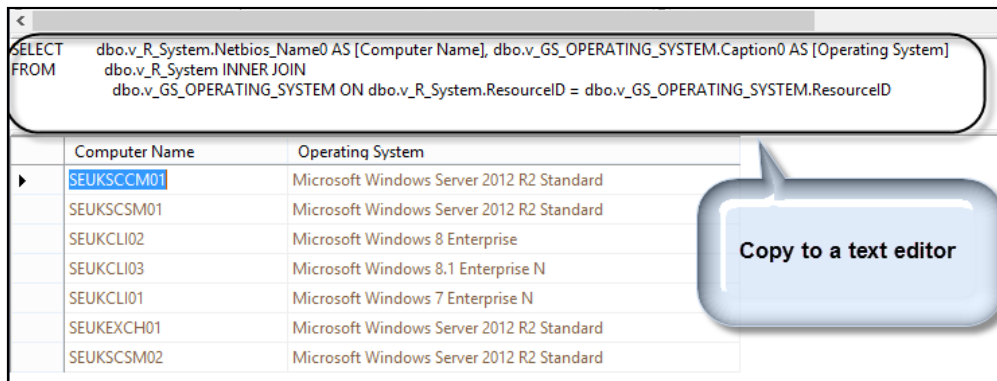


7. Check out the **Netbios_Name0** field in the **v_R_System** view. Check out the **Caption0** field in the **V_GS_OPERATING_SYSTEM** view.

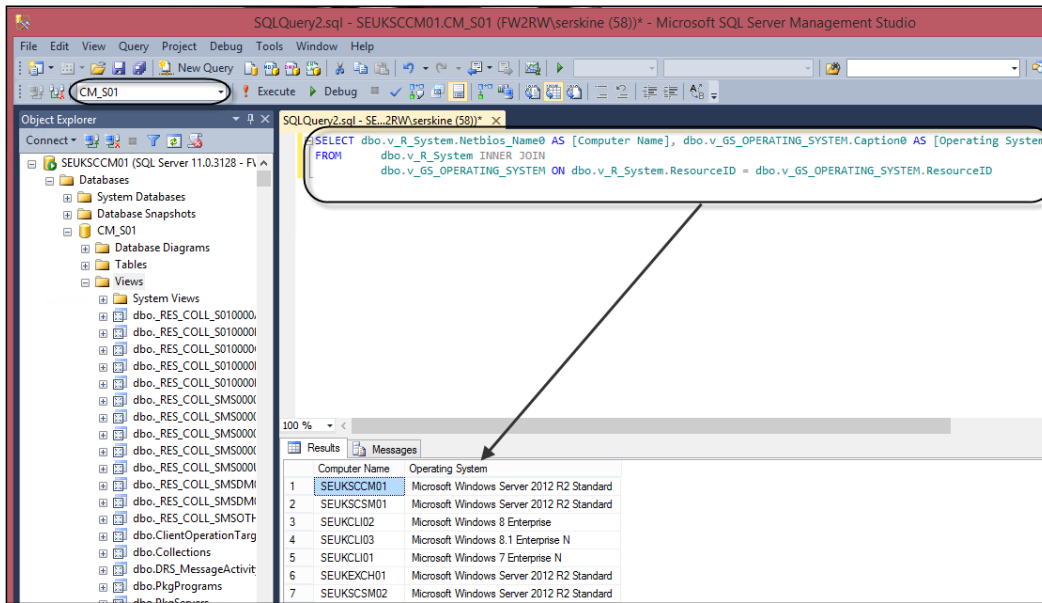
- In the middle pane under **Alias** and next to **Netbios_Name0**, type **Computer Name**. Type **Operating System** in the **Alias** field next to **Caption0** ([and] are automatically wrapped around the text in the **Alias** field), as shown in the following screenshot:



- Test the output of the query by clicking on the **Execute** icon (this is the red exclamation icon).
- The results pane shows the output of the query. Copy the automatically generated query into a text editor (for example, `Notepad.exe`), as shown in the following screenshot:



11. Right-click on the new view tab and click on **Close**. Do not save the view when prompted.
12. In MSSMS, click on **New Query** and paste the query you copied in step 10 into the new query window.
13. Ensure that the SCCM database is selected as indicated and click on the **Execute** icon. The results pane should display the same output as in step 10, as shown in the following screenshot:



14. Use the **Save** icon or **File** menu option to save the query for future use.

How it works...

The steps and options you use to create a view in MSSMS automatically generate the SQL query syntax. This is a very basic approach to perform select statements on the views. You can use this approach to view the contents of the system-generated views of the System Center component. In the steps provided, the view is not saved to prevent us from creating unnecessary views. Only save the views you intend to use for reporting and have them thoroughly tested.

See also

- ▶ The process discussed in this recipe and additional information on the System Center Configuration Manager view can be found at <http://technet.microsoft.com/en-us/library/dn581954.aspx>

3

Unpacking System Center Report Building Tools

In this chapter, we will cover the following recipes:

- ▶ Planning the Report Builder installation
- ▶ Installing Report Builder
- ▶ Navigating through the Report Builder interface
- ▶ Organizing the reporting environment and delegating access to reports
- ▶ Creating data sources
- ▶ Creating datasets and a basic report in Report Builder
- ▶ Preparing for report subscriptions

Introduction

You have the strategy, and you have generated the data. The question is, "What is the best tool available to you to create the reports?" The answer is, "It depends!"

The following are a list of tools that are available from Microsoft:

- ▶ **Microsoft Excel**
- ▶ **Report Builder**
- ▶ **SQL Server Data Tools** (formerly known as **Business Intelligence Development Studio**)
- ▶ **Power BI tools**

The list includes the general and common tools used by report designers. Which one you choose to use typically depends on your level of expertise and the features or limitations of your choice.

This chapter focuses on Report Builder. *Appendix, Useful Websites, Chapter Code, and Community Resources*, includes links to additional resources on reporting tools.

Planning the Report Builder installation

The installation of Microsoft Report Builder is simple. You have two options available:

- ▶ **Standalone:** Install using the downloadable installation file
- ▶ **ClickOnce:** Streamed installation from SQL Reporting Services

You must plan the installation appropriately for your needs. This recipe discusses and provides steps for common planning tasks that you must perform to successfully install Report Builder. This book and chapter focuses on Report Builder 3.0 as the tool used to publish and manage reports on Microsoft SQL Server 2012 Reporting Services.

Getting ready

The authors recommend that you review the latest information on Microsoft Report Builder at <http://technet.microsoft.com/en-us/library/ee633667.aspx>, as the requirements of the product and supported platforms are regularly updated by Microsoft.

How to do it...

The activities and requirements you must plan for are in the following table:

Category of requirement	Requirement	Description
Client operating system	Windows Vista SP2 and above	Report Builder can be installed on client and server operating systems. This is the requirement for clients.
Software prerequisites	Microsoft .NET Framework 3.5 and 4.0	You must install this requirement by either enabling it in the Control Panel or downloading it from the official Microsoft download site.
Hardware requirements	The supported operating system's minimum requirement is an additional 512 MB of memory and 80 MB of hard disk drive space	These are the minimum requirements in this category, you must increase these resources to improve the user experience when appropriate and relevant.

Category of requirement	Requirement	Description
User privileges	Local administrative rights on the computer you install Report Builder on	The user account used to install or stream Report Builder must have administrative rights on the target computer you install Report Builder on.
SQL Reporting Server requirements	Default port 80 (http) or 443 (https)	The port that Report Builder uses to connect to the report server to publish (save) reports for general access; these ports will vary if a custom port is configured

How it works...

The planning activities discussed are the minimum activities that the authors recommend. The tasks performed at this stage will ensure that you plan for all your requirements before investing time in the actual installation.

There's more...

The streaming installation option has additional requirements.

ClickOnce additional requirements

ClickOnce installation requires the following prerequisites in addition to the requirements discussed in this recipe:

- ▶ Report Builder must be enabled in the advanced properties of the Report Server instance. This is enabled by default and can be verified in **SQL Server Management Studio (SSMS)**. The `EnableReportDesignClientDownload` property must be set to `True`.
- ▶ The user must be assigned role permissions in the Report Manager application.
- ▶ The Report Server must be configured with the appropriate authentication. The recommended authentication method is the default configuration that uses **Integrated Windows Authentication (IWA)**.

See also

- ▶ You can find additional information on the installation planning requirements at <http://technet.microsoft.com/en-us/library/ms365173.aspx>

Installing Report Builder

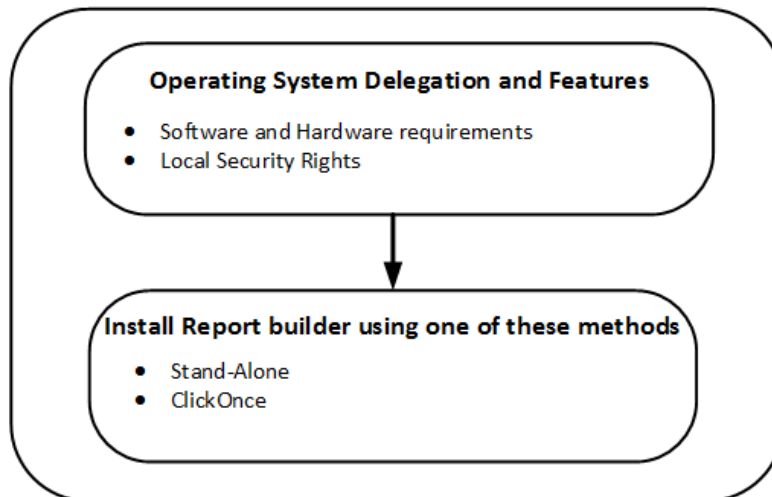
This recipe discusses the steps you follow to perform the two methods of Report Builder installation. The two methods are Standalone (local) and ClickOnce (streamed from the report server).

Getting ready

You must review the information provided in the *Planning the Report Builder installation* recipe.

How to do it...

The following figure provides a visual summary and order of the tasks you need to perform to complete this recipe:

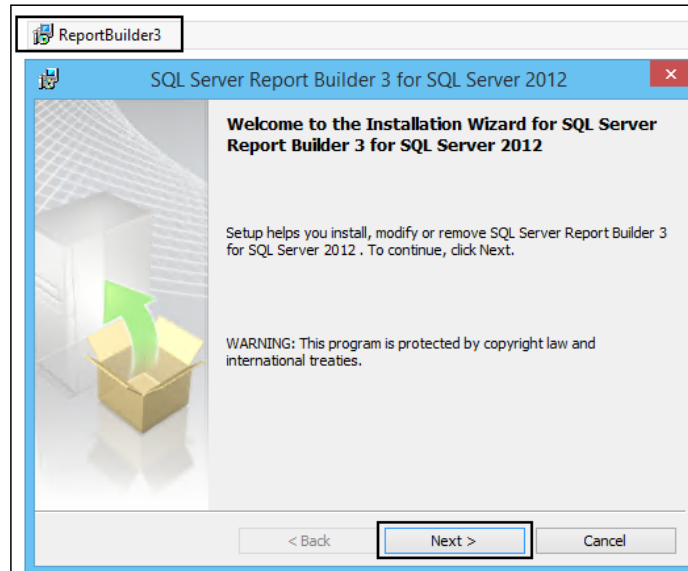


Report Builder Standalone installation

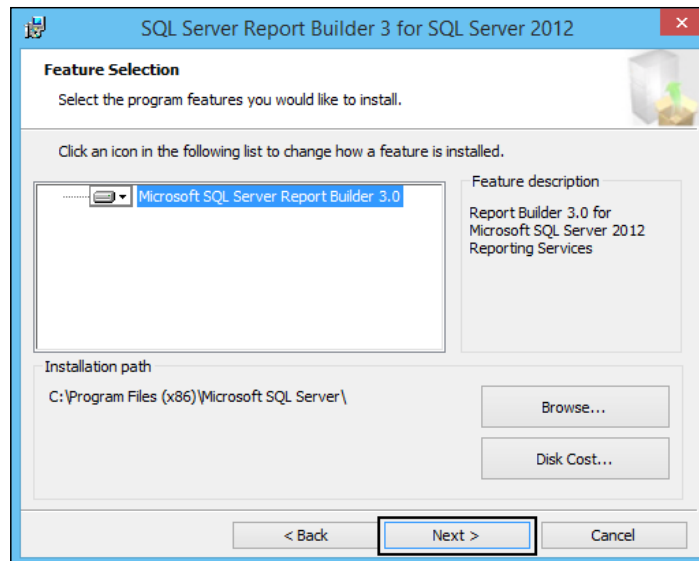
Follow these steps to install the Standalone version of Report Builder on a client operating system machine:

1. Log on to the computer with a user account with permission to perform local installations.
2. Download the latest supported version of Report Builder. The file process discussed in this recipe uses version 3.0 from the Microsoft SQL Server 2012 features.

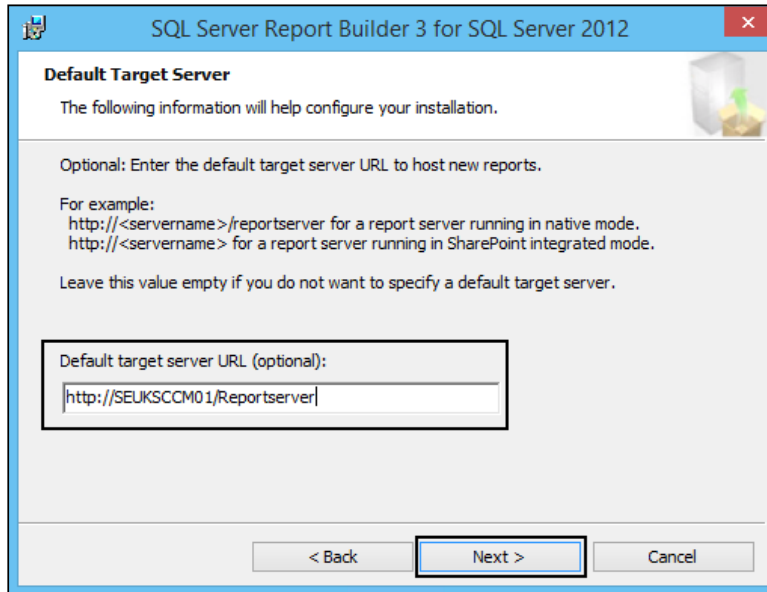
3. Launch the installation using the `ReportBuilder3.msi` file. Click on **Next** on the welcome wizard page, as shown in the following screenshot:



4. Review the **License Agreement** page and accept to continue with the installation. Click on **Next**.
5. On the **Features Selection** wizard page, accept or change the default options. Click on **Next**, as shown in the following screenshot:



6. On the **Default Target Server** page, optionally type the URL for the server that hosts reports. This will typically be the instance of report server you plan to use to store the reports you create, as shown in the following screenshot:



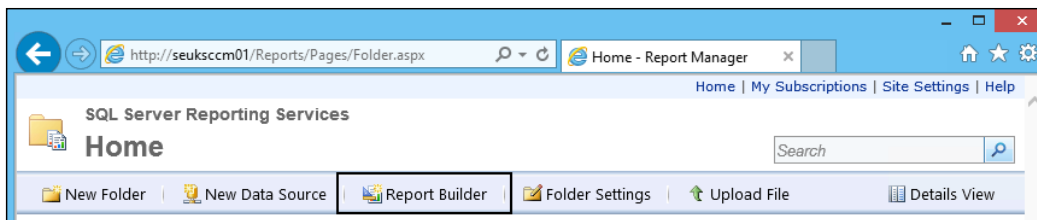
7. On the **Ready to Install the Program** page, click on **Install** to start the installation.
8. Click on **Finish** to close the wizard on successful installation.

This completes the Report Builder Standalone installation steps.

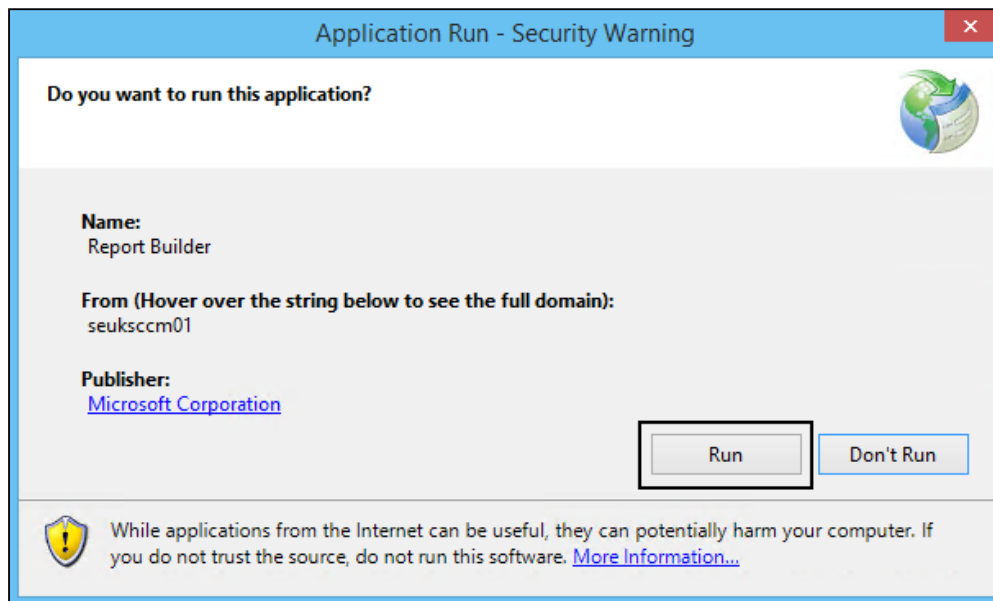
Report Builder ClickOnce installation

Follow these steps to install Report Builder using the ClickOnce method on a client operating system machine:

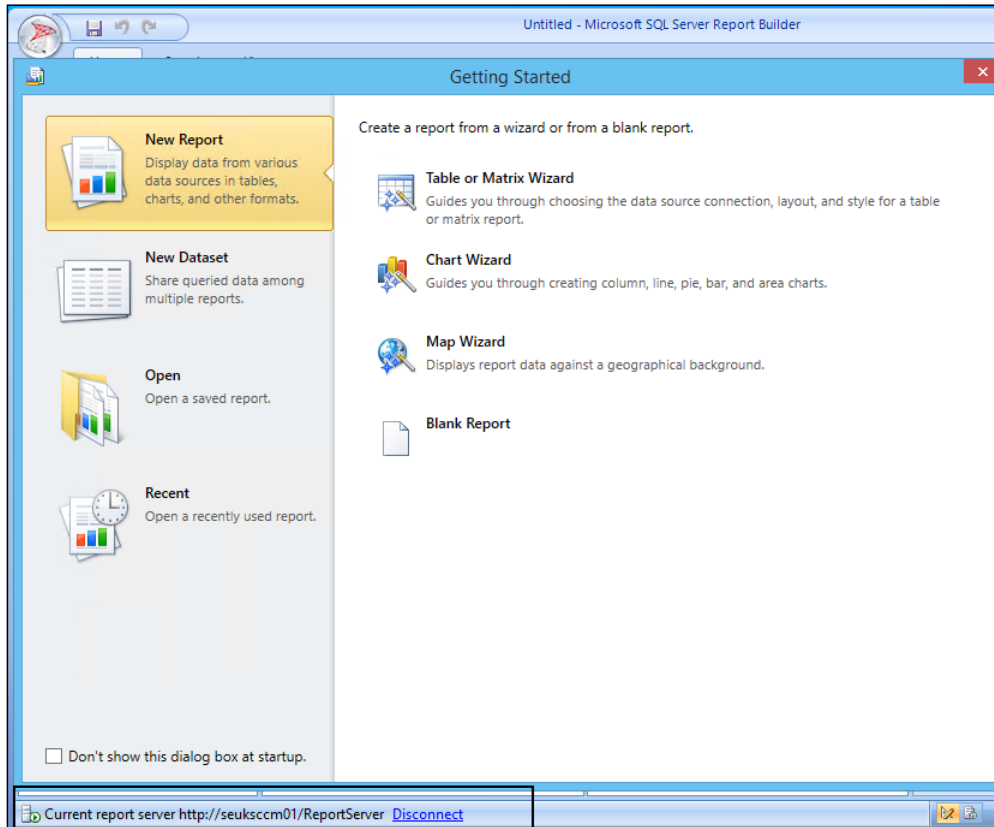
1. Log on to the computer with a user account with permission to perform local installations. The user must also have the right delegation, as discussed in the *Planning the Report Builder installation* recipe.
2. Using a web browser, navigate to Report Manager on the **SQL Server Reporting Services (SSRS)** server, as shown in the following screenshot. The default location is `http://<servername>/Reports`, where `<Server Name>` is the name of the SSRS server.



3. Launch the installation by clicking on the **Report Builder** icon. Click on **Next** on the welcome wizard page.
4. Click on **Run** on the **Application Run - Security Warning** page, as shown in the following screenshot, and accept to continue with the installation. Click on **Next**.



5. Report Builder downloads, and the installation proceeds unattended. On completion, the Report Builder application will launch and connect to the SSRS server, as shown in the following screenshot:



This completes the Report Builder ClickOnce installation.

How it works...

Installing Report Builder is very simple. The most important aspect is to plan and configure all the required prerequisites before you start the actual installation.

The choice you have to make is either using the local installation option or ClickOnce. There are merits for either option but the recommendation is to use the ClickOnce option as the default. The local installation requires you to perform updates manually, whereas the ClickOnce option automatically updates with each initiation.

The installation requires a number of options, and the wizard guides you throughout the process.

See also

- ▶ The official online documentation is updated regularly and should be a point for reference at <http://technet.microsoft.com/en-us/library/hh420371.aspx>

Navigating through the Report Builder interface

This recipe provides a brief overview of the Report Builder interface and serves as a primer for chapters 4 to 8.

Getting ready

You must have installed the Report Builder application using one of the two options discussed in the *Installing Report Builder* recipe.

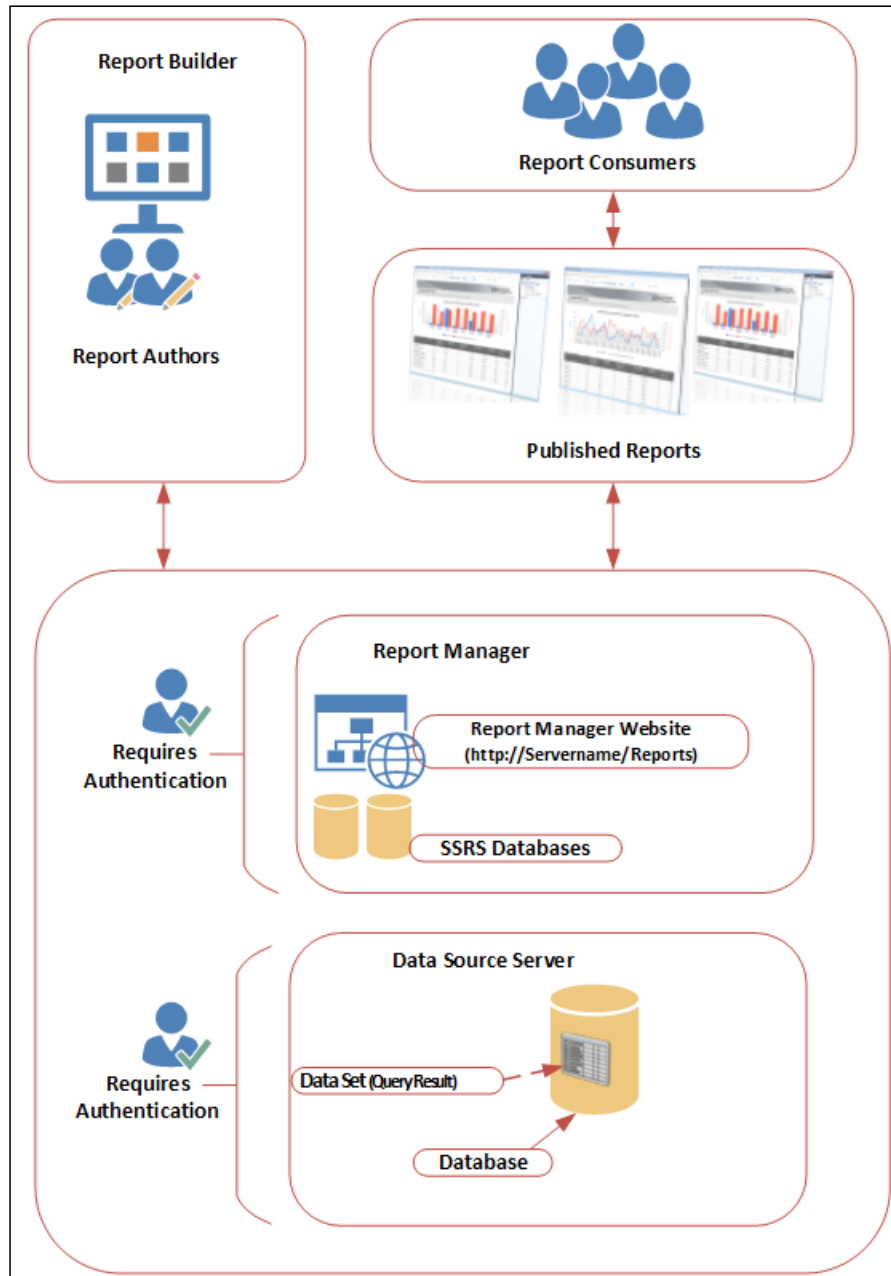
How to do it...

Some of the basic terms used in this book that you need to be familiar with and their descriptions are detailed in the following table:

Report Builder interface and terms	Description
Data source	This is the location of the raw data for reports. The data source comprises of a server (for example, Windows Server), the database, and the connection credentials.
Dataset	The dataset is created using a query to the data source. This is what the report uses as its source for the presentation of the data.
Report Builder	This is the report authoring tool similar in layout to Microsoft Office applications.
Report Manager	This is a component of SQL Server Reporting Services. You connect to the Report Manager website to administer and publish the reports you create in the Report Builder application.
Report types	You have multiple options for the type of reports you create, including tablix (table and matrix), chart, and maps.
Query Designer	This is a graphical query builder interface included with Report Builder.
Report parts	You have the option to save and reuse parts of reports, such as the datasets. You save these artifacts as report parts. The saved report parts become available to other report authors.

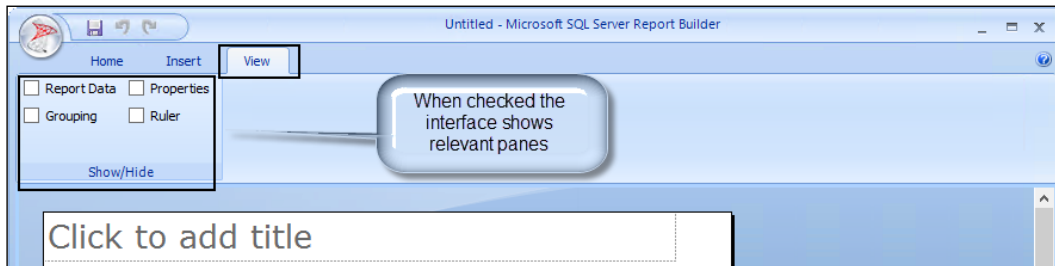
Report Builder interface and terms	Description
Built-in fields	These are the preconfigured fields in Report Builder that are available to you to insert into your report visualization. Examples include execution time, report name, and report server URL.
Parameters	You can define parameter names for your reports. These parameters provide the report consumer with the option to specify accepted values for a report. An example is a report that requires a computer name. You define a parameter for the computer name, and the consumer can change the value to tailor the report.
Images	You have the option to insert images into reports. These images can be imported into Report Builder and stored in the image section.
Report authors	These are users designated as report creators. The users in this category use Report Builder to create reports.
Report consumers	Users in this category access the reports created by the report authors using the supported options available. Examples include web browsers, Microsoft Word, and PDF readers, depending on how the report is rendered.
Report rendering	Rendering is the basic term used in reporting to describe the output format of a report. So, you may receive a report that has been rendered in the PDF format or similarly rendered in Microsoft Excel.
Connection string	When you set up a data source, you are required to provide credentials to the source server. These credentials must have the required level of delegation assigned. The setting configuration is known as a connection string. For example, a connection to a particular database on an MS SQL database server will require either Windows credentials or SQL Server login details.

Additionally, the following figure provides a high-level logical view of the concepts discussed:

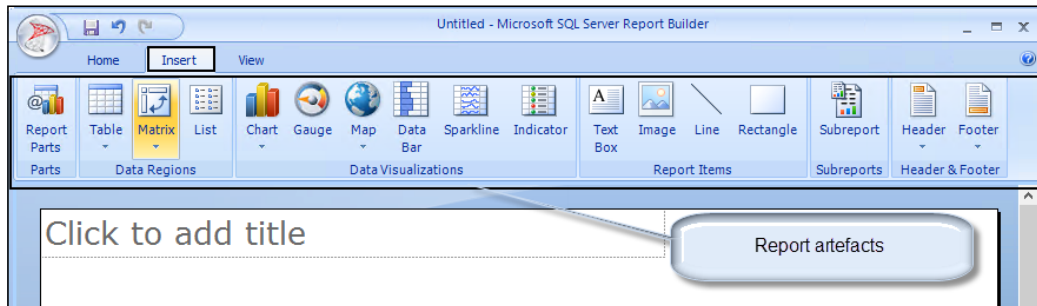


When you launch the Report Builder application, the interface has the following three tabs:

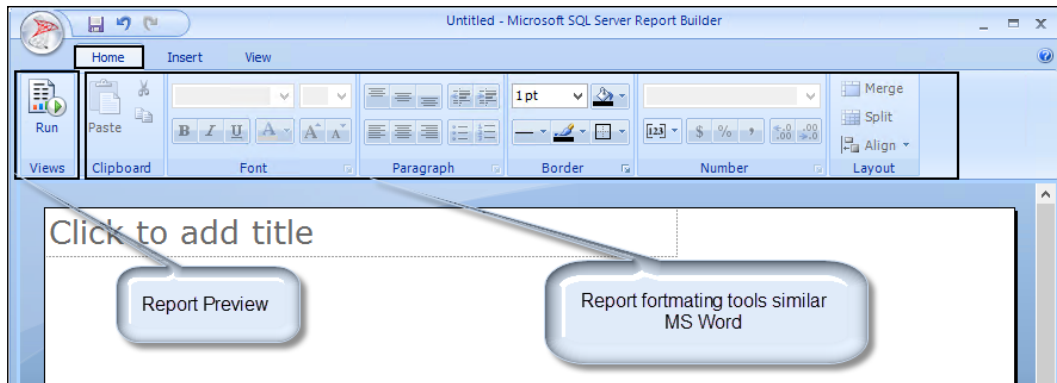
- ▶ **View:** This tab determines what you see in the interface based on the following options:
 - ❑ **Report Data**
 - ❑ **Properties**
 - ❑ **Grouping**
 - ❑ **Ruler**



- ▶ **Insert:** This tab provides you with the tools to create the presentation of your reports. It includes the **Report Parts**, **Data Regions**, **Data Visualizations**, **Report Items**, **Subreports**, and **Header & Footer** controls, as shown in the following screenshot:

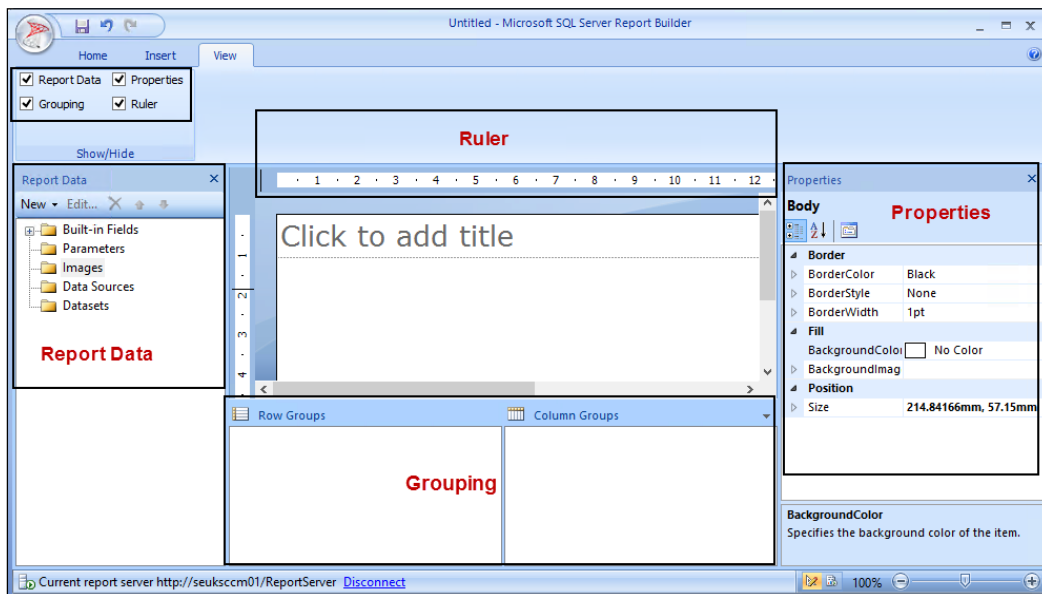


- ▶ **Home:** This tab contains the report preview and formatting tools, as shown in the following screenshot:

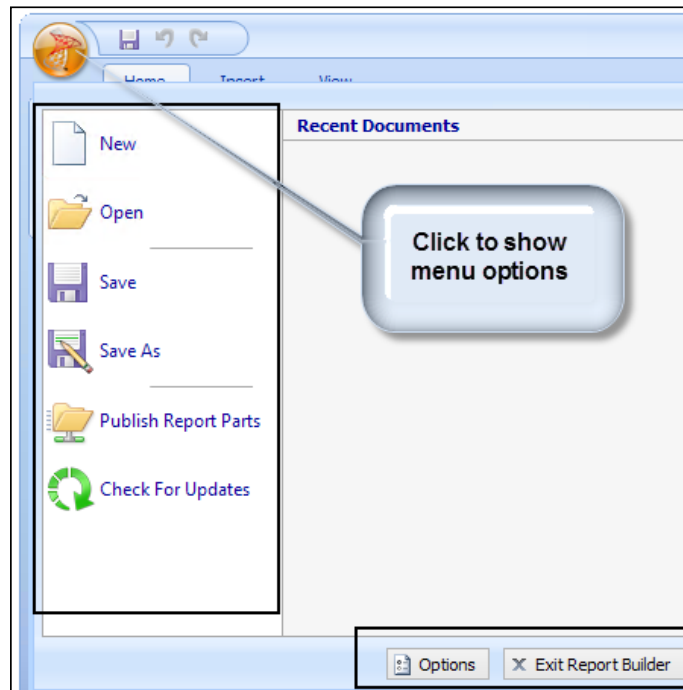


The first recommended task you must perform is enabling the **View** options. Perform this task using the following steps:

1. Launch the Report Builder application using the Report Manager website option by navigating to `http://<ServerName>/Reports` (substitute the URL with your specific environment values) and click on **Report Builder**.
2. Close the **Getting Started** wizard page. Select the **View** tab; ensure that the **Report Data, Properties, Grouping, and Ruler** options are checked.
3. Verify that the options appear as illustrated in the following screenshot:



There are additional options that you access using the application icon (top-left corner in the interface), as shown in the following screenshot:



When you click on the icon, you will get the following options:

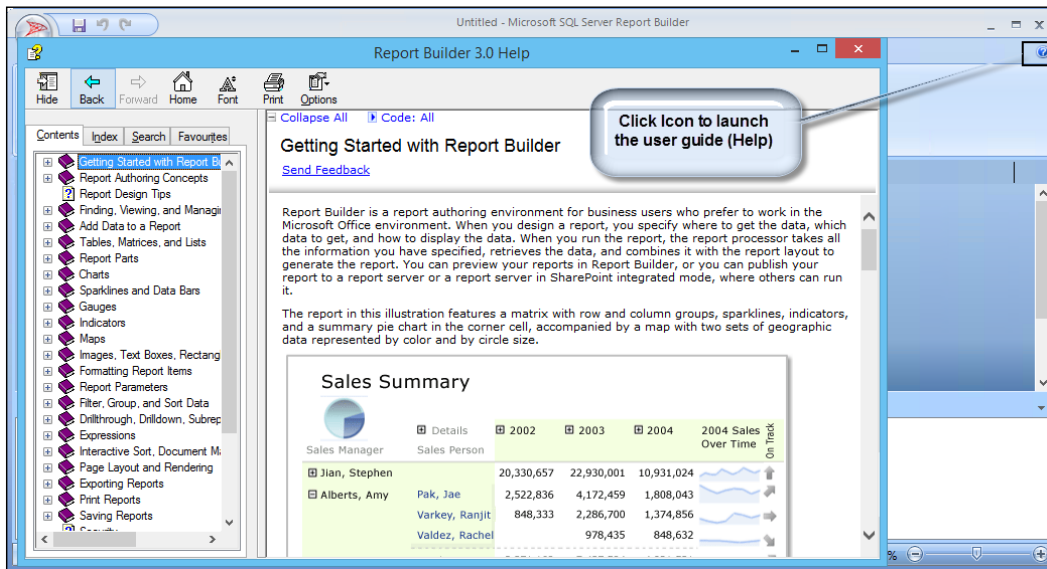
- ▶ **New:** This creates a new report.
- ▶ **Open:** This opens the existing reports.
- ▶ **Save:** This saves active reports (new or existing reports that you have modified in the interface).
- ▶ **Save As:** This saves the report with a new file definition name.
- ▶ **Publish Report Parts:** This makes parts of the report available for reuse or modifies the existing report parts.
- ▶ **Check For Updates:** The ClickOnce option automatically updates if the source SSRS server has updates applied. The option is mainly used for the Standalone installation of Report Builder to check for recent updates to the application.
- ▶ **Options:** These are additional options for Report Builder, including the ability to show the **Getting Started** wizard and the number of recent documents and the ability to set the default report server or SharePoint site.
- ▶ **Exit Report Builder:** This closes the application.

How it works...

This recipe is a primer for the Report Builder application. The focus is on understanding terminology and providing information on navigating the interface prior to creating your first report. You may already be familiar with the interface; in that case, use this recipe as a quick reference and a refresher as required.

There's more...

The Report Builder application has a very extensive built-in help file. You can access this help file using the question icon in the top-right hand corner of the application, as illustrated in the following screenshot:



The help included with Report Builder is extensive and very comprehensive. If you have been in the technology industry for a while, then all the authors can share the famous polite versions of the acronyms **RTM (Read The Manual)** and **JDI (Just Do It)**.

Organizing the reporting environment and delegating access to reports

There is an old but valuable saying that goes *Measure twice, cut once*. The saying maps originally to the carpentry trade where, if you want to build a table, you will:

- ▶ Plan for the size and dimensions of the table using a diagram
- ▶ Buy the raw materials, which usually include planks of wood
- ▶ Measure and cut the required wood to the right size
- ▶ Build the table using your plan (diagram)

This ensures that you know what you are building and reduces the risk of wasting the wood due to incorrectly measured pieces.

The same principle holds for organizing the environment for the reporting cycle (plan, create, manage, and retire). This recipe provides recommended practices and examples on how to best deploy the principle of *Measure twice, cut once* to organizing and managing reports.

Getting ready

You must plan to review *Chapter 1, Understanding the Goals of Reporting*, as a primer to this recipe. Additionally, you must have installed the Report Builder application using one of the two options discussed in the *Installing Report Builder* recipe.

How to do it...

The tasks discussed in this recipe fall into three categories: planning the organization of the environment, creating the organization folders, and delegating access to the environment.

Planning the organization of the reporting environment

The minimum planning tasks that the authors recommend are as follows:

- ▶ Plan report manager folders
- ▶ Plan reporting delegation roles

Plan report manager folders

The first part of the organization of folders deals with how you store reports. The reports you create are stored and by default accessible from the report manager website. You have the option to store these reports in folders for ease of organization, and they additionally provide you with folder-level security delegation.

The table that follows provides a sample structure you can follow for an environment:

Folder name	Folder level	Description
Data_Sources	Root folder	This folder is created for shared data sources. You can create subfolders under this root folder as required to provide additional granular management.
Data_Sets	Root folder	This folder is created for shared datasets. You can create subfolders under this root folder as required to provide additional granular management.
Custom_WebReports	Root folder	This is a root folder for reports accessed using a web browser. Plan to save reports optimized for web rendering in this folder and subfolders.
Report_Subscriptions	Root folder	This is a root folder for subscription-enabled reports. Plan to separate subscription reports from web reports to provide maximum flexibility in types of reports and security delegation.
Custom_WebReports\ General	Subfolder	This is a subfolder for generally accessible reports. It is typically used to store reports available to all report browsers.
Custom_WebReports\ Inventory	Subfolder	This is an example System Center Configuration Manager (SCCM) category subfolder. It is a subfolder designated for inventory reports.
Custom_WebReports\ Compliance	Subfolder	This is an example SCCM category subfolder designated for compliance reports.
Report_Subscriptions\ Inventory\ Inventory\ Inventory	Subfolder	This is a subfolder for inventory category subscription reports.
Report_Subscriptions\ Compliance	Subfolder	This is a subfolder for compliance category subscription reports.
Report_Subscriptions\ Inventory\ Inventory\ Email	Second-level subfolder	This is a a subfolder for inventory category subscription reports delivered by e-mail.
Report_Subscriptions\ Inventory\ Inventory\ FileShare	Second-level subfolder	This is a subfolder for inventory category subscription reports delivered to a file share.

The structure and guidance is an example that focuses on functions of an organization. Plan to implement what works in your organization.

Planning report delegation roles

The following are the planning report delegation roles:

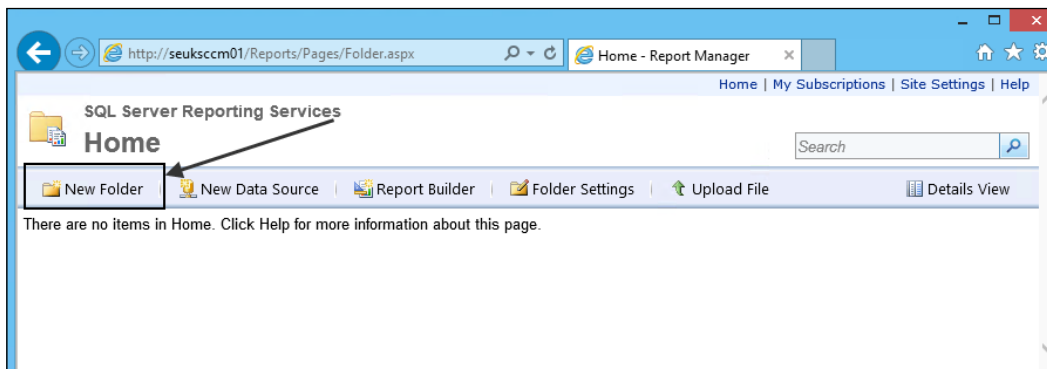
Organization role	Active Directory group	Description
System Center Report Admins	SC_REP_ADMINS	This has the highest level of reporting security access.
System Center Report Authors	SC_REP_AUTHORS	Users in this group can create reports using the authorized reporting tool.
Report Browsers	SC_REP_BROWSERS	Users in this group can browse reports stored in the general category folders.
Compliance Report Browsers	SC_REP_COMPL	Users are authorized to view this category of reports.
Inventory Report Subscribers	SC_REP_INV	Users are authorized to subscribe to this category of reports.
Compliance Report Subscribers	SC_REP_COMPL_SUB	Users are authorized to subscribe to this category of reports.

The structure and guidance is an example that focuses on the recommended and real-world security practices of industry-standard organizations. Plan to implement what works in your organization.

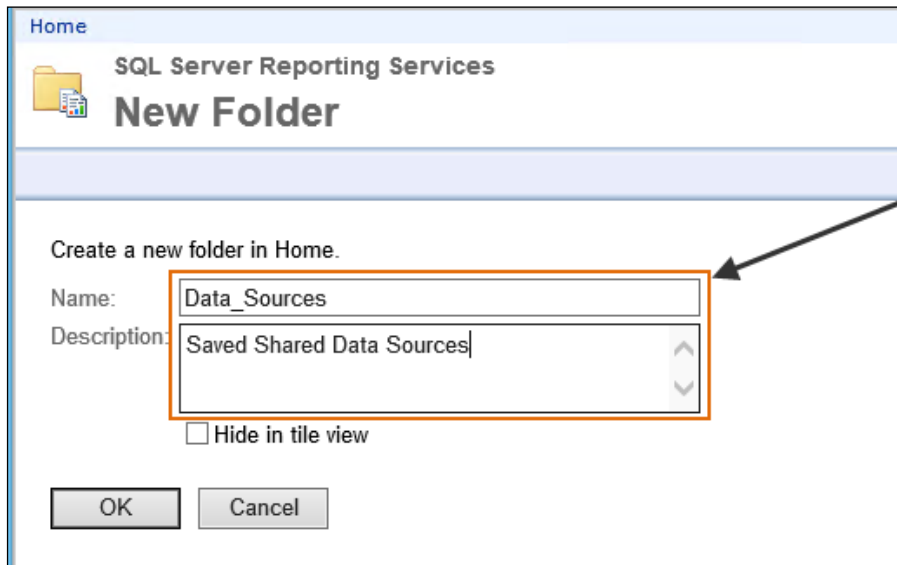
Creating the reporting organization folders

Once you have completed the planning, the next step is to create the folders. Perform these steps to complete this task:

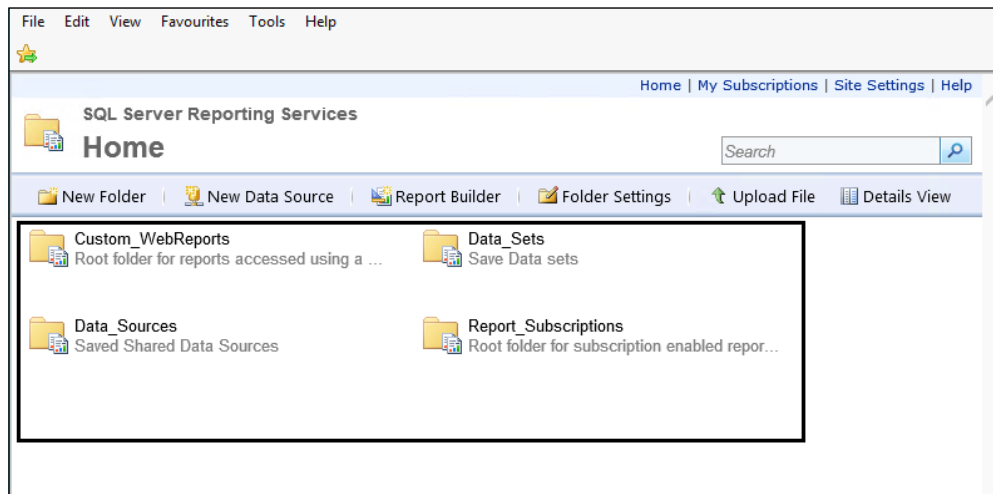
1. Launch the Report Builder application using the **Report Manager Website** option by navigating to `http://<ServerName>/Reports` (substitute the URL with your specific environment values).
2. On the Report Manager home page, click on **New Folder** to launch the new folder **Properties** page, as shown in the following screenshot:



3. Provide a folder name, and optionally (but this is recommended), provide a description for what the folder will store. Click on **OK** to complete creating the folder, as shown in the following screenshot:



4. Repeat steps 2 and 3 to create the root-level folders.
5. You can perform the same steps to create the subfolders, but you must click on the relevant root folder to expand it before performing steps 2 and 3.
6. Verify that the root folders are created as you planned, as shown in the following screenshot:

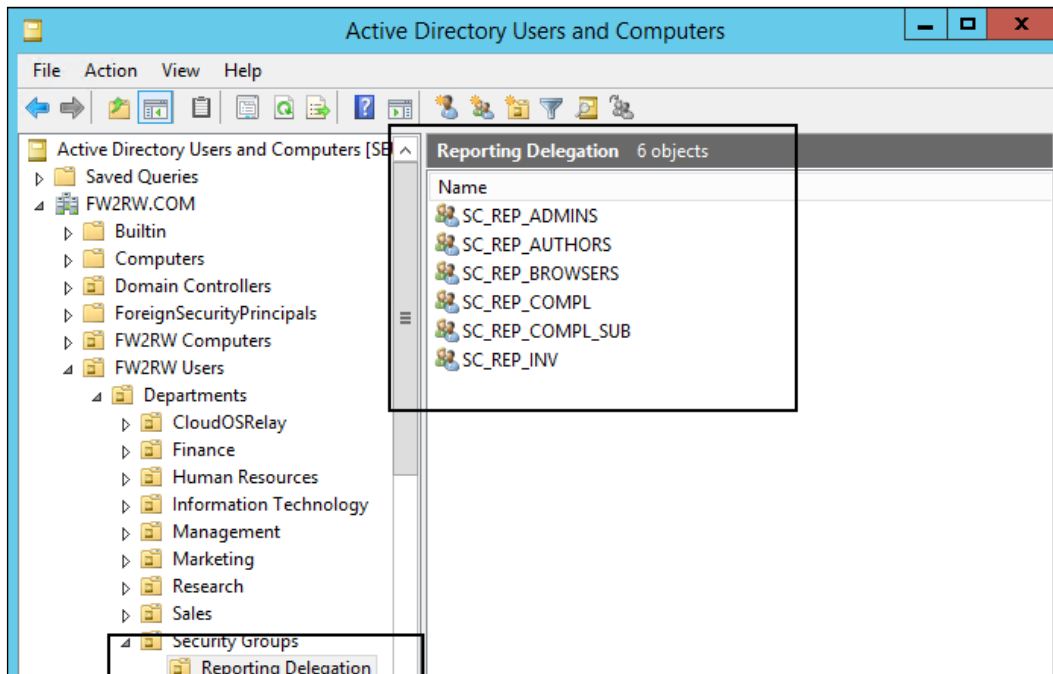


7. Verify that the subfolders are created as you planned. You must click on the relevant root folder and traverse to the subfolders.

Delegating access to the organization folders

The next task you perform after planning the organization and creating the initial folders for reporting is security delegation. Use the table in the *Planning reporting delegation roles* subsection of this recipe as a guide. You must first create the Active Directory groups (this book assumes that you are performing these tasks in a Microsoft Active Directory domain environment).

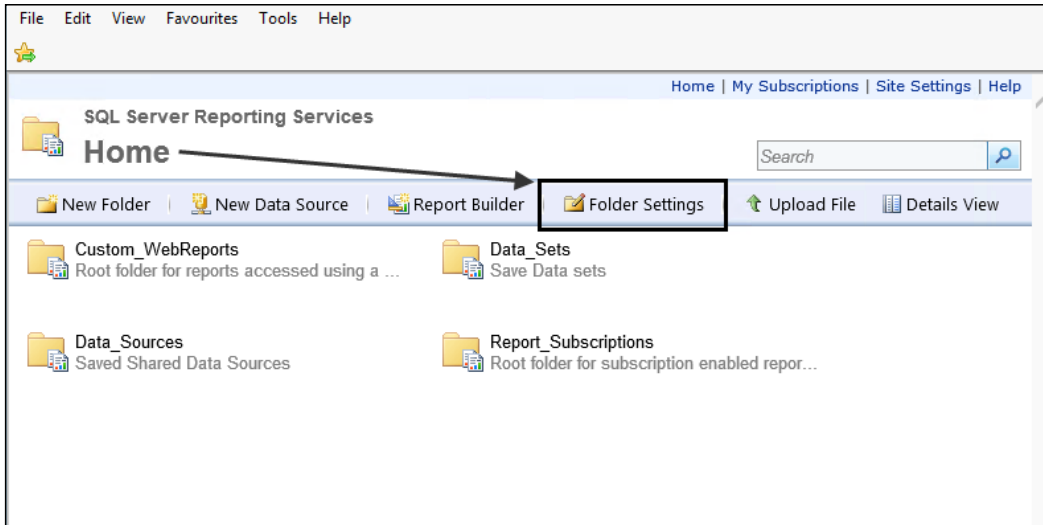
The following screenshot shows the groups created in an **Organization Unit (OU)** dedicated to the reporting security organization:



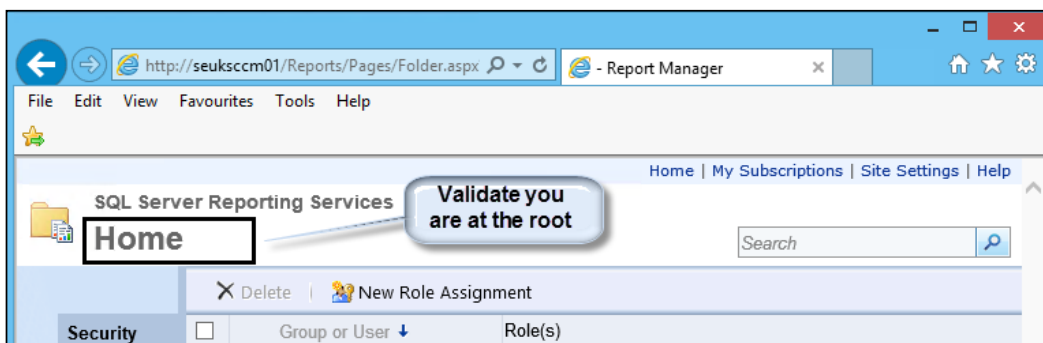
You need the right permissions in Active Directory to create groups. You have to request this action from the administrators in your organization if you do not have the relevant rights.

The steps you will perform in Report Manager are as follows:

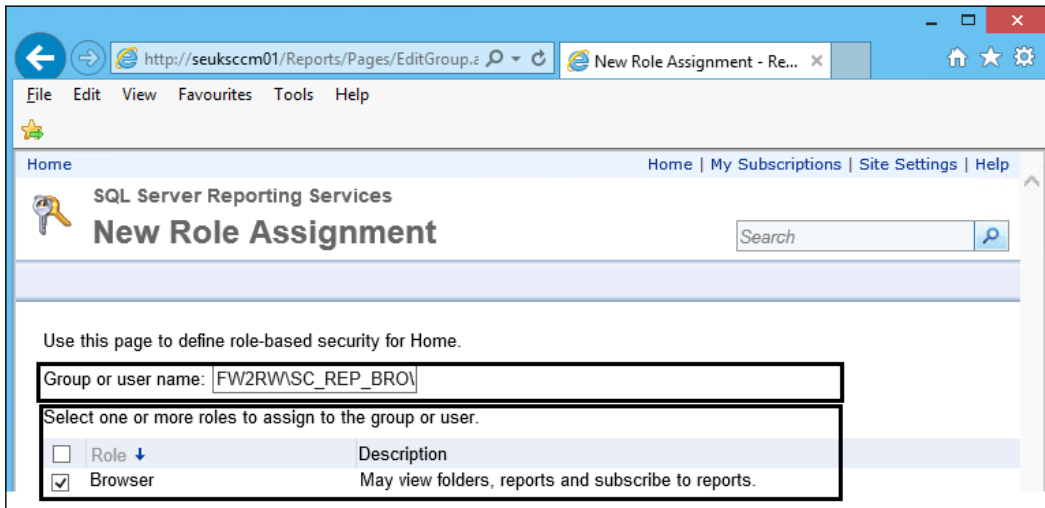
1. Launch the Report Builder application using the **Report Manager Website** option by navigating to `http://<ServerName>/Reports` (substitute the URL with your specific environment values) and click on **Report Builder**.
2. On the Report Manager home page, click on **Folder Settings** to launch the root folder security properties page, as shown in the following screenshot:



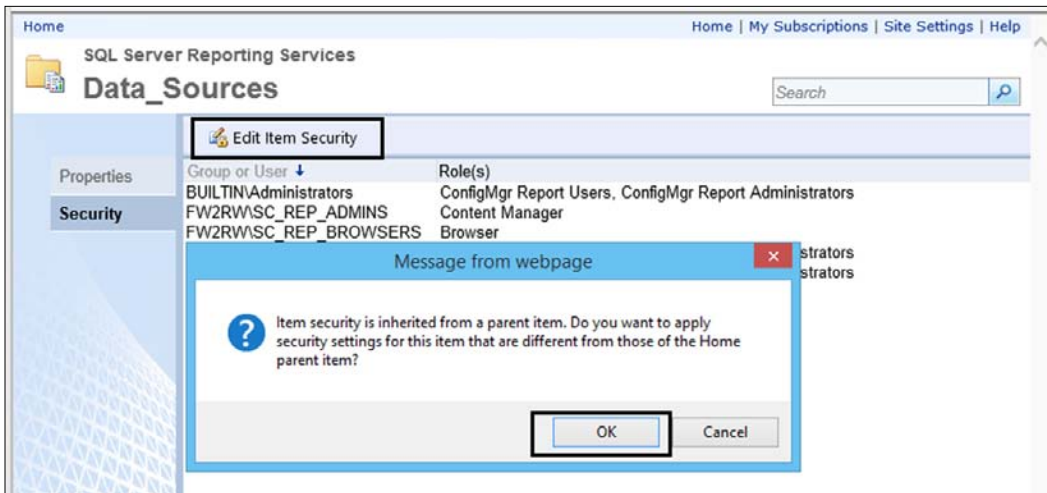
3. The Report Manager root folder **Properties** page is presented. Validate that you are at the root-level folder, as illustrated in the following screenshot:



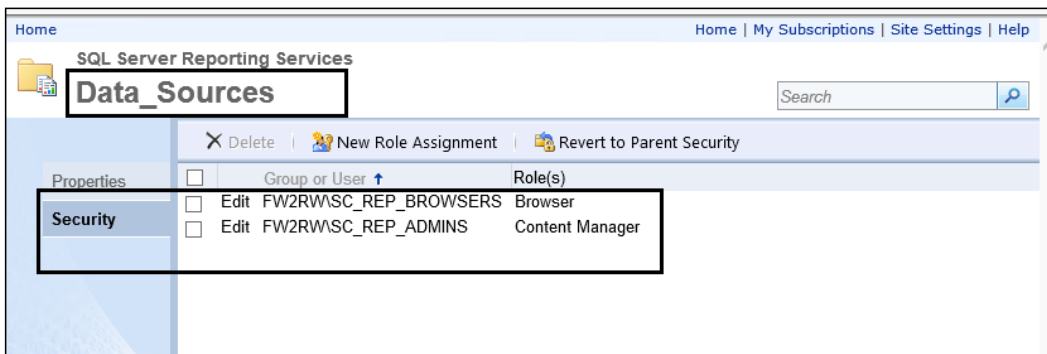
4. Click on **New Role Assignment** to navigate to the **New Role Assignment** page. Type the group you created for the reporting role using the <Domain>\<Group Name> syntax. Select the role to assign to the group. In the screenshot, the Active Directory group is FW2RW\SC_REP_BROWSERS, and the security role is **Browser**. Click on **OK**.



5. Repeat step 4 using in our example FW2RW\SC_REP_ADMINS as the domain group for the role, and the security role is **Content Manager**.
6. The process for the subfolders is similar, but you must initially break the inheritance from the root folder.
7. Click on the **Home** link to return to the root folder. Click on one of the root folders, for example, *Data_Sources*. Ensure that the folder is expanded when you click on the name.
8. Click on **Folder Settings** and then on **Edit Item Security**. Then, click on **OK** on the **Message from webpage** window to break the inheritance from the root folder, as shown in the following screenshot:



- Check the security roles that should not be part of the folder administrators and then click on **Delete**. Click on **OK**. Add any additional required roles. In the example screenshot, the `SC_REP_ADMINS` group is the only group that is assigned the rights to the data source folder as content managers. The `SC_REP_BROWSERS` group is assigned the **Browser** role:



- Repeat the delegation steps using the appropriate groups and roles for your environment to complete the access rights delegation.

How it works...

This recipe focused on preparing the reporting environment by first planning, then creating the folders used for delegation, and finally performing the delegation steps using the Report Manager interface.

You perform these steps initially as a start and proceed to modify as and when your requirements change. The recipe provides the recommended steps and examples, but you must tailor the execution to your environment.

See also

- ▶ You can find additional information on the security delegation at <http://msdn.microsoft.com/en-us/library/ms159820.aspx>

Creating data sources

This recipe provides the steps you must follow to create data sources. There are two kinds of data sources: embedded and shared. The steps provided show how to create these two types of data sources for your reports.

Getting ready

You must have installed the Report Builder application using one of the two options discussed in the *Installing Report Builder* recipe.

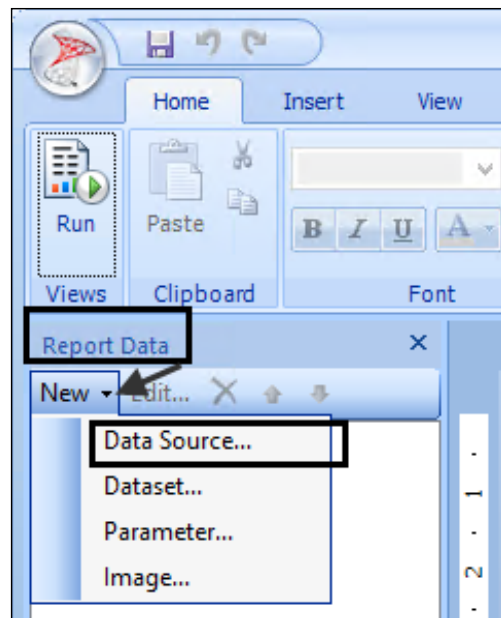
How to do it...

In this recipe, we will create a data source for an SCCM database. The recipe will demonstrate how to create an embedded data source and a shared data source.

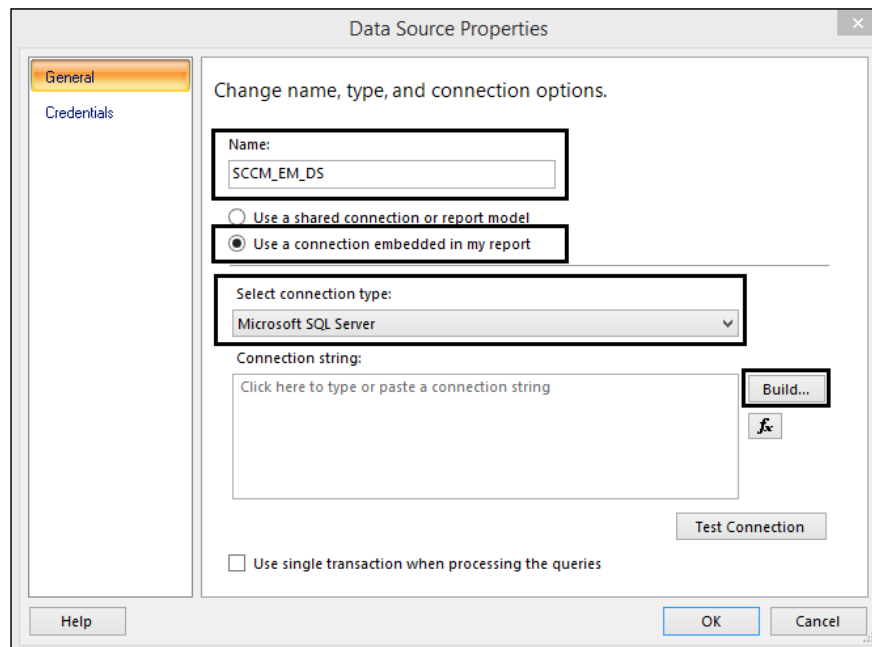
Creating an embedded data source

Follow these steps to create an embedded data source:

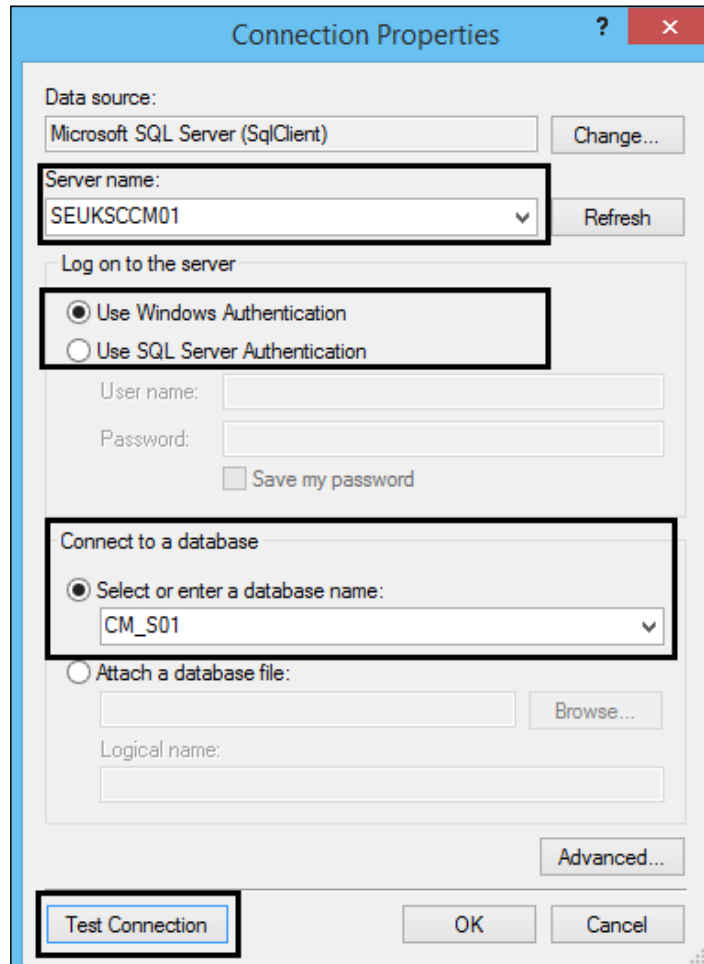
1. Launch the Report Builder application using the **Report Manager Website** option by navigating to `http://<ServerName>/Reports` (substitute the URL with your specific environment values) and then click on **Report Builder**.
2. Close the **Getting Started** page. Ensure that the **Home** tab is selected. Under **Report Data**, click on **New** and select **Data Source...**, as shown in the following screenshot:



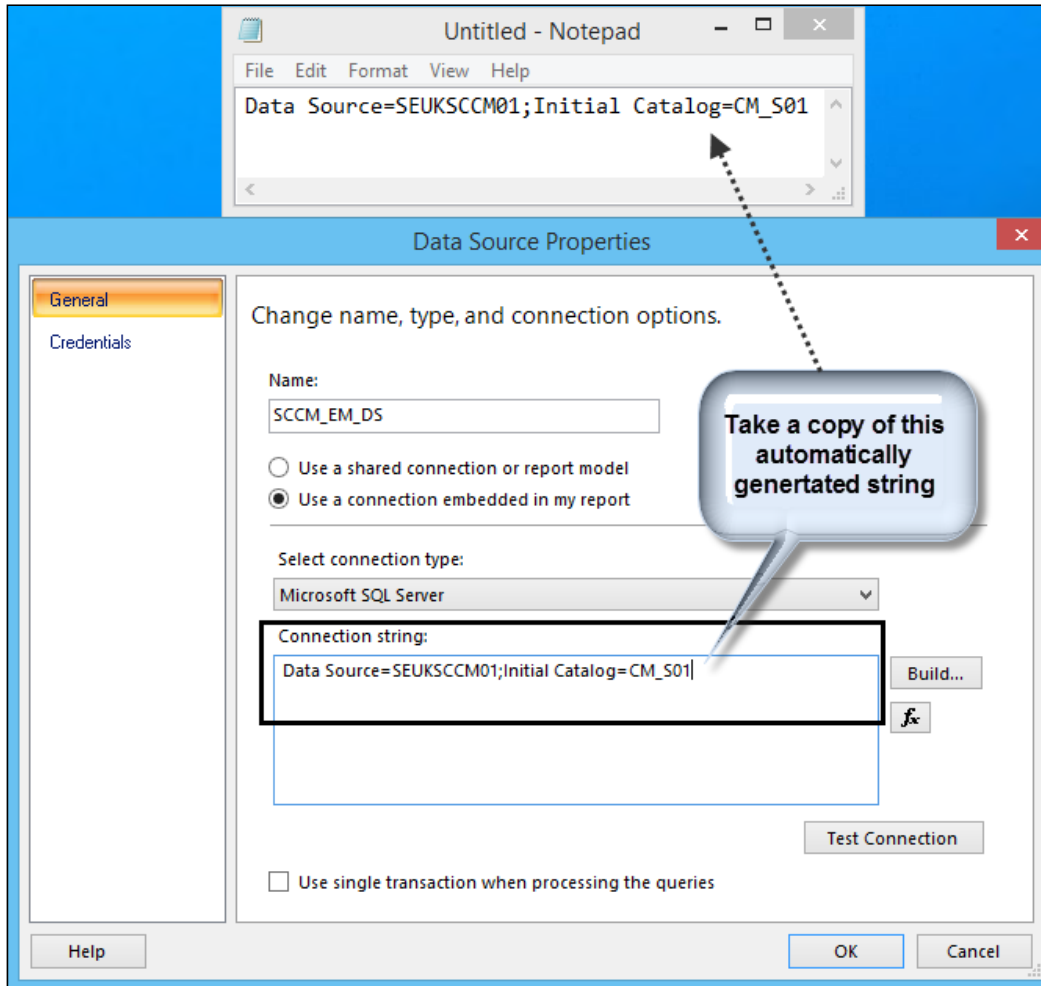
3. On the **General** tab, type a name for the data source in the **Name:** field. Select **Use a connection embedded in my report** and ensure that **Microsoft SQL Server** is selected as the connection type, as shown in the following screenshot:



4. Under **Connection String**, click on **Build...** Type the data source server name (for this example, the SCCM server is named SEUKSCCM01) and accept the **Use Windows Authentication** to log on to the server.
5. Under **Connect to a database**, select the SCCM database (for this example, the database is named CM_S01).
6. Click on **Test Connection** to validate the connection settings, as shown in the following screenshot:



- Click on **OK**. Open Notepad or any text editor. While in the **Data Sources Properties** page, copy the text in the **Connection string** box, as shown in the following screenshot:

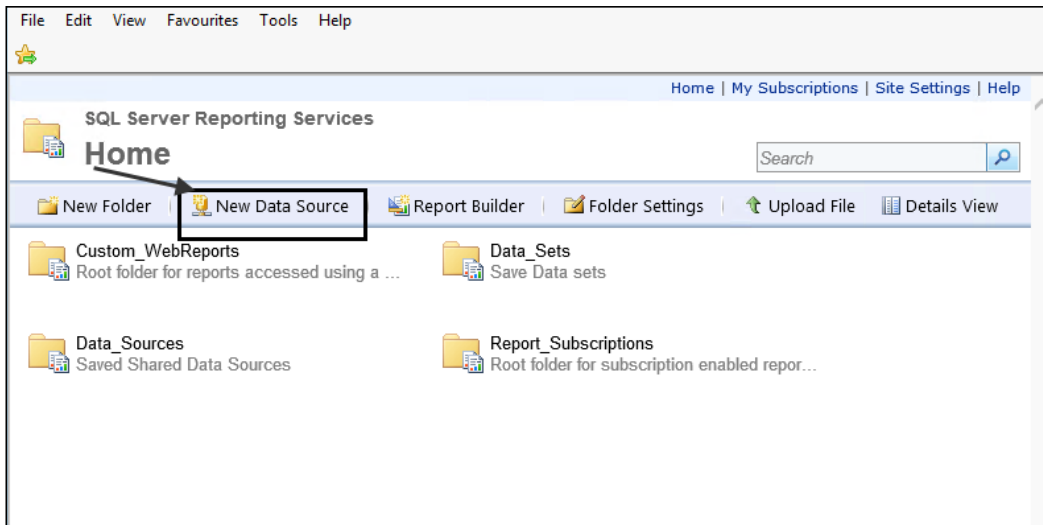


- Click on **OK** to complete the embedded data source creation.
- Leave the Report Builder application open.

Creating a shared data source

Follow these steps to create a data source:

1. Launch the Report Builder application using the **Report Manager Website** option by navigating to `http://<ServerName>/Reports` (substitute the URL with your specific environment values).
2. On the **Report Manager** home page, click on **New Data Source** to launch the **New Data Source** properties page, as shown in the following screenshot:



3. On the **New Data Source** page, type a name for the data source in the **Name:** field. Optionally, provide a description in the **Description** field. Select **Microsoft SQL Server** for **Data source type**. Type the connection string information you copied to notepad from the Creating an embedded Data Source section (step 7). Select **Windows integrated security** and click on **Test Connection**, as shown in the following screenshot:

Home | My Subscriptions | Site Settings | Help

SQL Server Reporting Services

New Data Source

Search

Name:

Description:

Hide in tile view

Enable this data source

Data source type:

Connection string:

Connect using:

Credentials supplied by the user running the report

Display the following text to prompt user for a user name and password:

Use as Windows credentials when connecting to the data source

Credentials stored securely in the report server

User name:

Password:

Use as Windows credentials when connecting to the data source

Impersonate the authenticated user after a connection has been made to the data source

Windows integrated security

Credentials are not required

Connection created successfully.

4. Verify that the connection is a success and click on **OK**.
5. Optionally, click on the new data source to open its properties. Click on **Move** and select the data source folder you created in the *Creating Reporting organizational folders* section of the *Organizing the reporting environment* recipe.
6. Click on **Apply** to complete the move.
7. Close the Report Builder application from the previous task steps.

How it works...

The recipe illustrated how to create the two types of data sources: embedded and shared. You can create shared and embedded data sources using the Report Manager website. You can create only embedded data sources in the Report Builder application.

The difference between the two types of data sources is that embedded data source is limited to the report it is defined in, whereas the shared data source can be referenced by many reports.

Creating datasets and a basic report in Report Builder

In this recipe, you will learn how to create a dataset and use the dataset to create a basic report in Report Builder.

Getting ready

You must have installed the Report Builder application using one of the two options discussed in the *Installing Report Builder* recipe. Additionally, you must have performed the steps required to create a shared data source, as discussed in the *Creating data sources* recipe. You will need access to a functional SCCM database with active agents to complete the steps discussed.

The following table lists the details required to complete this recipe:

Requirement or information	Description
Create a report that lists the computers in SCCM with operating system names. The report must be accessed using Internet Explorer.	This is the report requirement, including output target
Data source has been created. It is accessible and stored in a folder on the reporting server called <code>Data_Sources</code> .	You can use a precreated shared data source
The reporting server URL is <code>http://SEUKSCCM01/Reports</code> .	This is the reporting server running the Report Manager
SCCM views with required data: <code>V_GS_SYSTEM</code> and <code>V_GS_OPERATING_SYSTEM</code>	These are SCCM views containing required report data
Fields required: computer name and operating system name	These are the required columns in the report
Column that connects the views <code>RESOURCEID</code>	This is a column common to both views and used as the join

The table will serve as an input for the tasks required to complete this recipe.

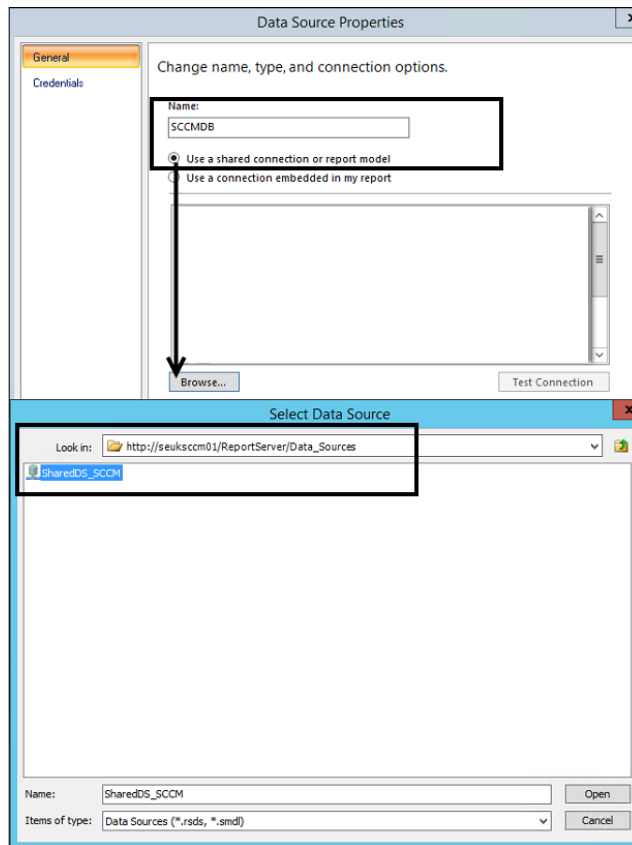
How to do it...

The steps to complete this recipe are as follows:

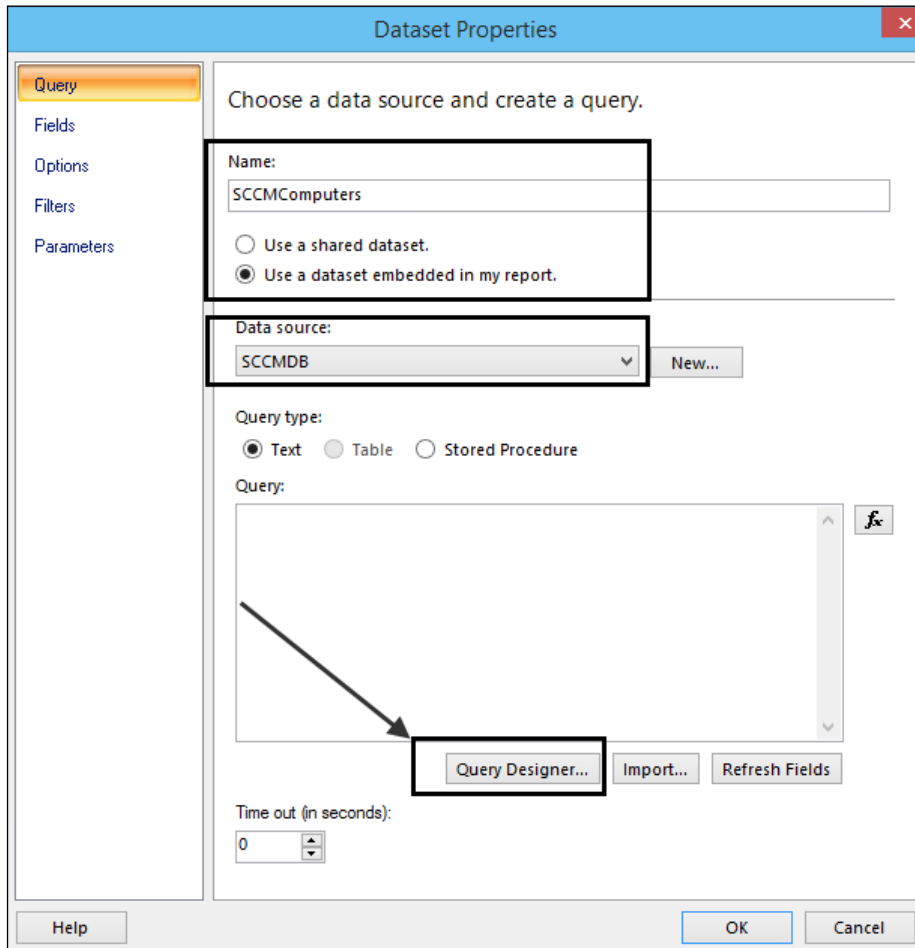
Creating a dataset

You must first create the dataset. The dataset is the result of a query to the data source and is created in the Report Builder application. Perform the following steps:

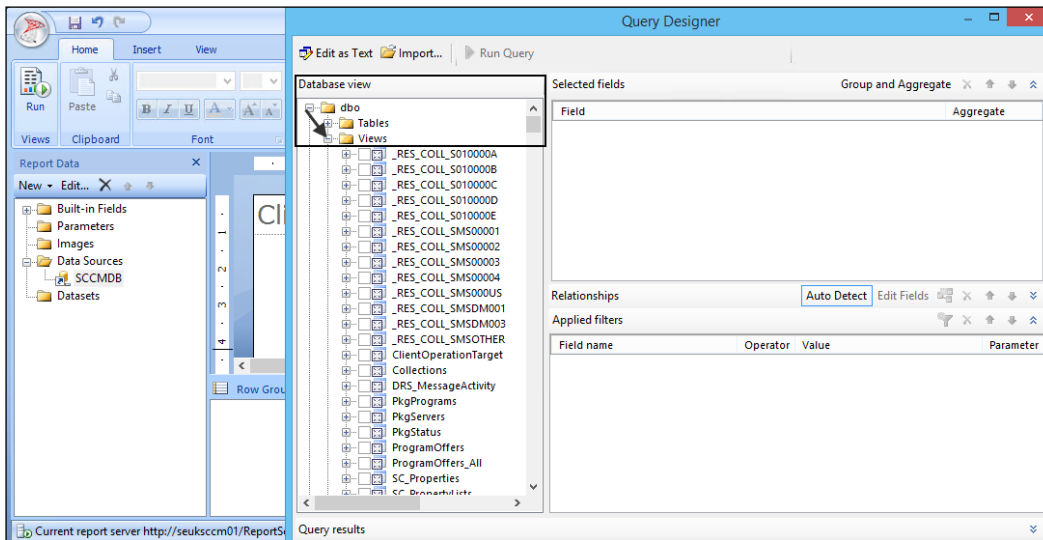
1. Launch the Report Builder application using the **Report Manager Website** option by navigating to `http://<ServerName>/Reports` (substitute the URL with your specific environment values) and then click on **Report Builder**.
2. Close the **Getting Started** page. Ensure that the **Home** tab is selected. Under **Report Data**, click on **New** and select **Data Source...**
3. On the **General** tab, type a name for the data source. Ensure that **Use a shared connection or report model** is selected and click on **Browse...**, as shown in the following screenshot:



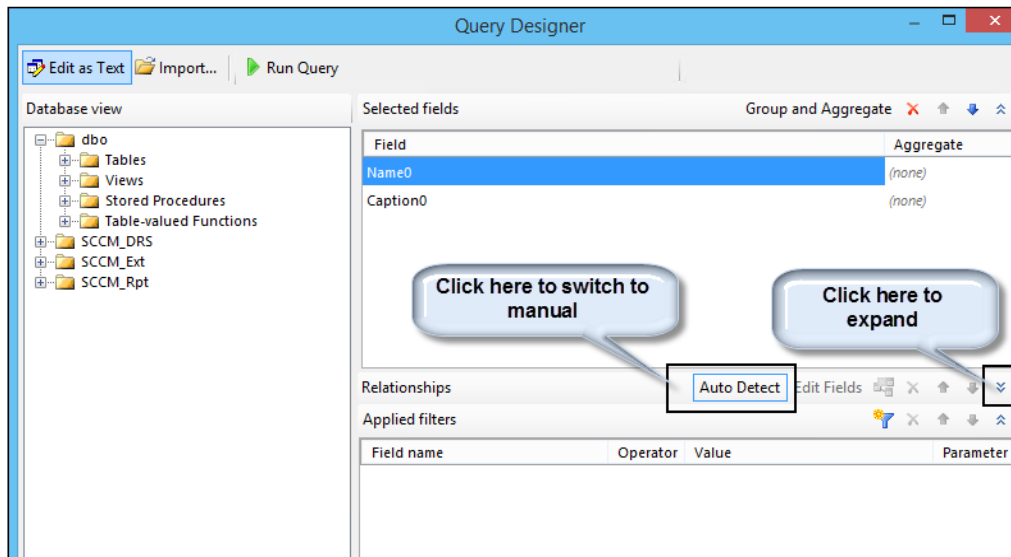
4. Navigate to the `Data_Sources` folder and select the data source available. Click on **Open** when the data source is selected. Note that you must have completed the steps to create the shared data source in the previous recipe.
5. Click on **OK**. Ensure that the **Home** tab is selected. Under **Report Data**, click on **New** and select **Data Set...**
6. Type a name for the dataset. Select **Use dataset embedded in my report**. Under **Data Source**, ensure that the data source you created in step 4 is selected. Click on **Query Designer...**, as shown in the following screenshot:



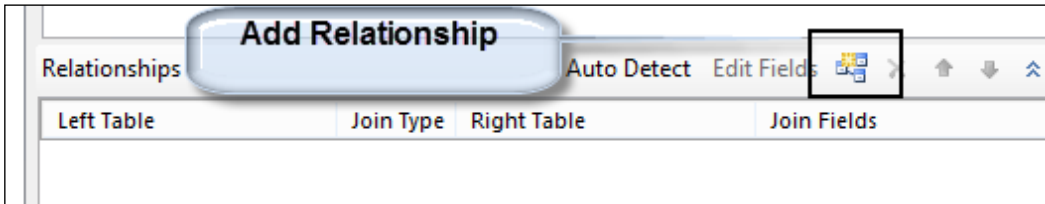
- Under **Database view**, expand **dbo** and then expand **Views**, as shown in the following screenshot:



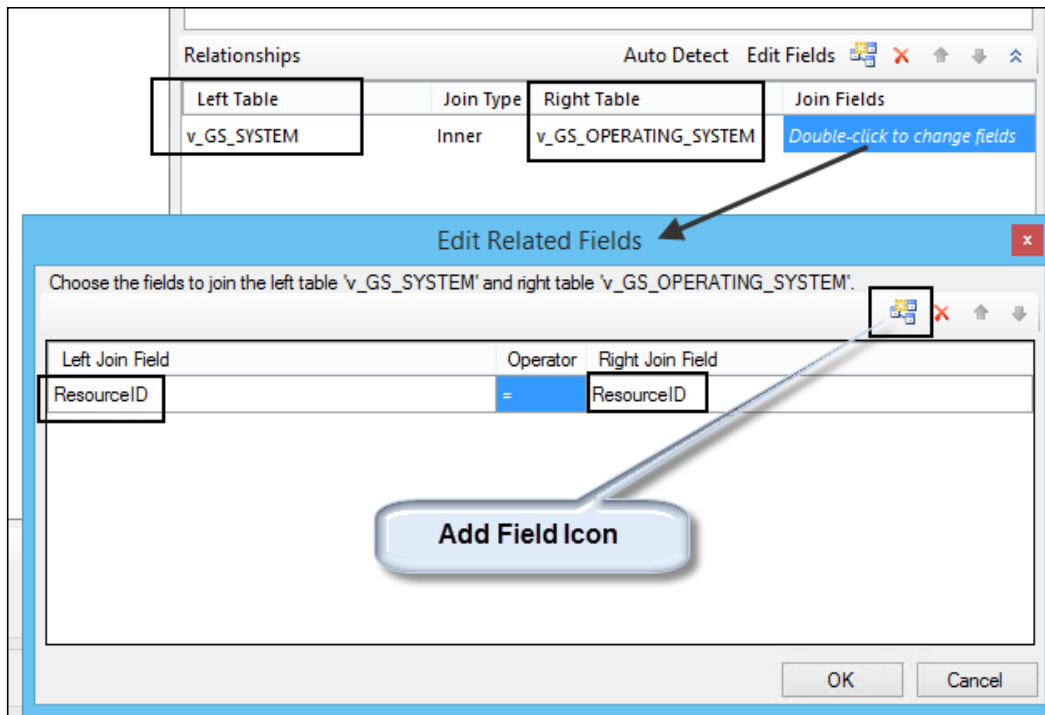
- Scroll down the list of views. Expand **V_GS_OPERATING_SYSTEM** and select **Caption0**.
- Scroll down the list of views. Expand **V_GS_SYSTEM** and select **Name0**.
- In the **Relationships** section, click on the double down arrow and click on **Auto Detect**, as indicated in the following screenshot:



11. In the **Relationships** section, click on **Add Relationship**, as indicated in the following screenshot:



12. Select **V_GS_SYSTEM** as **Left Table** and **V_GS_OPERATING_SYSTEM** as **Right Table**. Double-click on the space under **Join Fields**. Click on the **Add Field** button in **Edit Related Fields**. Select **ResourceID** under **Left Join Field** and **ResourceID** again under **Right Join Field**. Click on **OK**, as shown in the following screenshot:

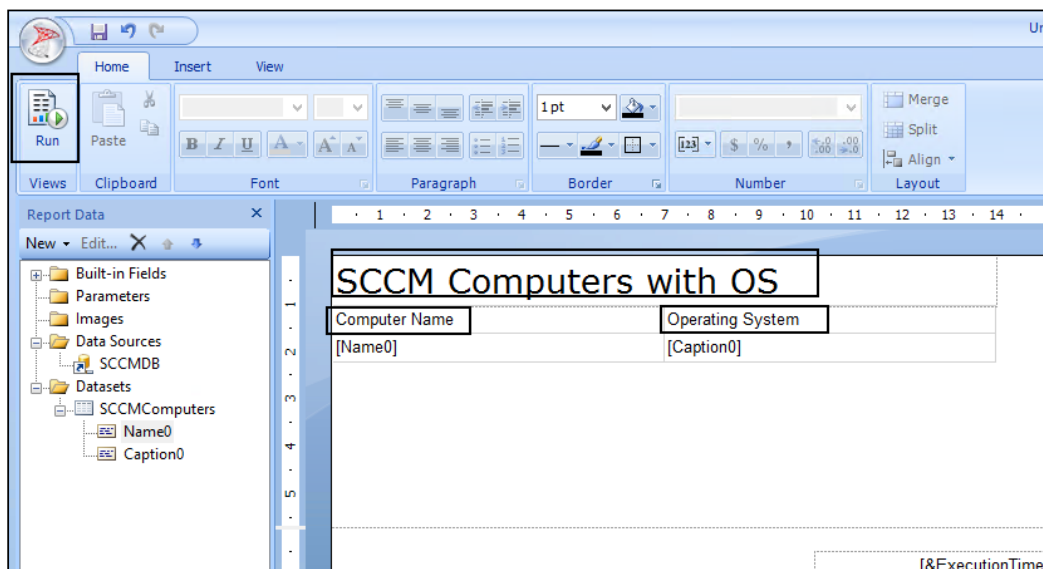


13. Click on **OK** twice. This completes the dataset creation.

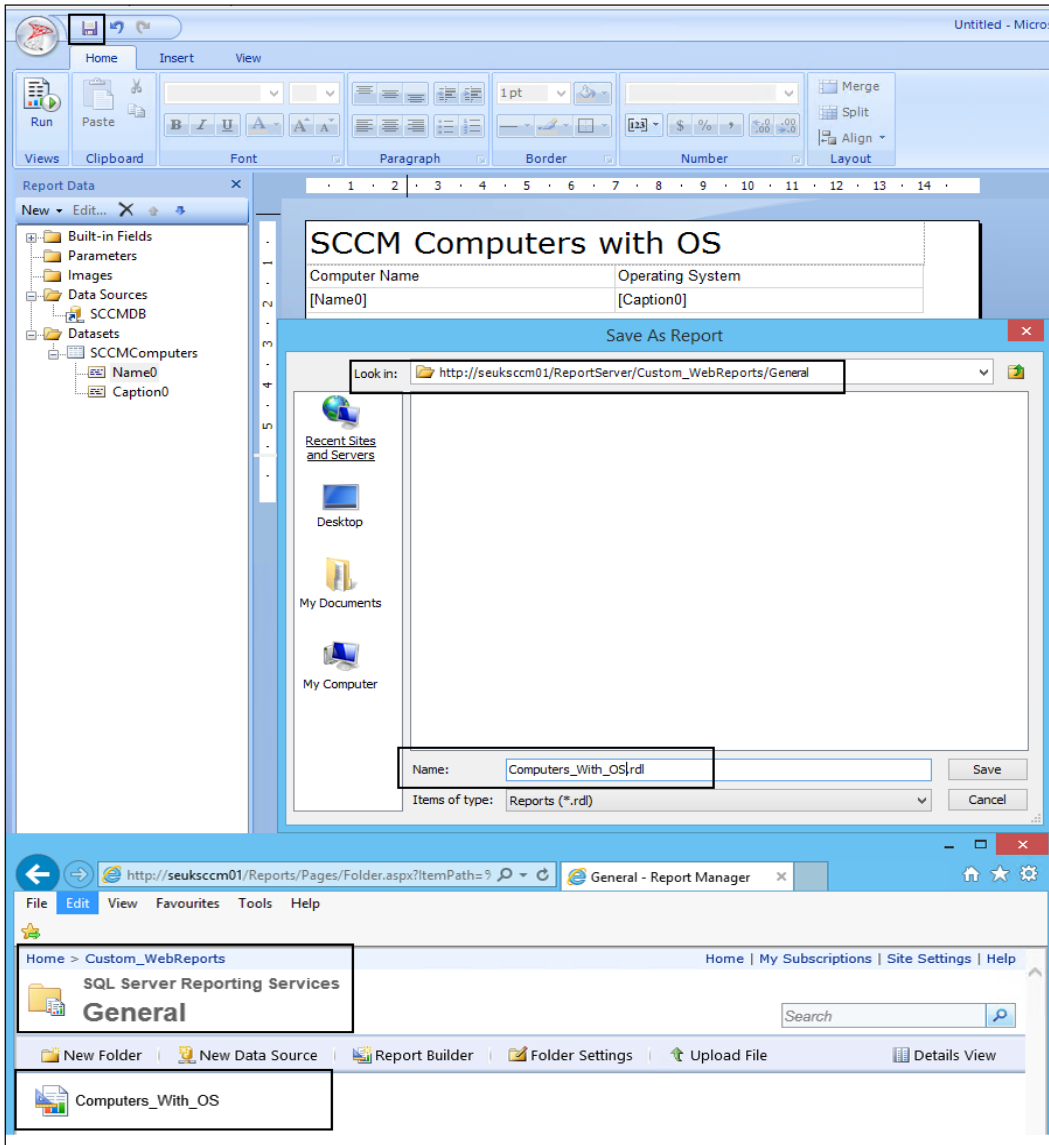
Creating a basic report

Use the following steps to create a basic report; this report uses the data source and dataset created in this chapter:

1. Ensure that the **Insert** tab is selected. Under **Data Regions**, click on **Table**. Then, click on **Table Wizard....** Select **Choose an existing dataset in this report or a shared dataset**. Ensure that the dataset you created is selected and then click on **Next**.
2. On the **Arrange fields** page under **Available Fields**, select and drag **Name0** and **Caption0** to the **Values** box in the lower-right pane of the **New Table of Matrix** page and click on **Next**.
3. Click on **Next** on the **Layout** page. Select **Generic** on the **Choose style** page and then click on **Finish**.
4. Click on the **Click to add title** space and type `SCCM computers with OS`.
5. Click on the field titles in turn and change `Name0` to `Computer Name` and `Caption0` to `Operating System`. Use the **Run** button to test the report, as shown in the following screenshot:



- Verify that the report executes. Click on the **Design** button and save the report with the **Save** button. Select your preferred custom folder to store the report, as shown in the following screenshot:



This completes the basic report creation steps.

How it works...

The recipe illustrated how to create a dataset using a shared data source created earlier. The steps provided give you the logical process you must follow to create a report using the Report Builder application. The reusability of shared data sources is also illustrated. The remainder of the book will delve deeper into the options you have to create and modify reports.

Preparing for report subscriptions

You have the option to deliver reports you create and publish to your report consumers using e-mail. You must prepare your **SSRS** instance before you get this option. This recipe discusses the steps you must take to prepare the SSRS instance for e-mail subscriptions.

Getting ready

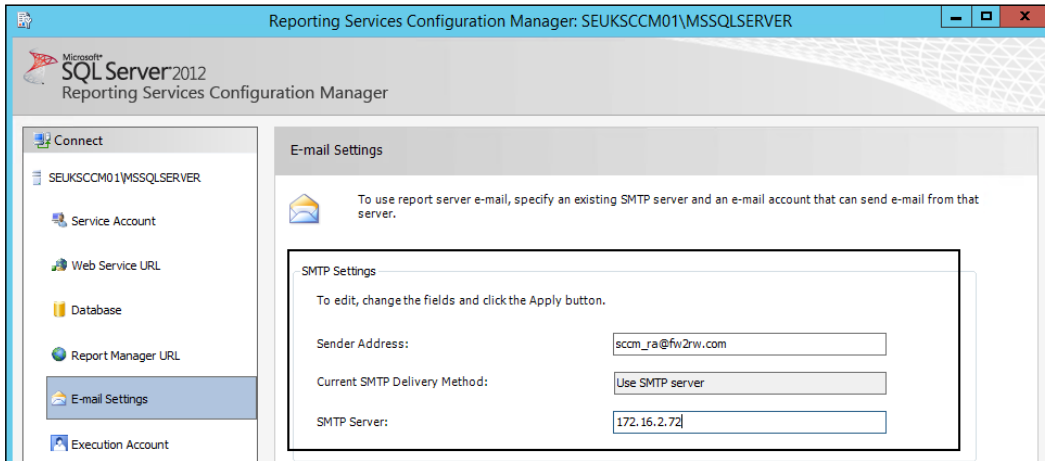
You must have an SSRS deployed and operational to perform these task steps. The user performing these steps must have administrative rights to SSRS. Additionally, you need the details of an SMTP server. The example used in this recipe is based on Microsoft Exchange Server as the SMTP server.

How to do it...

Perform these steps on the SSRS:

1. Launch **Reporting Services Configuration Manager** with the **Run as administrator** option.
2. Connect to the SSRS instance.
3. Select **E-mail Settings** and provide the **Sender Address** and **SMTP Server** details (**Fully Qualified Domain Name (FQDN)** or IP address).

4. Click on **Apply to confirm the settings** and exit to close the **Configuration** page, as shown in the following screenshot:



How it works...

The steps provided prepare your SSRS infrastructure for report delivery using e-mail. You will now see the e-mail option when you select the subscription option in the Report Manager website.

4

Creating Reports for System Center Configuration Manager

In this chapter, we will cover the following recipes:

- ▶ Exploring the Configuration Manager database schema
- ▶ Delegating access to out-of-the-box Configuration Manager reports
- ▶ Preparing the SCCM reporting environment for reporting
- ▶ Creating Configuration Manager custom reports
- ▶ Applying role-based security to custom reports

Introduction

This chapter focuses on understanding and creating the **System Center Configuration Manager (SCCM)** reports. The reporting role for SCCM is optional, but it is recommended as it provides the easiest access to the state of the infrastructure, client information, and status of deployments. The recipes in this chapter use System Center 2012 R2 Configuration Manager.

For all the recipes in this chapter, the requirements are as follows:

- ▶ Deployed System Center 2012 Configuration Manager environment
 - Agents deployed and healthy
 - Hardware inventory successfully performed on deployed agents
- ▶ Reporting services point role should be successfully enabled



The majority of this chapter should apply to SCCM 2012 SP1, but note that 2012 R2 introduced changes, such as role-based security, on reports that the final recipe is based on. This recipe will only be applied to SCCM 2012 R2.

Exploring the Configuration Manager database schema

The SCCM schema is documented and published on the product official documentation website. This recipe compliments the official documentation and provides steps using the console to view and visualize the schema.

Getting ready

You must have access to a System Center Configuration environment with the appropriate role to use the console to explore computer resources. Additionally, you need a SQL Server connection and read-only rights to the SQL Server instance that hosts the SCCM database.

How to do it...

The tasks discussed in this recipe are as follows:

- ▶ System Center Configuration Manager Schema primer
- ▶ Exploring the schema using **Microsoft SQL Server Management Studio (MSSMS)**
- ▶ Exploring the schema using the System Center Configuration Manager console

System Center Configuration Manager (2012 and above) primer

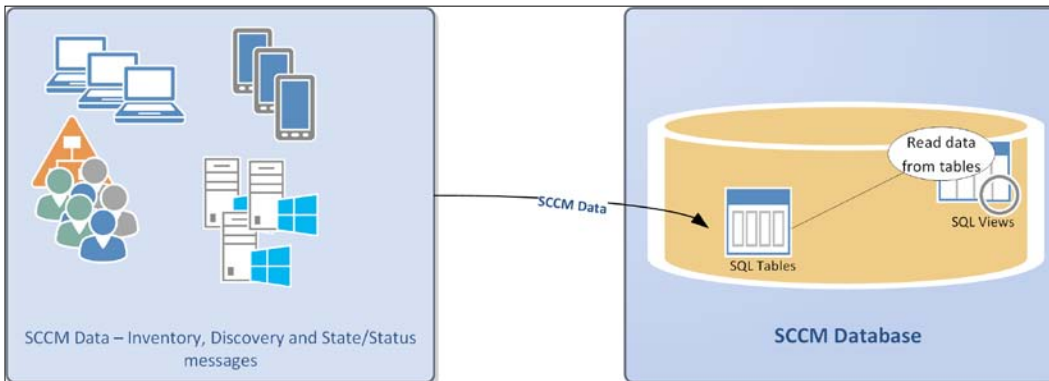
The schemas that we will discuss in this section are the Microsoft SQL Server views used for reports. The overall database schema involves a whole lot more, including the information retrieved from **Windows Management Instrumentation (WMI)** classes and the infrastructure components that make up the SCCM deployment.

When you install SCCM in the SQL Server that hosts the database, two view types are created using the static and dynamic table information. The static view type is what data is known, and it does not change for the specific SCCM environment deployment. The dynamic view types are data types that are created and deleted over the life time of the deployment. Examples of dynamic views type are collections, inventory extensions, and when you enable asset intelligent classes. The following table provides a list of the view categories and their brief description:

View category type	Description
Application Management	This is a view used to create reports on application deployments and associated artefacts, such as packages and programs. It is joined to a collection view to scope the report.
Client Deployment	This is a view category for client deployment and notification status reports.
Client Status	You use the views in this category for client health check and status reports.
Collection	There are two view types in this category. Collection membership views that list members and collection property-only views.
Compliance Settings	These are the compliance settings (formerly DCM) specific views that you use to create compliance state reports.
Content Management	Views in this category are used to create reports on content infrastructure and distribution status.
Discovery	The views in this category are based on discovery for resource types supported by SCCM. This data is from the discovery methods you configure for SCCM.
Endpoint Protection	These are the views for endpoint protection activity reports.
Inventory	Views in this category are based on agent inventory. The information is from the hardware inventory, software inventory, and asset intelligent classes information. You have current information views and historic information views.
Migration	These are the views to report on SCCM migration task activities.
Mobile Device Management	These are the Mobile Device Management (MDM) views for reports. They are based on two categories in MDM: light (Exchange Active Sync) and full (full MDM client deployment).
Network Access Protection	Network Access Protection policy restrictions and errors use these views.
Operating System Deployment	These are the views used for operating system deployment reports. The data scoped in these views include boot image packages and operating system images.
Out of Band Management	These are the Out of Band Management report views.
Power Management	These are the Power Management views used in reports on power plans and power management details when the appropriate policies are configured and deployed to clients.

View category type	Description
Query	There is only one view in this category, that is, v_Query. This view is scoped to all the data on queries used in collections and more.
Schema	The views in this category provide you with the means to write reports on the reporting schema for SCCM.
Security	Views in this category are used for reports on security settings and role-based security configurations.
Site Administration	These are views for reports on site infrastructure.
Software Metering	These are views for Software Metering reports, including metering rules and usage data.
Software Updates	These are views for software update related reports.
Status and Alert	These are views for status updates and alerts. The views in this category are typically combined with other categories to provide status data in reports.
Wake On LAN	This is the view for wake on LAN enabled deployment reports.

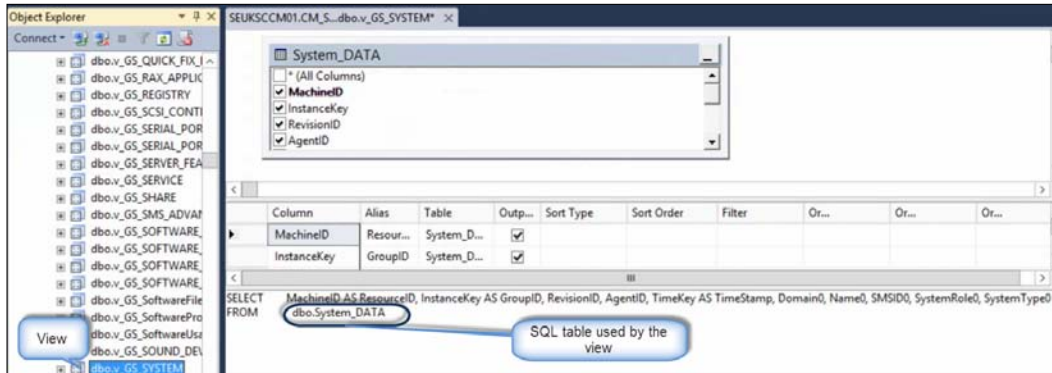
Through discovery, inventory, and state/status messages, the data from the SCCM clients is written to the SCCM database tables. Microsoft creates views using the table data, and you use these views to create your reports. The following figure is a logical representation of the process:



You can see the relationship between views and the underlying tables using Microsoft SQL Management Studio. An example is the computer system information collected by the SCCM agent. Use the following steps to see the relationship:

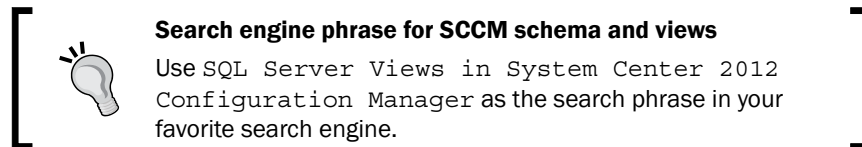
1. Identify the database server and instance of the Configuration Manager site.

2. Use Microsoft SQL Server Management Studio to connect to the database server. You must connect with a user account with the appropriate permission to view the Configuration Manager database.
3. Expand **Databases**, the `CM_<site code>` database, and **Views**.
4. Scroll to the view named **dbo.v.GS_SYSTEM**.
5. Right-click on the view and select **Design**, as shown in the following screenshot:



6. The view's design is presented with the SQL query used and the table targeted in the middle pane.
7. Close the design window, and when prompted to save, click on **No**.

The full details of all the views can be found at <http://technet.microsoft.com/en-us/library/dn581978.aspx>.



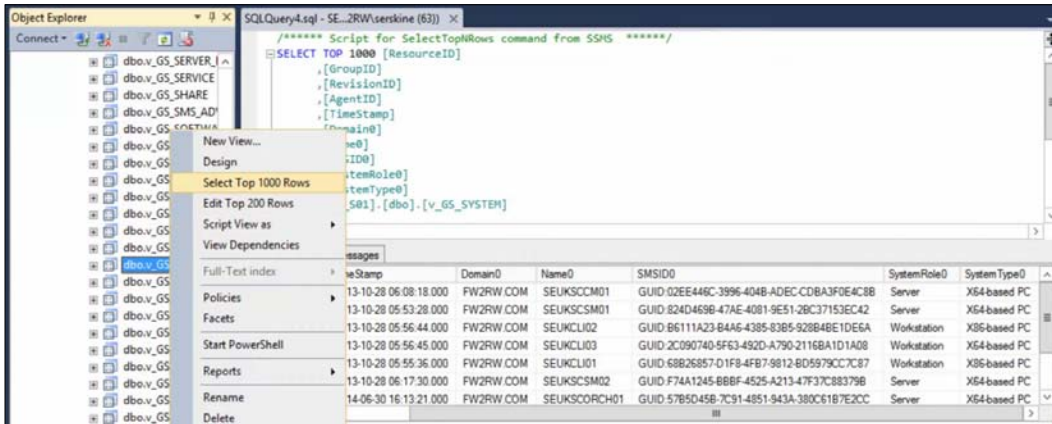
Exploring the reporting schema using MSSMS

You can familiarize yourself with the SCCM reporting views using MSSMS. The following steps use the subcategory of hardware for the Inventory category views:

1. Identify the database server and instance of the Configuration Manager site.
2. Use Microsoft SQL Server Management Studio to connect to the database server. You must connect with a user account with the appropriate permission to view the Configuration Manager database.

Creating Reports for System Center Configuration Manager

- Expand **Databases**, the CM_<site code> database, and **Views**.
- Scroll to the view named **dbo.v_GS_SYSTEM**.
- Right-click on the view and click on **Select Top 1000 Rows**, as shown in the following screenshot:

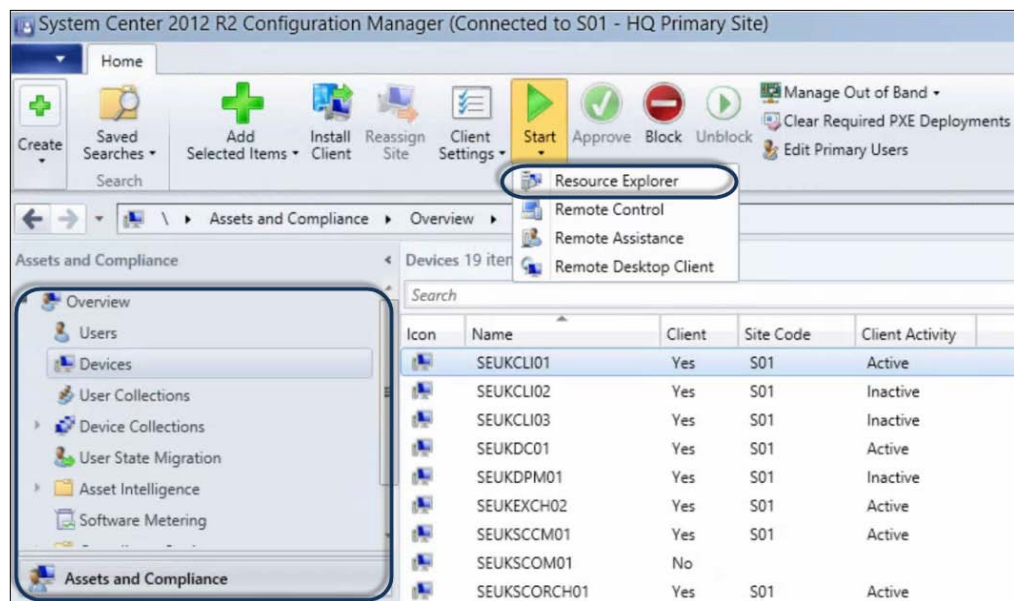


- Review the result of the query executed by the view in the middle pane.
- Close the query window and exit MSSMS.

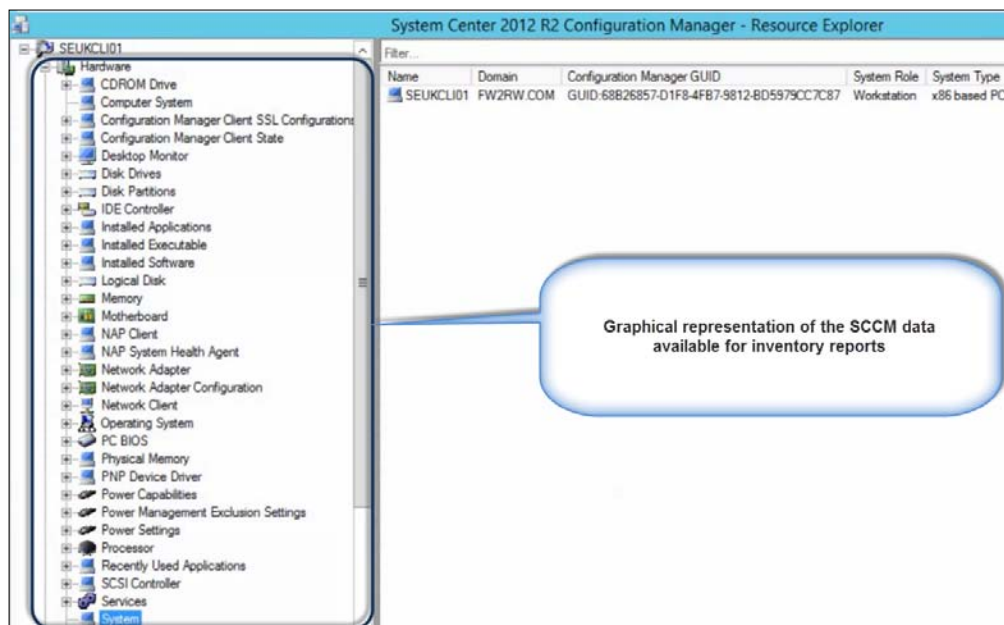
Exploring the reporting schema using the System Center Configuration Manager console

You have the option to view what information is available in the database using the Configuration Manager console. There is a tool called **Resource Explorer** that gives you a quick view of the data. An example is hardware inventory data for a computer. The following steps show hardware inventory data available for reports using Resource Explorer:

- Launch the Configuration Manager console and connect to a site with active computers.
- Select the **Assets and Compliance** node. Expand **Overview** and select **Devices**.
- Select an active client, from the toolbar ribbon, click on **Start**, and then select **Resource Explorer**, as shown in the following screenshot:



4. Expand **Hardware** and select **System**.
5. Review the information available in the middle pane, as shown here:



6. Select other categories and review the information available to you.

How it works...

The quality of the System Center Configuration Manager reports you create depends on a good understanding of how the data is stored and what type of data you can access. This is similar to getting access to the map of a city in order to navigate to the destination of your choice. The recipe is split into three subsections that provide a primer and exploratory examples.

The subsections can be broken down as follows:

- ▶ **System Center Configuration Manager Schema primer:** This is a brief discussion of the SCCM schema as it relates to SQL views for reporting. The section is deliberately brief as the online information available to you is extensive.
- ▶ **Exploring the schema using Microsoft SQL Server Management Studio:** MSSMS is not a natural tool for the SCCM admin, but when it comes to reporting, this is your best tool to explore the data available. You can use select statements (read mode) to retrieve the information you need for reports. This section of the recipe walks you through an example of exploring hardware inventory information available for reports.
- ▶ **Exploring the schema using the System Center Configuration Manager console:** The SCCM console is a graphical window to the information in the database. Almost everything you view in the console can be used as a source for reports. The recipe section uses the Resource Explorer tool to view hardware inventory data available for reports using a computer client as an example.

The SQL views and data available to you for reports are vast. It can be a challenge to think about where to start when planning to create reports. This recipe provides that starting point for SCCM reports.

See also

- ▶ Detailed information on the SCCM database views and reporting can be found in the official online documentation at the following:
 - Microsoft TechNet (<http://technet.microsoft.com/en-us/library/dn581978.aspx>)
 - Microsoft TechNet (<http://technet.microsoft.com/en-us/library/dn581954.aspx>)

Delegating access to out-of-the-box Configuration Manager reports

System Center Configuration Manager has a large set of predefined reports. These reports are available to you when you enable the reporting feature of the product.

This recipe will show you how to grant access to the deployed reports using the console and the Report Manager site.

Getting ready

You must have access to a configured System Center Configuration environment that has the **Reporting Service Point** role successfully enabled. Additionally, the account you use must be member of either the Configuration Manager **Security Administrators** or **Full Administrator** roles.

How to do it...

The steps to create a reporting role for non-administrators use the information in the following table:

Requirement	Information	Description
Active Directory Group for administrative users in SCCM	CM_Report Viewers	This is a global group with users who need access to SCCM reports using the Report Manager website. The group name is an example. Follow your approved naming convention for Active Directory objects.
Report Manager website details	The reporting server URL is <code>http://<SCCM SSRS Instance>/Reports</code>	This is the reporting server running the Report Manager.
SCCM reporting delegation details	Custom-report access only	This is a copy of the Read-only Analyst role scoped to read site objects and run reports.
Report Manager Security role	ConfigMgr report users	This is the SSRS reporting role assignment.

Active Directory group for reporting role

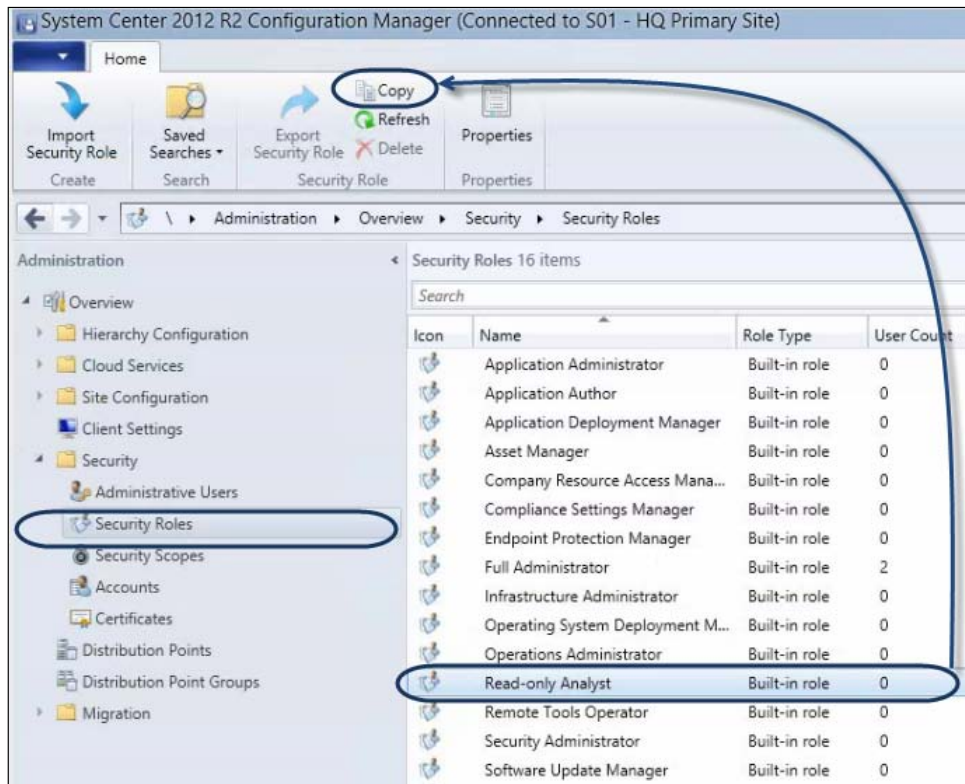
In Active Directory, users and computers create a domain global group called **CM Report Viewers**. Use a universal group for environments where the users reside in a child domain that is different from the SCCM installation.

Add users that should have access to System Center 2012 Configuration Manager reports to this group.

The next steps must be performed in the SCCM console using an account with either the **Full Administrator** or **Security Administrators** role.

Creating the Report Only security scope

1. Launch the Configuration Manager console and connect to the standalone primary or central administration site.
2. Select the **Administration** node. Expand **Security** and select **Security Roles**.
3. In the middle pane, select **Read-only Analyst** and click on **Copy** from the ribbon, as shown in the following screenshot:



4. Type a name and description, for example, Custom - Report Access Only for the name and Reporting Only Security Role. Combine this role with other roles to grant users additional access for the description.
5. Use the details in the following table to complete the permissions. Permissions not listed should be set to **No** under the respective category.

Permission category	Details
Alerts	Run report
Antimalware Policy	Run report
Application	Run report
Certificate Profile	Run report
Collection	Read
Communications Provisioning Profile	Run report
Device Drivers	Run report
Firewall Settings	Run report
Inventory Reports	Run report
Migration Job	Run report
Package	Run report
Settings	Run report
Settings for remote connection profile	Run report
Settings for use data and profile management	Run report
Sideload Key	Run report
Site	Read and run report
Software Metering Rule	Run report
Software Updates	Run report
Status Messages	Run report
Task Sequence Package	Run report
Trusted CA Certificate Profile	Run report
User Device Affinities	Run report
Users	Run report
VPN Profile	Run report
Wi-Fi Profile	Run report
Windows CE Device Settings Item	Run report

6. Click on **OK** to complete the new role creation.

The next set of steps to add the Administrative Users will follow.

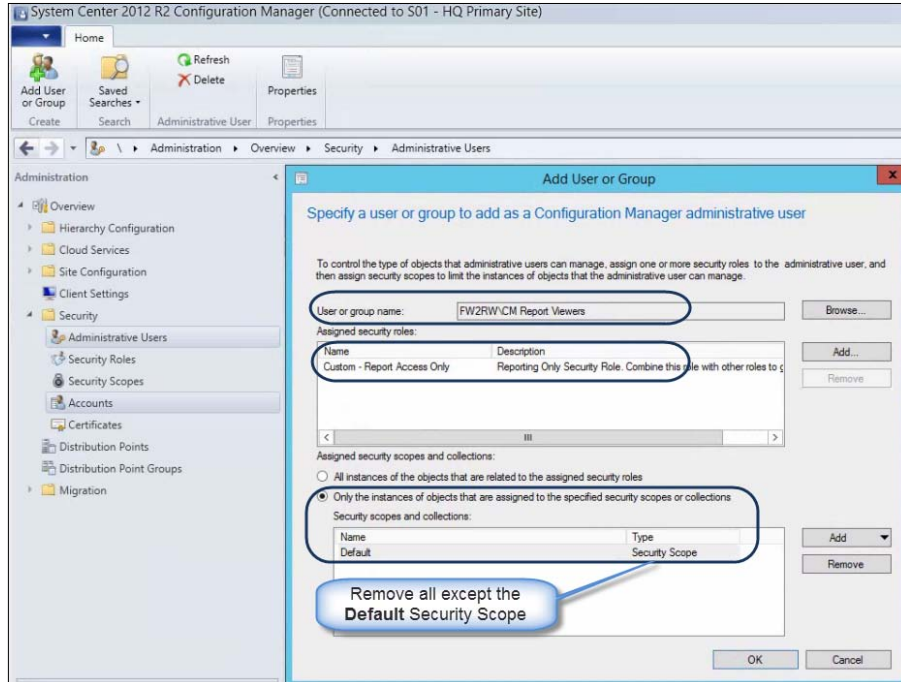
Adding Active Directory user group and configuring the reporting scope

Perform the following steps:

1. Still in the **Administration** node, under **Security**, select **Administrative Users**.
2. Click on **Add User or Group** from the ribbon and configure the required details using the following table:

Setting	Details
User or group name	Browse to the Active Directory group (for example, CM Report Viewers)
Assigned security roles	Add the Custom – Report Access Only role you created in the security scope of this recipe
Assign security scopes and collections	Assign only the instances of objects to the specified security scopes or collections
Security scopes and collections	Remove all, except the default security scope

3. Click on **OK** to complete the administrative user configuration for report viewers, as shown in the following screenshot:



How it works...

The default settings for access to System Center Configuration Manager reports is limited to administrators and a number of feature-related security roles. This recipe provides with the steps to grant users role access purely for report access either using the console or the Report Manager website. The recipe is split into three subsections, and these subsections can be broken down as follows:

- ▶ **Active Directory Group for reporting role:** The recommended practice is to use Active Directory groups for security delegation. You can create the group that will be mapped to the SCCM security role.
- ▶ **Creating the report only security scope:** There is no default security role for reporting only. In this subsection, you can create the default role for security using a copy of the least privileged out-of-the box role (Read-only Analyst). The Read-only Analyst role has the sole purpose of granting view-only access for all objects in the SCCM console. You must remove the read access for categories not required for reporting. The key categories you must configure are Read and Run reports for the Site category and Read for the Collection category. The site option ensures the reporting node is available and the collection category ensures reports that require collection names as parameters can run. You must be delegated access to the collections you select in the reports.
- ▶ **Add the Active Directory user group and configure the reporting scope:** The steps in this subsection cover the final part of the reporting delegation. You can add the administrative user for the reporting view by selecting an Active Directory group or user. You can then select the relevant role, in this case, the custom reporting role, and finally, you can select the scope. The minimum requirement is the default security scope.

There's more...

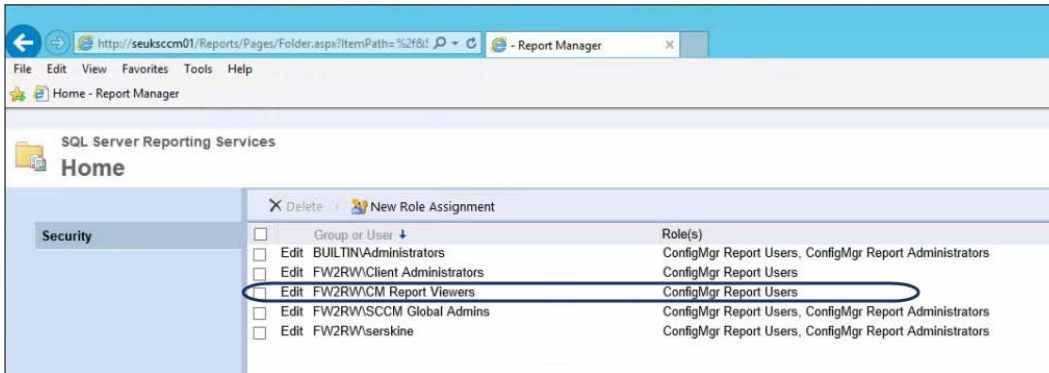
System Center 2012 Configuration Manager and the preceding features integrate seamlessly with the SQL Server Reporting Services instance that hosts the site's reporting point role. You can verify the delegation in the Report Manager website.

Verifying the delegation in the Report Manager website

Perform the following steps to view and verify the delegation in the report manager website:

1. Launch the Report Manager website by navigating to `http://<ServerName>/Reports` (substitute the URL with your specific environment values).
2. On the Report Manager homepage, click on **Folder Settings** to launch the root folder security properties page.

- The Report Manager root folder properties page is presented. Verify that **CM Report Viewer** is assigned the **ConfigMgr Report Users** role. Also, notice that there is a higher role called **ConfigMgr Administrators**, as shown in the following screenshot:



The two roles are automatically created and managed by SCCM. When you delete the administrative user with reporting rights in the SCCM console, the user rights are also amended in the Report Manager website. Note that there is a delay between modifications, and changes may not appear immediately. The delay is approximately 10 minutes.

See also

- ▶ The *Organizing the reporting environment and delegating access to reports* recipe in *Chapter 3, Unpacking System Center Report Building Tools*, provides additional information to compliment this recipe
- ▶ The authors acknowledge two key blog post contributions to this recipe, which are as follows:
 - Coretech blog from Kent Agerlund at <http://blog.coretech.dk/kea/creating-the-reporting-user-role-in-configmgr-2012/>
 - Blog from James Mymryk at <http://www.systemcentercentral.com/sccm-2012reporting-in-console-for-non-admins-reporting-user-role-v2/>

Preparing the SCCM reporting environment for reporting

"I will do that later" and "I will remember" are two examples of how not to prepare and stay organized. The alternative is to "Just do it" when it comes to organizing your environment for reporting. This recipe will save you time, so just do it!

Getting ready

You must have access to a configured System Center Configuration environment that has the **Reporting Service Point** role successfully enabled. Additionally, the account you use must be a member of either the Configuration Manager **Security Administrators** or **Full Administrator** role.

You must plan to review *Chapter 3, Unpacking System Center Report Building Tools*, as a primer to this recipe.

You must also review and optionally perform the delegation steps discussed in the *Delegating access to out-of-box Configuration Manager reports* recipe.

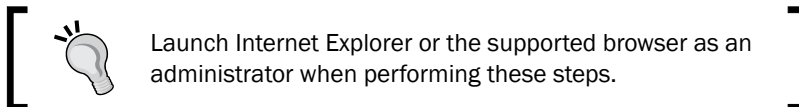
How to do it...

The tasks discussed in this recipe are as follows:

- ▶ Create custom report organizational folders
- ▶ Create a shared data source for SCCM reports

Creating SCCM Report Manager folders and performing delegation

The first recommended action you must perform to organize your reports is create a folder structure in Report Manager. You must connect to the SCCM SSRS instance to perform this task. *Chapter 3, Unpacking System Center Report Building Tools*, provides the steps you need to follow to create folders in Report Manager.



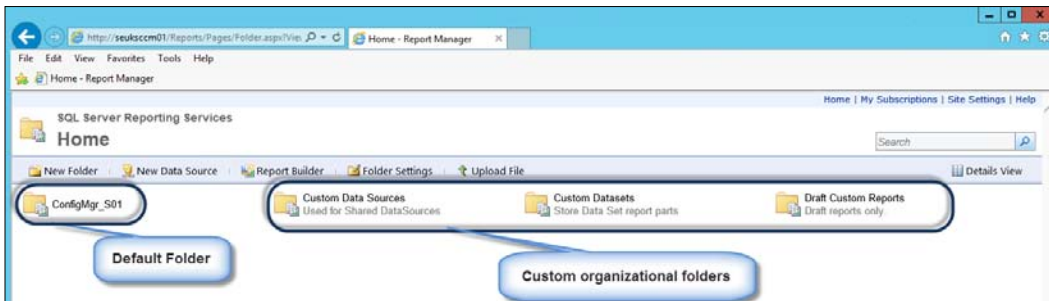
Use the information in the following table to create your initial folders to organize the SCCM reports and supporting objects:

Folder name	Location	Description
Custom Data Sources	Root of the Report Manager website (Home)	Create this folder to store custom shared data sources. The location is typically <code>http://<SCCM SSRS Instance>/Reports</code>

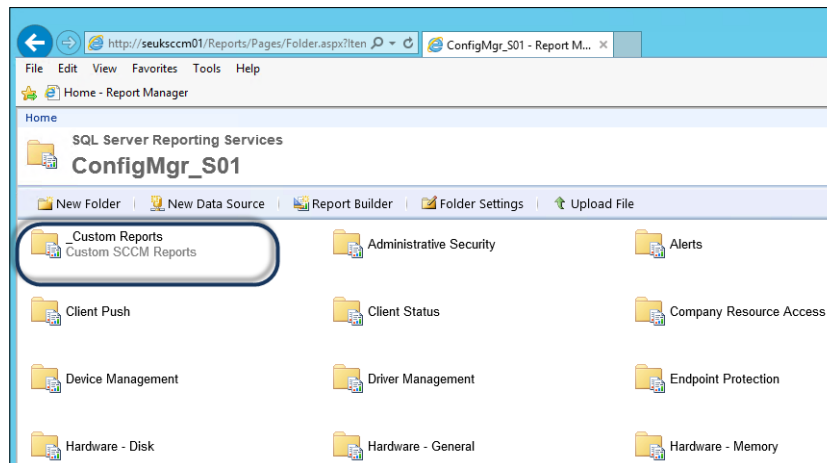
Creating Reports for System Center Configuration Manager

Folder name	Location	Description
Custom Reports	The ConfigMgr<SiteCode> folder	Create the folder under the default folder created for the SCCM reporting point role. Use the _ prefix to put the folder at the top of the list.
Custom Datasets	Root of the Report Manager website (Home)	Store reusable datasets in this folder.
Draft Custom Reports	Root of the Report Manager website (Home)	Use this folder to store reports you are not ready to publish in the configuration manager console.

The illustration that follows shows an example of the structure used in the environment in the root folder for the recipes in this chapter:



The illustration that follows shows an example of the structure used in the environment in the folder created by installing the reporting services point in SCCM:

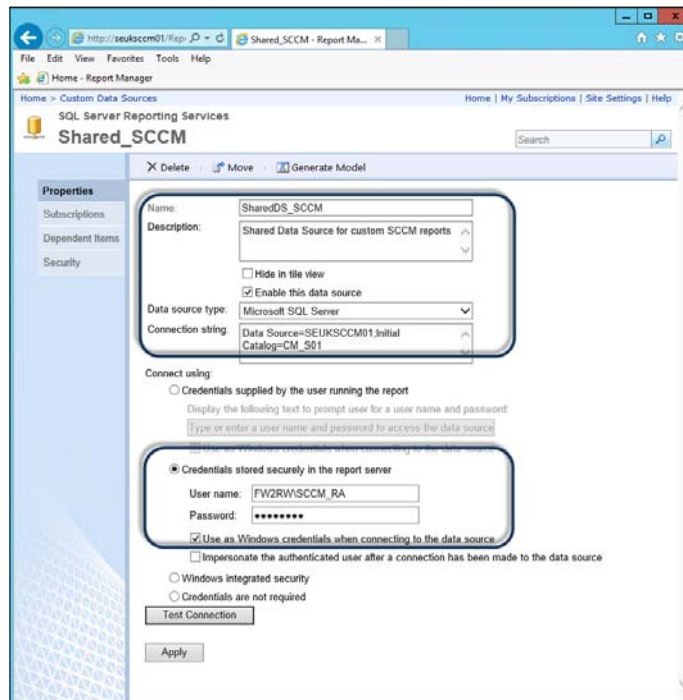


Delegate access to the folders as discussed in *Chapter 3, Unpacking System Center Report Building Tools*.

Creating a shared data source for the SCCM reports

The final preparation step to perform is to create a shared data source. Follow the data source creation steps in *Chapter 3, Unpacking System Center Report Building Tools*. Configure the data source as follows:

- ▶ Data source name: **SharedDS_SCCM**
- ▶ Folder: **Custom Data Sources**
- ▶ Description: **Shared Data Source for custom SCCM reports**
- ▶ Connection string: **Data Source=<SCCM Reporting Server Name>;Initial Catalog=<SCCM Database>**
- ▶ Connect using:
 - ❑ **Credentials stored securely in the report server**
 - ❑ **User name:** and **Password:** as <the report user account you specified for the reporting services point role>
 - ❑ Check **Use as Windows credentials when connecting to the data source**, as shown in the following screenshot:



You are now ready to create your custom SCCM reports.

How it works...

The reports you create depend on a data source and the report author's/consumer's access to this data source. The steps discussed in this recipe will prepare your environment to author reports using the SCCM database as the data source.

The steps are described as follows:

- ▶ **Creating SCCM Report Manager folders and delegating access:** Perform this step to include organization in managing reports from the onset. This prevents creating reports and its dependents, such as data sources in a disorganized environment. You can also create a folder under the default folder for the SCCM reporting point installation. This step ensures that reports are visible in the console and also separates the reports you create from the hundreds of default reports provided by SCCM.
- ▶ **Creating a shared data source:** A shared data source is a recommended practice for report authoring when you intend to create multiple reports from the same database.

When the SCCM environment deployed successfully, it will include a data source and a default folder. The steps in this recipe are recommended as they provide you with organization and control outside of the default configuration.

See also

- ▶ You can find additional information on the recipes discussed in this chapter, in *Chapter 3, Unpacking System Center Report Building Tools*.

Creating Configuration Manager custom reports

This recipe will show you how to use Report Builder to create a report using SCCM data. The reports that you create in this recipe are based on the following business requirements:

- ▶ The SCCM environment must document the use of collections and ensure that their use does not introduce policy conflicts and impact operational performance. The business requires visibility of computers assigned to multiple collections to ensure that policies do not overlap.

Getting ready

You must plan to review *Chapter 3, Unpacking System Center Report Building Tools*, as a primer to this recipe. You must have access to a configured System Center Configuration environment that has the **Reporting Service Point** role successfully enabled. Additionally, the account you use must be a member of either the Configuration Manager **Full Administrator** role or a role with delegated access to create reports.

The following table lists the details required to complete this recipe:

Requirement or information	Description
Data source name	SharedDS_SCCM. Follow the steps provided in the <i>Preparing the SCCM reporting environment</i> recipe.
The reporting server URL is <code>http://<SSRS Instance>/Reports</code>	Reporting server running the Report Manager for the Configuration Manager site
The List_Collections dataset query, <pre>SELECT SiteID AS CollectionID ,CollectionName ,CollectionComment AS Description FROM v_Collections Where IsBuiltIn=0</pre>	The query for the List_Collections dataset.
The CollsForComputers dataset query, <pre>SELECT v_R_System.Name0 As [Computer Name] ,v_Collection.Name As [Collection Name] FROM v_FullCollectionMembership JOIN v_R_System ON v_FullCollectionMembership.ResourceID = v_R_System.ResourceID JOIN v_Collection ON v_FullCollectionMembership.CollectionID = v_Collection.CollectionID</pre>	The query to get computer collection membership.
Draft custom reports	Folder for customer reports

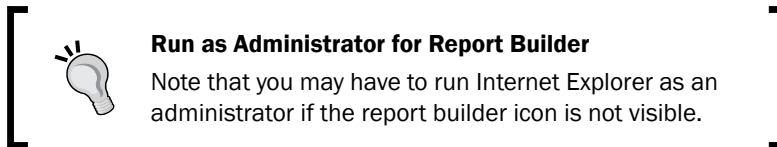
The table will serve as an input for the tasks required to complete this recipe.

How to do it...

The following sections will show you how to configure the various steps to create the report for this recipe.

Creating the data source and datasets

1. To create the data source and data sets, launch the Report Builder application using the Report Manager website option by navigating to `http://<ServerName>/Reports` (substitute the URL with your specific environment values) and click on **Report Builder**.



2. Close the **Getting Started** page. Ensure that the **Home** tab is selected. Under **Report Data**, click on **New** and then select **Data Source**.
3. On the **General** tab, in the **Name:** field, type `SCCMDB` and ensure that **Use a share connection or report model** is selected.
4. Select the shared data source you created in the previous recipe and click on **OK**.
5. Under **Report Data**, click on **New** and select **Dataset...**
6. On the **Query** tab, in the **Name:** field, type `List_Collections` and select **Use a dataset embedded in my report**.
7. Under **Data source:** select the `SCCMDB` data source you created.
8. In **Query type:**, ensure that **Text** is selected and type the **List_Collections** query from the preparation table into the text box, as shown in the following screenshot:

Dataset Properties

Choose a data source and create a query.

Name:
List_Collections

Use a shared dataset.
 Use a dataset embedded in my report.

Data source:
SCCMDB New...

Query type:
 Text Table Stored Procedure

Query:
SELECT SiteID AS CollectionID,
CollectionName , CollectionComment AS Description
FROM v_Collections
Where IsBuiltIn=0

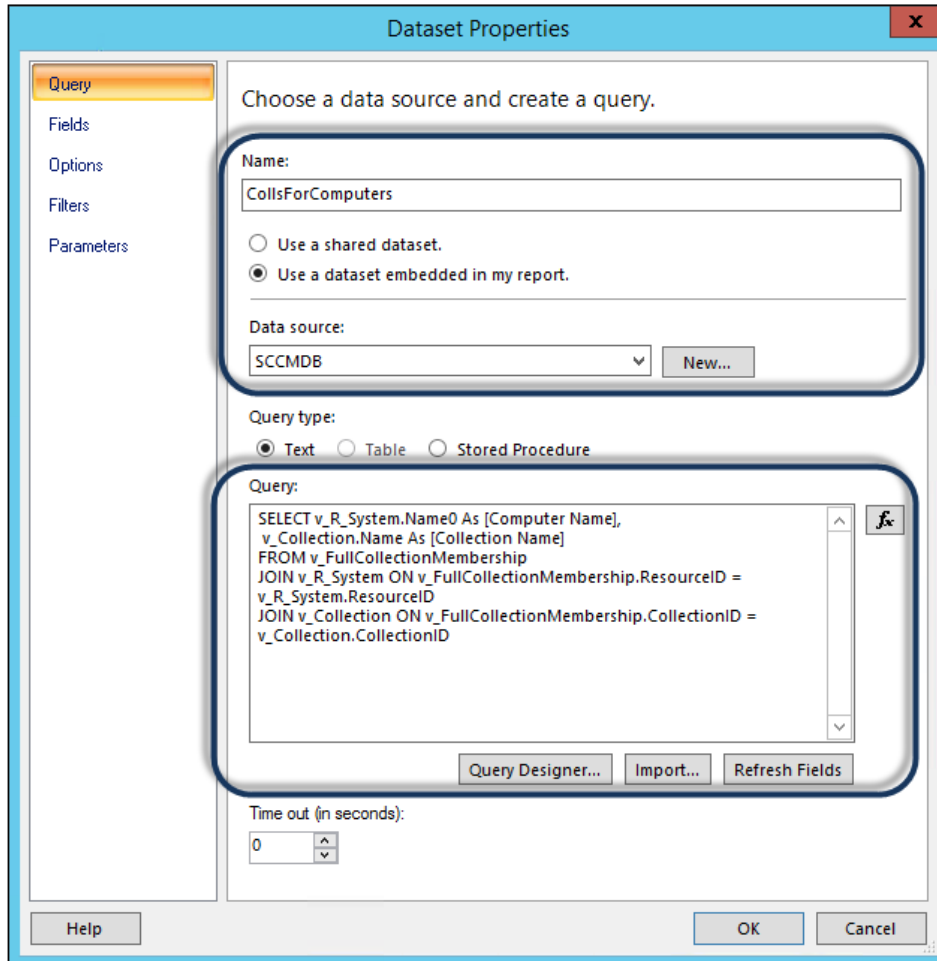
Query Designer... Import... Refresh Fields

Time out (in seconds):
0

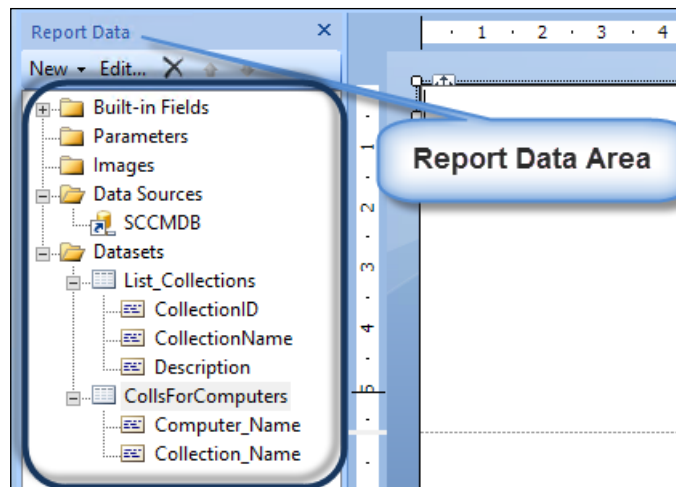
Help OK Cancel

9. Click on **OK**.
10. Verify the dataset called **List_Collection** in the Report Data section.

- Repeat steps 5 to 9 to create a second dataset called **CollsForComputers**. Use the query from the table in the *Getting Ready* section, as shown in the following screenshot:



- Use the **Report Data** area on the left-hand side to verify that **Data Source** and **Datasets** have been successfully created, as shown in the following screenshot:

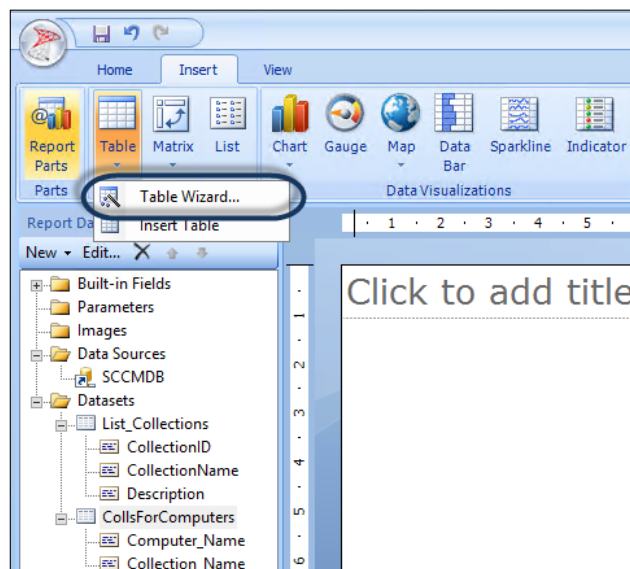


13. Leave **Report Builder** open and proceed to the *Creating the report* section.

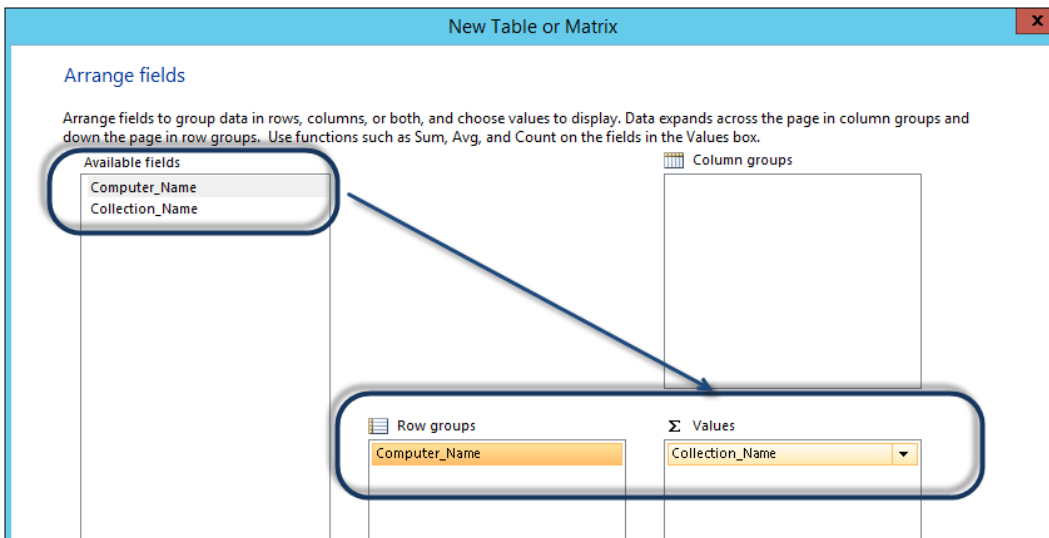
Creating the report

The next set of steps create a report using the datasets you created. The report will list all the computer devices, with the ability to expand to show the computer's collection membership. Perform the following steps:

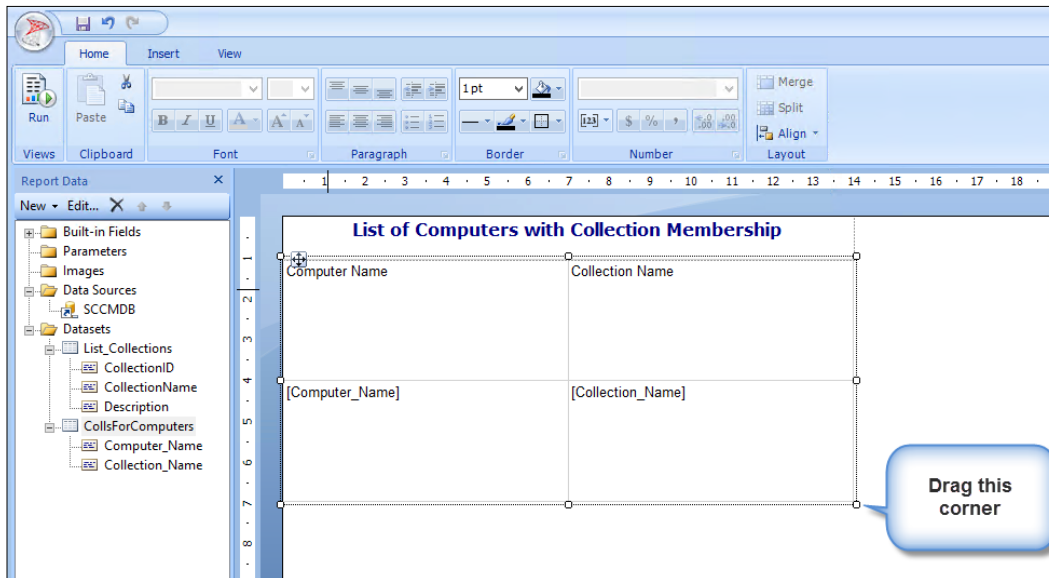
1. Click on the **Insert** tab and select **Table**. Then, click on **Table Wizard**, as shown in the following screenshot:



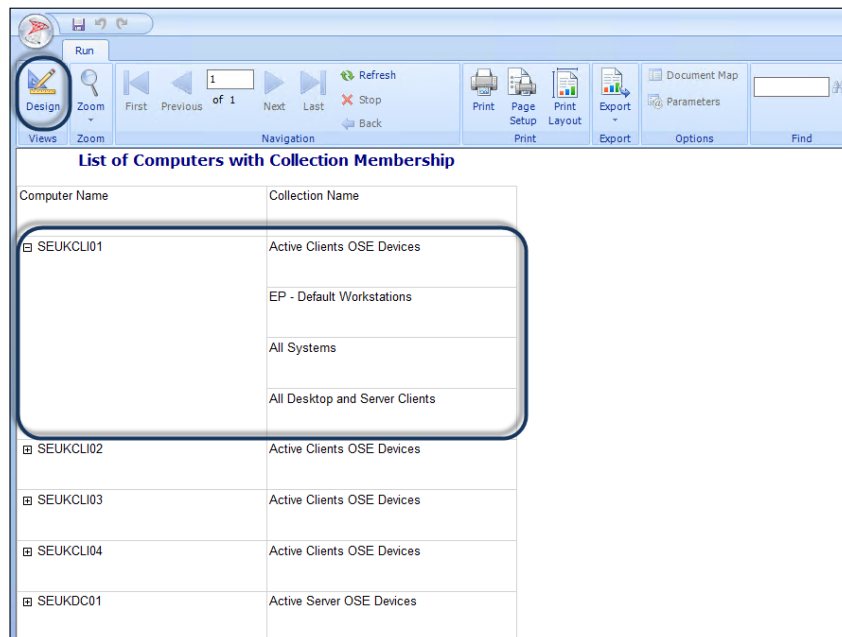
2. Select **CollsForComputers** under **Choose dataset** and click on **Next**.
3. Select and drag **Collection_Name** from **Available fields** to the **Values** box.
4. Select and drag the **Computer_Name** from **Available fields** to **Row groups**, as shown in the following screenshot:



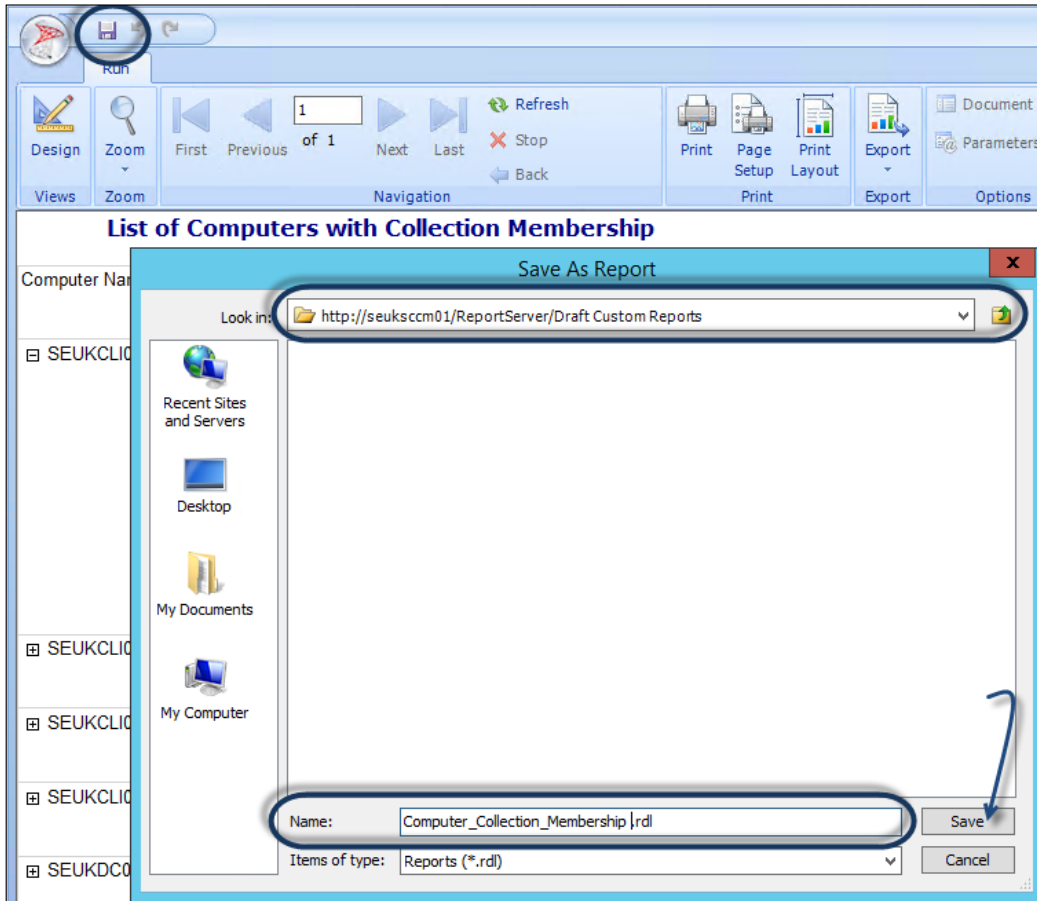
5. Click on **Next**, uncheck **Show subtotals and grand total**, and then click on **Next** on the **Choose the layout** page.
6. Select **Generic** on the **Choose a style** page and click on **Finish**.
7. Click on the title field in the middle pane and type a name for the report, for example, `List of Computers with Collection Membership`.
8. When the whole table is selected, use the lower-right corner to stretch out the layout, as shown in the following screenshot:



9. Click on the **Run** button to test the report. Click on **+** to expand and show the collection membership of a computer.
10. Click on the **Design** button to go back and edit the layout and column sizes, as shown in the following screenshot:

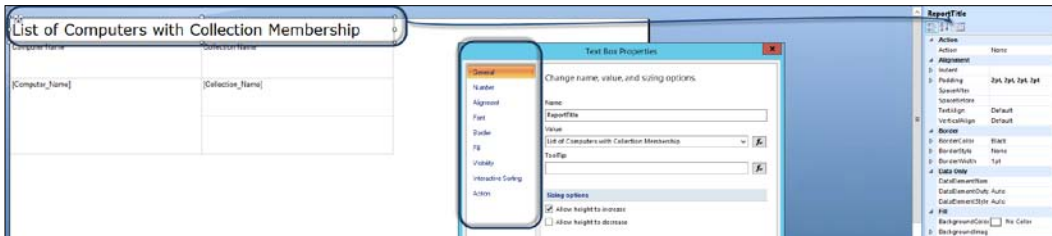


11. Click on the **Save** icon, navigate to the required folder in Report Manager, provide a name, for example, `Computer_Collection_Membership`, and click on **Save**, as shown in the following screenshot:

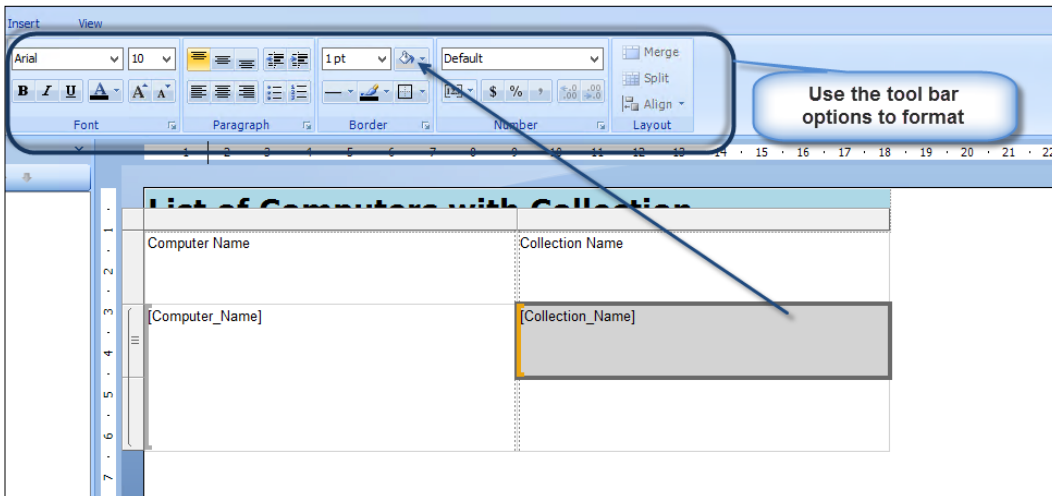


The new report is saved to the SSRS server. You can run and edit the report by connecting to the Report Manager website.

12. Click on any space in the title field and press the **Esc** key to select **ReportTitle**. On the right-hand side of the report design interface (the **Properties** area), select the icon below **ReportTitle** to show the text box properties, as shown in the following screenshot:



13. Change the fill color to, for example, light blue, and make the font bold. Click on **OK**.
14. Click on the space next to **Collection Name** in the second row and use the format options in the toolbar to change the fill color to gray, as shown in the following screenshot:



15. Click on the **Save** icon to update the report.

How it works...

The recipe shows you how to create a report using the dataset created by connecting to a System Center Configuration Manager database data source. The steps demonstrate the connections between the required prerequisites, data source creation, and dataset queries in order to create the report.

The datasets are SQL queries that allow you to create the list of fields you use in your reports. The layout of the report depends on the type you select and the grouping options you plan to have. In the recipe example, the query (dataset) returns a list of computers and their collection membership.

When the values field is used, it behaves like a standard flat table where all the selected fields you use become column headings. The data is dynamically populated at the time the report is run. The report is a definition of what you need to visualize.

The **Row** group field provides grouping and allows you to expand it to show the list of values for each row selection.

The Report Builder interface provides you with the formatting tools and wizards to transform the data into your desired visual representation. This visualization and formatting is similar to what you would perform in a word processor program; you can test out the various formatting options by toggling between the run and design modes.

The final part of the recipe discuss how to save the report. Saved reports can be edited, downloaded, and even saved with a different name to create a new version of an existing report.

There's more...

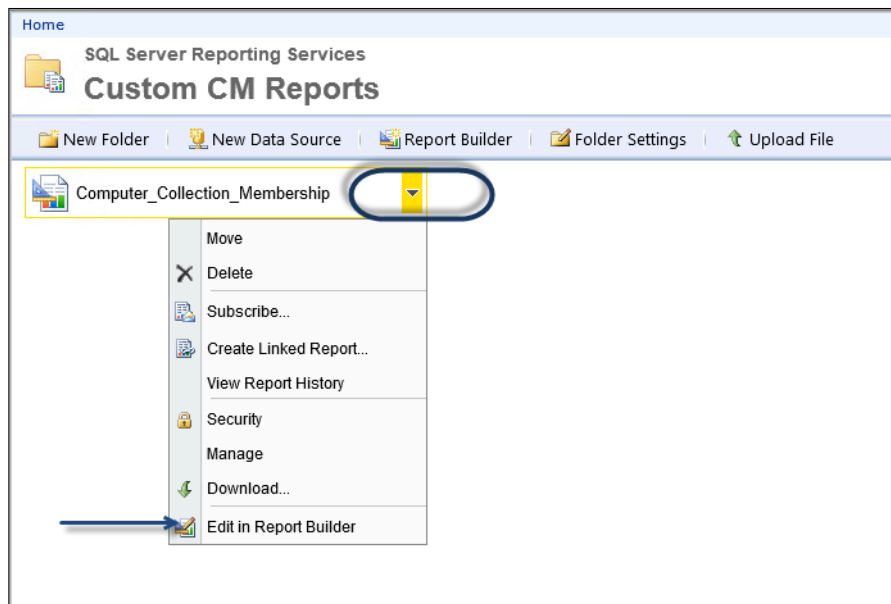
When you create reports, you have other options that do not require the standard report types. You can modify the report with additional options. Here are some examples:

- ▶ Adding a description using a text box and count of collections
- ▶ Inserting your company logo

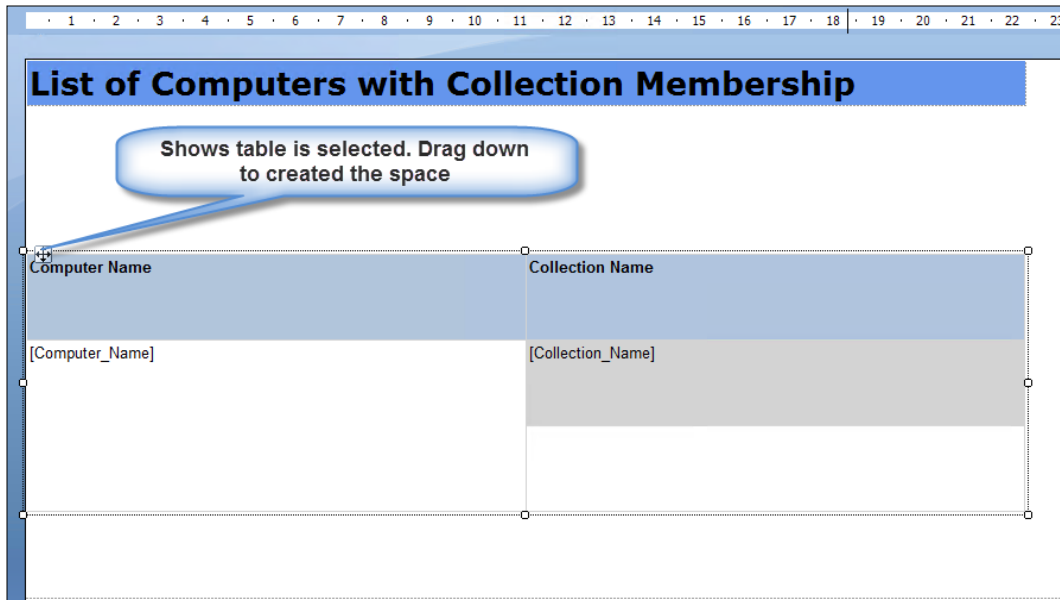
Adding a description using a text box

Perform the following steps:

1. If you closed the Report Builder after the previous step, edit the report by navigating to the Report Manager website. Click on the arrow to the right of the report and select **Edit in Report Builder**, as shown in the following screenshot:



2. Select the table by clicking on the first column and pressing the *Esc* key. With the whole table selected, drag the table object down to create a space between the table and the report title, as shown in the following screenshot:

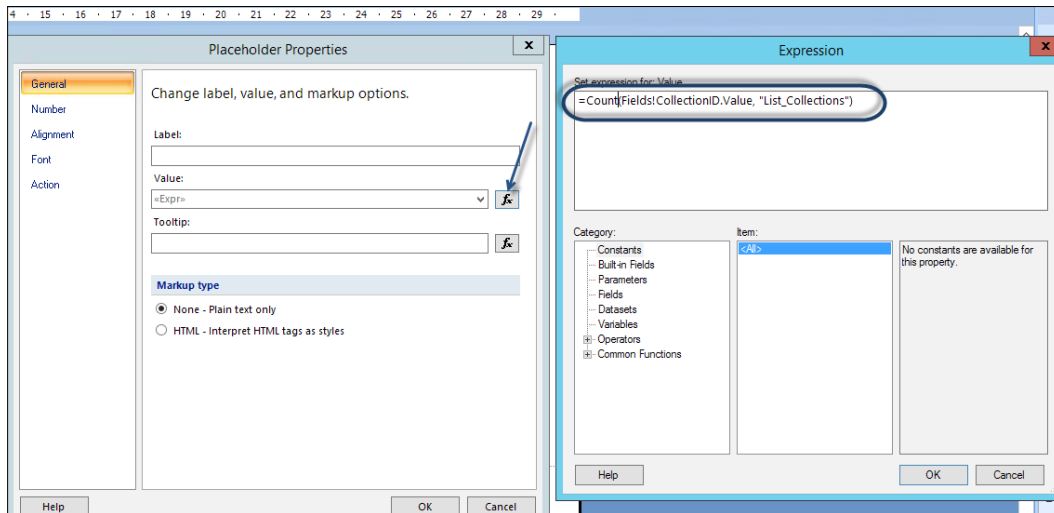
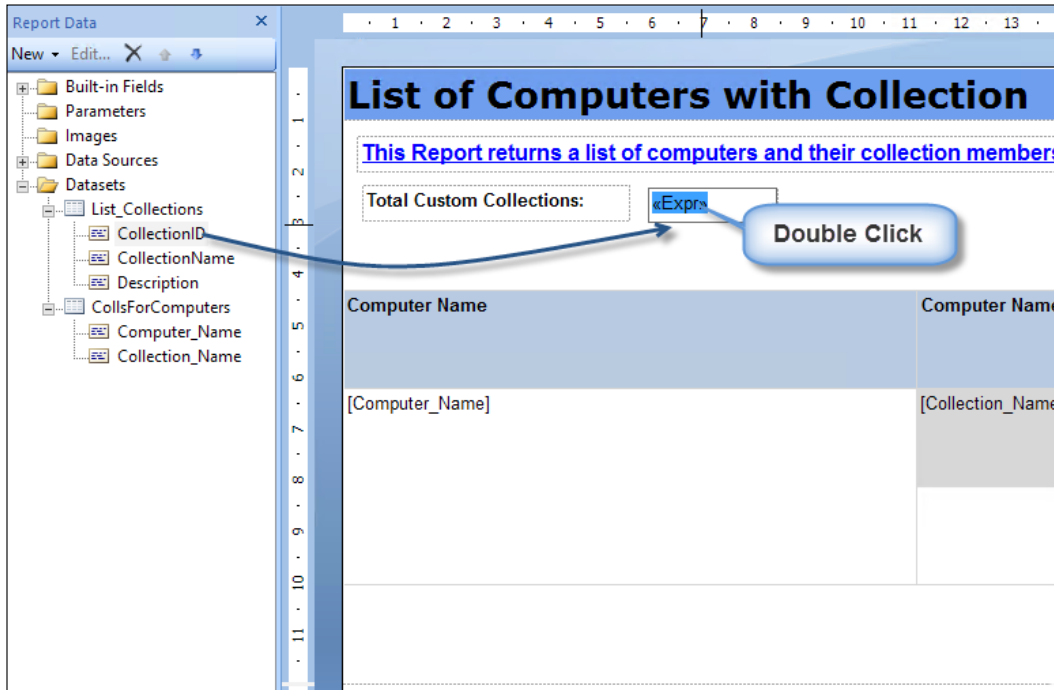


3. Click on the **Insert** tab and then on the **Text Box** icon on the toolbar. Then, drag and draw a box in the space you created in step 2, just below the title.
4. Type a description for the report in the box, for example, `This report returns a list of computers and their collection membership.`
5. Use the formatting options to make the content of the text box stand out. For example, use bold and underline and change the font color to blue (you may need to use the *Esc* key to ensure that the text box property is selected).
6. Click on the **Insert** tab and then on the **Text Box** icon on the toolbar. Then, drag and draw a box in the space below the last text box.
7. Type `Total Custom Collections:` in the text box and format the text as bold, as shown in the following screenshot:

The screenshot displays a report titled "List of Computers with Collection Membership". Below the title, there is a text box containing the description: "This Report returns a list of computers and their collection membership". Below this, another text box contains the text "Total Custom Collections:". At the bottom of the report, there is a table with two columns: "Computer Name" and "Collection Name". The table has a header row with these column names and a data row with placeholders "[Computer_Name]" and "[Collection_Name]".

8. Under **Datasets**, expand **List_Collections** and select and drag **CollectionID** to the space on the right-hand side of the **Total Custom Collections** text box. Be careful not to drop the property on the text box.
9. Double-click on **<<Expr>>** to open up the **Placeholder Properties** box.

10. Click on the **fx** button found to the right of the **Value:** field and replace **First** with **Count** in the **Set expression for: Value** field, as shown in the following screenshot:

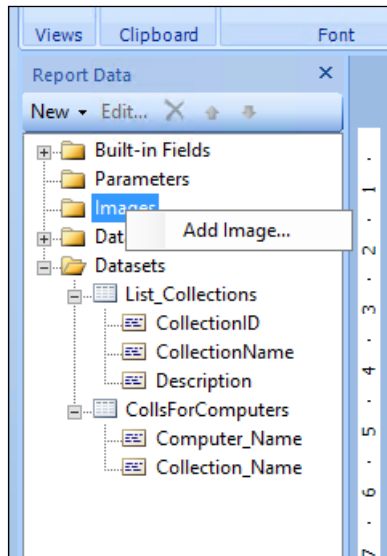


11. Click on **OK** twice and click on **Run** to test the changes. Make additional formatting changes and toggle between **Design** and **Run** until you are satisfied with the format. You may have to drag the text boxes closer to show the total count figure next to the label in the text box.
12. Save the edits to the report.

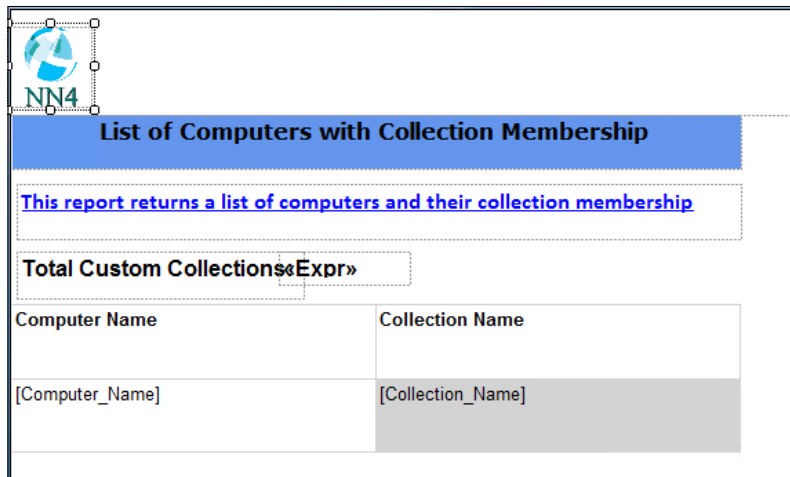
Inserting a company logo

This final part will walk you through adding branding to your report, in this case, a company logo. Perform the following steps:

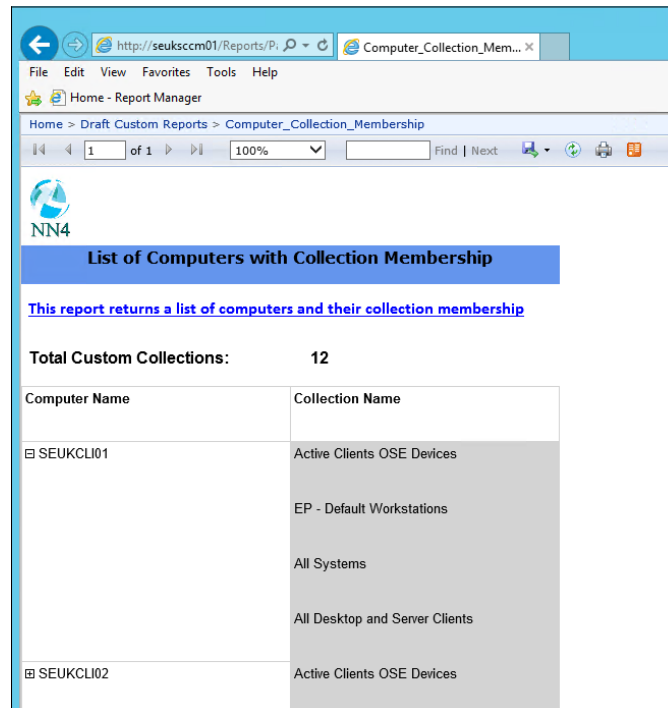
1. If you closed the Report Builder after the previous step, edit the report by navigating to the Report Manager website. Click on the arrow to the right of the report and select **Edit in Report Builder**.
2. Under **Report Data**, right-click on **Images** and select **Add Image...**, as shown in the following screenshot:



3. Navigate to a location on the filesystem where you have your company logo or image for the report, select the image, and click on **Open** to add to the **Report Data** area.
4. Click on the **Insert** tab, click on **Header**, and then select **Add header**.
5. Select the image under **Images** and drag it to the header space, as shown in the following screenshot:



6. Click on **OK** to close the **Image Properties** dialog.
7. Click on **Run** to test the report. Toggle back to edit with the **Design** button and when complete, save the report.
8. Test your report using the Report Manager website, as shown in the following screenshot:



Applying role-based security to custom reports

Microsoft SQL Reporting Services has a security delegation model that you can apply to all the reports that you create. System Center 2012 R2 Configuration Manager introduced role-based security on the default reports. This role-based security model ensures that when running reports, the user can only see objects that they have been granted access to view.

The delegation does not, however, extend to custom reports you create. You must secure these reports using the standard SSRS model. This recipe shows you how you can modify your custom reports to comply with and use the new security model introduced with the 2012 R2 version on the Configuration Manager.

Getting ready

You must plan to review *Chapter 3, Unpacking System Center Report Building Tools*, as a primer to this recipe. You must have access to a configured System Center 2012 R2 Configuration Manager environment that has the **Reporting Service Point** role successfully enabled. Additionally, the account you use must be a member of either the Configuration Manager **Full Administrator** role or a role with delegated access to create reports.

You must also complete the *Creating Configuration Manager custom reports* recipe prior to performing the steps in this recipe.

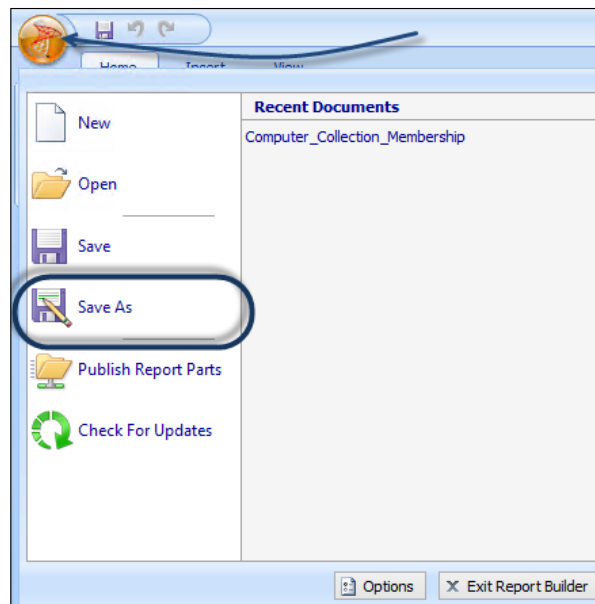


The recipe steps are for environments with the SCCM 2012 R2 version. Versions below R2 do not have the role-based views that the recipe uses.

How to do it...

The following steps will show you how to convert an existing custom report into a Configuration Manager 2012 R2 role-based report:

1. Launch the Report Builder application using the Report Manager website option by navigating to `http://<ServerName>/Reports` (substitute the URL with your specific environment values) and then click on **Report Builder**.
2. Navigate to the **Computer_Collection_Membership** report location in Report Manager, click on the arrow to the right of the report name, and then select **Edit in Report Builder**.
3. Click on the Report Manager icon in the top-left corner (next to the **Save** button) and select **Save As**, as shown in the following screenshot:

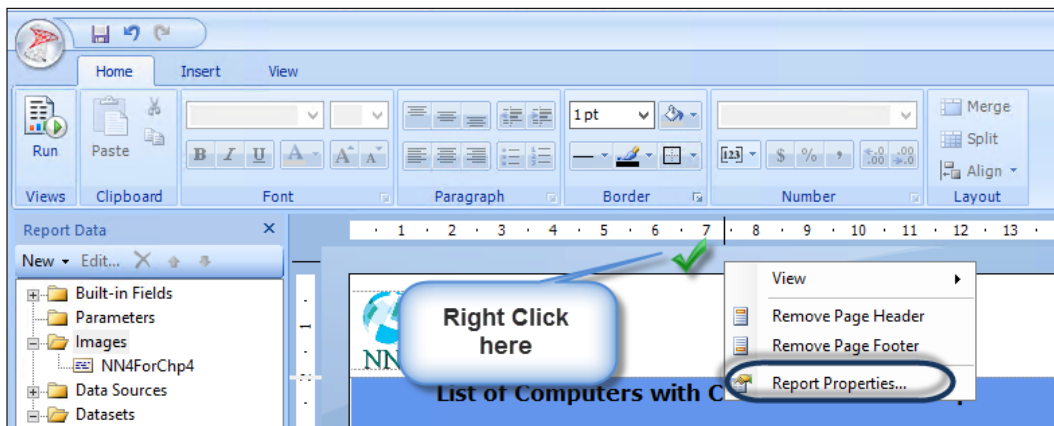


4. Give the report a new name, for example, My_Computers_Collection_Membership.

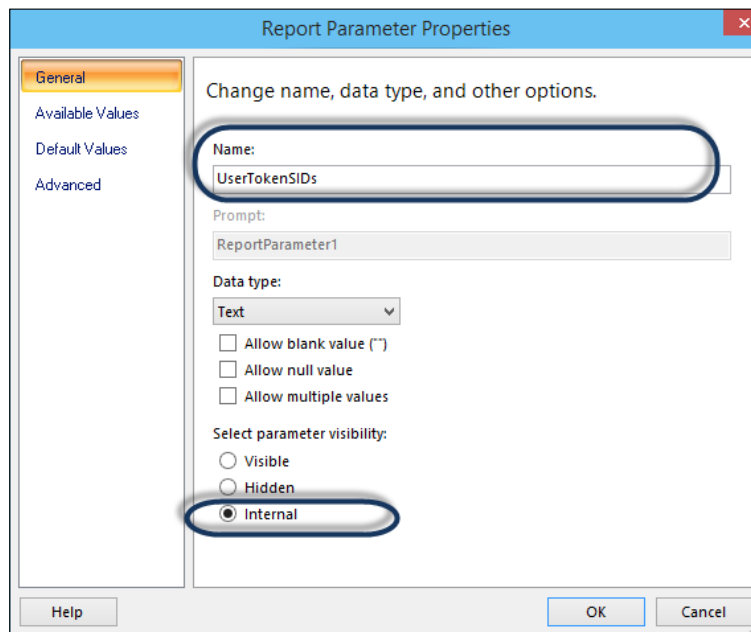
Getting the report user SID as a parameter

Perform the following steps to create the user Active Directory SID parameter:

1. Right-click on the blue space just below the toolbars (this would be below the ruler if it is enabled in the **View** tab of the toolbar), as shown in the following screenshot:

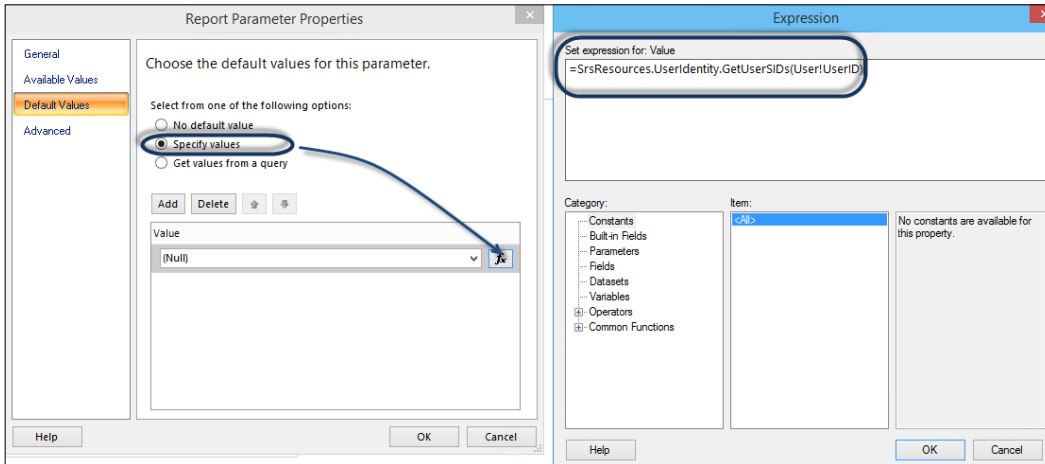


2. Select **Report Properties** and **References** and then click on **Add** below the **Add or remove assemblies:**.
3. Type `SrsResources, culture=neutral`.
4. Click on **OK**.
5. Change the title of the report to something like **My List Computers with Collection Membership**.
6. Change the description to **This report returns a list of computers and their collection membership that is scoped to my access**.
7. Under **Report Data**, right-click on **Parameters** and then select **Add Parameter....**
8. In the **General** tab, type `UserTokenSIDs` and select **Internal** under the **Select parameter visibility:** field, as shown in the following screenshot:



9. Click on the **Default Values** tab, select **Specify values**, and then click on **Add**.

- Click on **fx** to the right under **Value** and type `=SrsResources.UserIdentity.GetUserSIDs (User!UserID)` under **Set expression for: Value**, as shown in the following screenshot:



- Click on **OK** twice.

Creating the Configuration Manager administrative users dataset

Follow these steps to create the administrative users dataset:

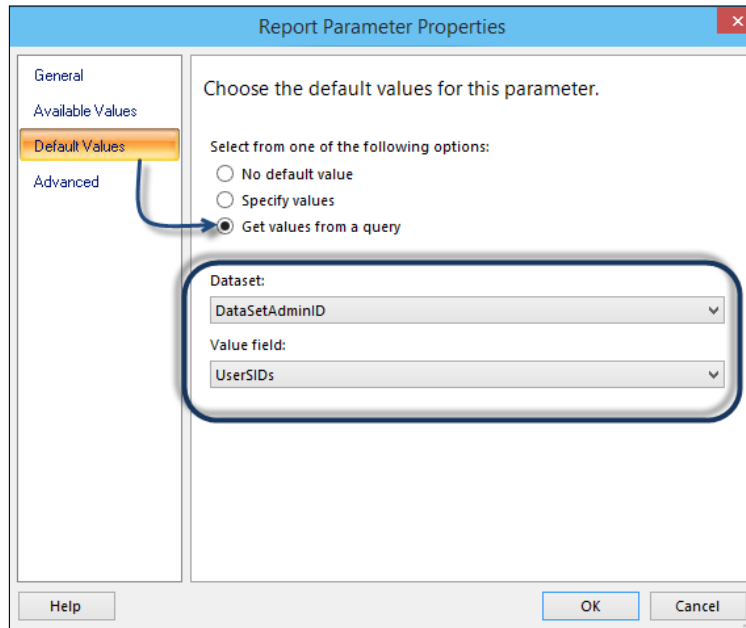
- Under **Report Data**, click on **New** and select **Dataset...**
- On the **Query** tab, in the **Name:** field, type `DataSetAdminID` and select **Use a dataset embedded in my report**.
- Under **Data source:**, select the **SCCMDB** data source you created.
- In the **Query type:** field, ensure that **Text** is selected and type `Select dbo.fn_rbac_GetAdminIDsfromUserSIDs (@UserTokenSIDs) as UserSIDs.`
- Click on **OK**.

Creating the parameter used by the admin dataset

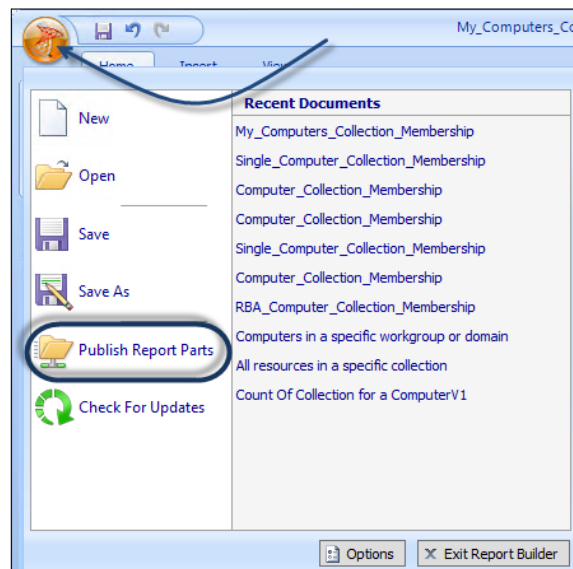
Perform these steps to create the user SID parameter:

- Under **Report Data**, right-click on **Parameters** and select **Add Parameter...**
- In the **General** tab, type `UserSIDs` and select **Internal** under the **Select parameter visibility:** field.
- Click on the **Default Values** tab and select **Get values from query**.
- Under **Dataset:**, select **DataSetAdminID** and then select **UserSIDs** under the **Value field:**

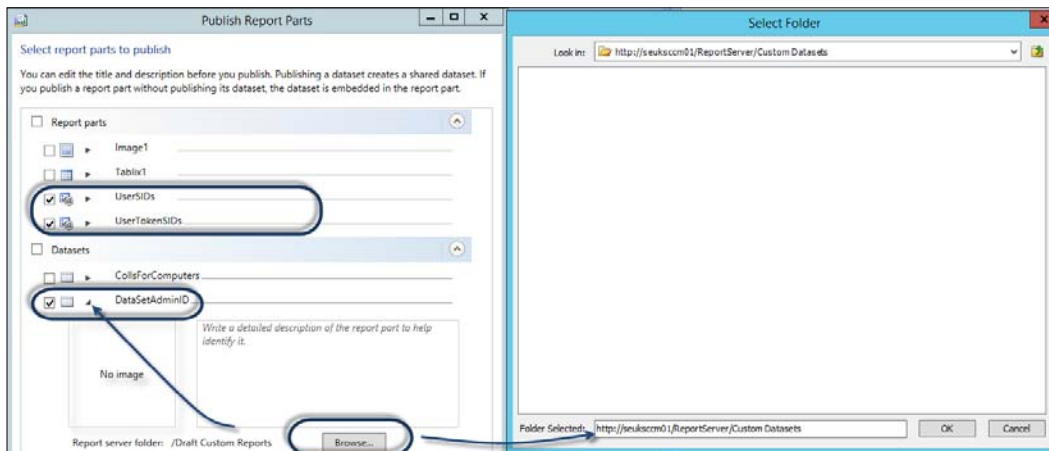
5. Click on **OK**, as shown in the following screenshot:



6. Click on the large Report Builder logo icon and select **Publish Report Parts**, as shown in the following screenshot:



7. Select **Review and modify report parts before publishing**.
8. Uncheck all the options under **Report parts** except **UserSIDs** and **UserTokenSIDs**.
9. Select **DataSetAdminID** and click on the little arrow next to it. Then, browse to the directory you created for custom datasets and click on **OK**, then on **Publish**, and finally, click on **Close**.



Changing the datasets to role-based

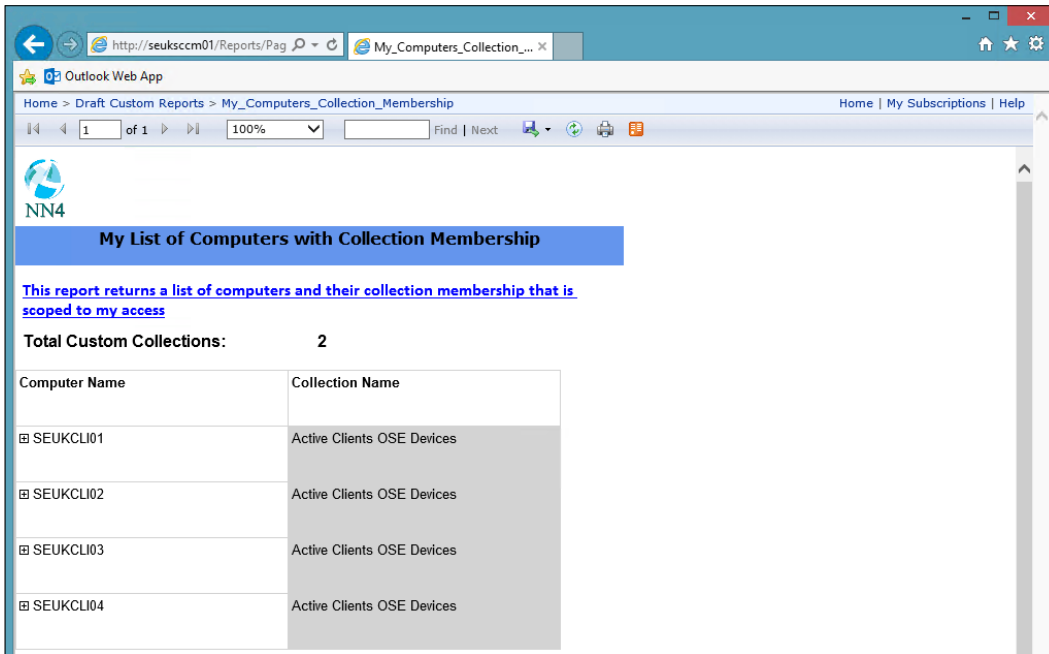
Perform the following steps:

1. Under **Report Data**, right-click on the dataset called **CollsForComputers**. Select **Data Properties**.
2. Change the query to the following:


```
SELECT fnRS.Name0 As [Computer Name],
       fnC.Name As [Collection Name]
FROM fn_rbac_FullCollectionMembership (@UserSIDs) fnFCM
JOIN fn_rbac_R_System (@UserSIDs) fnRS ON fnFCM.ResourceID
= fnRS.ResourceID
JOIN fn_rbac_Collection(@UserSIDs) fnC ON
fnFCM.CollectionID = fnC.CollectionID
```
3. Click on **OK**.
4. Under **Report Data**, right-click on the dataset called **List_Collections**. Select **Data Properties**.

5. Change the query to the following:

```
SELECT SiteID AS CollectionID,
       CollectionName, CollectionComment AS Description
FROM fn_rbac_Collections(@UserSIDs)
Where IsBuiltIn=0
```
6. Click on **OK**.
7. Save the report.
8. Test the report with the **Full Administrator** role user account. The result should list all the collections and computers.
9. Now, test the report with a user account with limited collections. In our example, the user only has access to two collections, and the result is illustrated in the following screenshot:



How it works...

The recipe shows how to modify an existing report to take advantage of the role-based security model available and applied to the default SCCM reports.

The custom reports you create do not automatically inherit the role-based model. The explanation of each section of the recipe is as follows:

- ▶ **Get the report user SID as a parameter:** This first section provides the steps to capture the user's logon token as a dynamic parameter at the time the report is run. This requires an SSRS assembly that you can define in the report properties.
- ▶ **Create the Configuration Manager administrative users dataset:** The second part of the recipe builds a dataset (query) that retrieves the details of the administrative users in the database. This information is what defines the user access to objects in SCCM as defined in the role-based security. The function uses the user token and converts it into the SCCM admin value as the `USERSIDs` value. The `USERSIDs` value is what is expected in the final part of the recipe when you convert the datasets to use the role-based views. The final part of this second section provides you with the steps to save the role-based parameters and the admin datasets using the option to publish report parts.
- ▶ **Change the report datasets to role-based ones:** The final section of the recipe modifies the datasets you use to retrieve the report data to use the role-based function (`fn_rbac`). You can change the leading **V** in the view name to `fn_rbac`, and (`@UserSIDs`) is the parameter that the function expects; you will add this to the end of the view you reference in the datasets.

The default reports leverage the `fn_rbac` views and also define the prerequisites of `fn_rbac` by creating the parameters and the admin dataset. This recipe ensures that you have a process to apply the SCCM role-based security to your reports, based on how you have the delegation and scope applied to administrative users.

See also

- ▶ Detailed information on the SCCM role-based security and reporting can be found in the official online documentation at the following:
 - Microsoft TechNet (http://technet.microsoft.com/en-us/library/7ca322fc-bbbf-42c8-82c9-6fc8941ef2e6#BKMK_RoleBaseAdministration)
 - Microsoft TechNet (<http://technet.microsoft.com/en-us/library/dn581954.aspx>)

5

Creating Reports for SCOM and SCVMM

In this chapter, we will cover the following recipes:

- ▶ Using out-of-the-box Operations Manager reports
- ▶ Understanding the `OperationsManagerDW` schema
- ▶ Preparing your environment to author reports
- ▶ Creating alert reports
- ▶ Creating event reports
- ▶ Creating performance reports
- ▶ Creating availability reports
- ▶ Using Virtual Machine Manager reports

Introduction

This chapter will walk you through the options of using and creating reports for **System Center Operations Manager (SCOM)** and **System Center Virtual Machine Manager (SCVMM)**.

System Center Operations Manager

System Center Operations Manager uses an optional but recommended database for reporting. The reporting database is known as **Data Warehouse**. Operations Manager periodically transfers data stored in the operational database to the Operations Manager Data Warehouse database named `OperationsManagerDW`. The `OperationsManagerDW` database is the database used for all Operations Manager related reports.

System Center Virtual Machine Manager

System Center Virtual Machine Manager does not offer any out-of-the-box reporting capabilities and does not have a dedicated data warehouse. Virtual Machine Manager leverages System Center Operations Manager to provide its reporting capabilities.

Using out-of-the-box Operations Manager reports

In this recipe, we will show you how to navigate through the reporting section of the Operations Manager Console and how to work with reports that are shipped with Operations Manager management packs.

Getting ready

Before you can work with Operations Manager reports, you must have already installed Operations Manager Reporting Server. It can take between 20–30 minutes for the reports to appear in the **Reporting** node in the Operations Manager Console after deploying the reporting server. You need to be a member of the Report Operator Users role in order to browse the reports.

Creating a folder for custom reports

It is recommended that you store all your reports in a custom folder structure in SQL Server Reporting Services. Follow these steps to create a custom folder in the SQL Server Reporting Services used by Operations Manager:

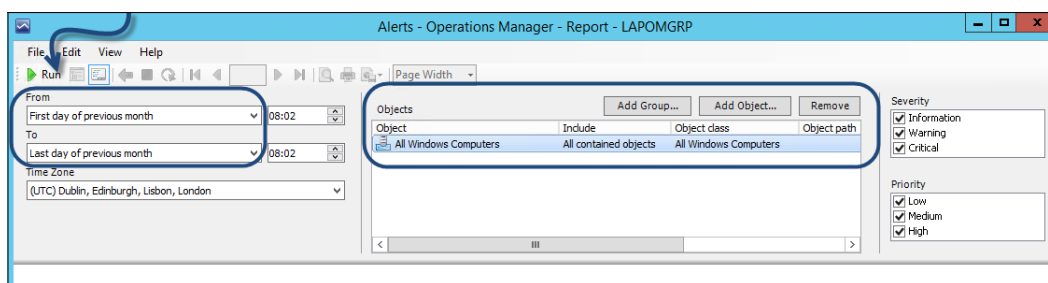
1. Open your web browser and navigate to `http://[SCOMRS]/Reports`. Replace [SCOMRS] with the Fully Qualified Domain Name of the SQL Server Reporting Server used for SCOM. If SQL Server Reporting Service is running as a named instance, the syntax of the URL will be `http://[SCOMRS]/Reports_[InstanceName]`.
2. Click on **New Folder**, enter a name for the folder such as `Custom Reports`, and optionally enter a description. Click on **OK**.

How to do it...

In this section, we will demonstrate how you can access reports from the **Reporting** node in the Operations Manager Console. Perform the following steps:

1. Navigate to the **Reporting** node of the Operations Manager Console.
2. Under the **Reporting** node, you will see folders that contain reports from different management packs. Click on the **Microsoft Generic Report Library** folder.

3. You will see the reports listed in the view. In this example, we want to retrieve a list of all the alerts that concern our domain controllers from the last 30 days. Click on the **Alerts** report and then click on **Open** from the taskbar.
4. In the parameter section, define the filters you would like to apply to the report. For instance, under **From**, click on **Previous month** then click **First day**, and under **To**, click on **Previous month**. Then, click on **Last day**.
5. When running Operations Manager reports, you always have to select the objects or groups of objects you want to include in your report. In this example, we will select the **All Windows Computers** group. Click on **Add Group...** and perform the following steps:
 1. In **Group Name: Contains**, type All Windows Computers.
 2. Click on **Search**. Select **All Windows Computers** under **Available items**.
 3. Click on **Add**. Click on **OK**.
6. Click on **Run** from the toolbar to generate and view the report, as shown in the following screenshot:



7. If you want to change the parameters for your report, click on the **View** menu and then click on **Parameters**.
8. You can export reports to various file formats. Click on the **File** menu. Then, click on **Export** and choose the desired format to export the report.

How it works...

The recipe provided the steps you must follow to run out-of-the-box reports. You can access the reports in Operations Manager Console using a user account with the appropriate rights to perform the actions. The recipe also discussed the options to filter output by specifying criteria. It also discussed the options to save and export reports.

There's more...

You have the option to reuse the criterion selections and reduce the effort required to run reports.

Creating favorite reports and publishing reports

Operations Manager allows you to save the selections you make in the **Parameters** section of the **Report** window for future use. You can either save your settings for just yourself or publish a report that will also be available to other users using the following steps:

1. Open the report of your choice in Operations Manager Console.
2. Make the desired selections in the **Parameters** section of the **Report** window and run the report.
3. In the **File** menu, click on **Save to favorites**, enter a name for the report, and then click on **OK**.
4. The favorite report will now appear under the **Favorite Reports** folder in the **Reporting** node of Operations Manager Console.

A published report is a favorite report that can be accessed by other users. Follow these instructions to create a published report:

1. Open the report of your choice in Operations Manager Console.
2. Make the desired selections in the **Parameters** section of the **Report** window and run the report.
3. In the **File** menu, click on **Publish**, enter a name and description for the report, and then click on **OK**.
4. All users with access to Operations Manager reports can now access the published report by navigating to the **Authored Reports** folder in the **Reporting** section of Operations Manager Console.



Note that you require **Publisher** or **Content Manager** permissions in SQL Server Reporting Services to be able to publish reports.

Scheduling reports

You can schedule Operations Manager reports to be delivered periodically as a shared file or by e-mail. Operations Manager leverages the SQL Server Reporting Services scheduler to provide this functionality. Before you can use scheduled reports, you need to make sure that SQL Server Agent is running on the Operations Manager reporting server.

**E-mail delivery**

Note that **E-Mail** requires additional configuration steps for it to show up as a delivery method when scheduling reports. Connect to the Operations Manager reporting server and start **Reporting Services Configuration Manager**. Connect to your Report Server instance and then, under **E-mail Settings**, enter the required SMTP Settings. Apply the changes.

1. Click on the report of your choice in Operations Manager Console.
2. Click on **Schedule** from the task pane.
3. Enter a description and choose a delivery method (for example, **E-Mail**).
4. Enter the settings for the chosen delivery method.
5. Type the delivery e-mail address in the **To** field.
6. Select the **Render Format** (for example, **PDF**), as shown in the following screenshot:

The screenshot shows a window titled "Subscribe to a Report - Alerts" with a "Delivery Settings" tab selected. The window is divided into a left sidebar and a main content area. The sidebar has three items: "Delivery" (selected), "Schedule", and "Parameters". The main content area has a "Description:" field with the text "Monthly alert overview". Below that is a "Delivery method:" dropdown menu set to "E-Mail". Underneath is a "Settings" section with several fields: "To (required):" with the value "foo@mycompany.com", "Cc:", "Bcc:", and "Reply-To:". There is a checked checkbox for "Include Report". Below that is a "Render Format:" dropdown menu set to "PDF", and a "Priority:" dropdown menu set to "Normal". At the bottom of the window are four buttons: "< Previous", "Next >" (highlighted with a dashed border), "Finish", and "Cancel".

7. Click on **Next** and then set the desired schedule, as shown in the following screenshot:

The screenshot shows a dialog box titled "Subscribe to a Report - Alerts" with a "Subscription Schedule" tab selected. The dialog is divided into three main sections: "Delivery", "Schedule", and "Parameters". The "Schedule" section is active and contains the following options:

- Generate the report:**
 - Radio buttons for frequency: Once, Hourly, Daily, Weekly, Monthly.
 - Checkboxes for months: January, February, March, April, May, June, July, August, September, October, November, December.
 - Option: On the following calendar days [1, 3-5].
 - Option: Every 1st week of the month on the following days:
 - Checkboxes for days: Mon, Tue, Wed, Thu, Fri, Sat, Sun.
- The subscription is effective beginning:** Saturday, May 31, 2014, 3:49 PM.
- The subscription expires on:** Saturday, May 31, 2014.

At the bottom of the dialog, there are four buttons: "< Previous", "Next >", "Finish", and "Cancel".

8. Click on **Next** and then enter the desired parameters for the report to be rendered.
9. Click on **Finish** to save the scheduled report.
10. The scheduled report will now appear under **Scheduled Reports** in the **Reporting** section of Operations Manager Console.

Understanding the OperationsManagerDW schema

The purpose of this recipe is to describe the database schema of the `OperationsManagerDW` database. Knowing the anatomy and the design of this database is the key to successfully creating custom reports for System Center Operations Manager.

Getting ready

The central repository that stores all the data available for Operations Manager reporting is the `OperationsManagerDW` database.

SQL Server Report Builder and other tools used to author reports offer visual wizards to create the queries that will be used to retrieve data from the `OperationsManagerDW` database. You must also have a basic knowledge of the **Structured Query Language (SQL)** used to query relational databases.

How to do it...

All the data relevant for reporting is exposed in views inside the `OperationsManagerDW` database. The recommendation is you should not access the tables directly in your report queries. Instead, it is recommended that you target the database views for your reporting needs. This recommendation is because Microsoft makes every effort to maintain the schema of the views even when structural changes are made to the underlying tables.

In the `OperationsManagerDW` database, data is grouped into different categories, which are also referred to as **datasets**. These datasets include the following:

- ▶ Alert dataset
- ▶ Event dataset
- ▶ Performance dataset
- ▶ State dataset

Additionally, there are more datasets for special monitoring capabilities of Operations Manager, such as the **Application Performance Monitoring (APM)** dataset and **Client Monitoring** dataset.

Managed entities

All data available in the various datasets reference monitored objects, management packs, groups, and so on. This information can be retrieved using the `ManagedEntity` views.

The views available for managed entities are typically joined to other views in the alert, event, and performance, or state datasets. These views resolve information about the monitored object and allow you to scope your query to a specific group, management pack, or management group.

vManagedEntity

The `vManagedEntity` view contains the names of all the objects in your Operations Manager environment, such as computers, SQL databases, and IIS websites. Use the `ManagedEntityRowId` column to join this view to views from other datasets. Run the following query using the `OperationsManagerDW` database to view the details in your environment:

```
SELECT
    ManagedEntityRowId,
    FullName,
    Path
FROM
    vManagedEntity
```

vManagedEntityType

To scope managed entities by a specific type of object, you can join the `vManagedEntityType` view to the `vManagedEntity` view on the `ManagedEntityTypeRowId` column. To retrieve a list of all the managed computers in your Operations Manager environment, you can run the following query using the `OperationsManagerDW` database:

```
SELECT
    ME.ManagedEntityRowId,
    ME.Name
FROM
    vManagedEntity ME
    INNER JOIN vManagedEntityType MET ON
        ME.ManagedEntityTypeRowId = MET.ManagedEntityTypeRowId
WHERE
    MET.ManagedEntityTypeSystemName = 'Microsoft.Windows.Computer'
```

Each entity type in Operations Manager has a set of properties. Each instance of a managed entity type has its own properties. The properties are stored in the `vManagedEntityTypeProperty` view, which can be joined to the `vManagedEntityType` view on the `ManagedEntityTypeRowId` column. If you want to retrieve the value of a specific property of a managed entity, you can use the `vManagedEntityTypePropertySet` view, which stores the values in the `PropertyValue` column. Use the `PropertyGuid` column and join it to the `PropertyGuid` column of the `vManagedEntityTypeProperty` view to retrieve the property name.

Entity types in Operations Manager can also have images (icons and pictures), which you might want to use in your reports. You can retrieve the image by joining `vManagedEntityType` to `vManagedEntityTypeImage` on the `ManagedEntityTypeRowId` column. The binary image data is stored in the `Image` column of the `vManagedEntityTypeImage` view.

To retrieve a list of all managed entities including their 16 x 16 icon and diagram image, you can run the following query using the `OperationsManagerDW` database:

```
SELECT
    ME.ManagedEntityRowId,
    ME.FullName,
    MET.ManagedEntityTypeSystemName,
    METIicon.Image AS [Icon_16x16],
    METIdiagram.Image AS [Icon_Diagram]
FROM
    vManagedEntity ME
    INNER JOIN vManagedEntityType MET ON
        ME.ManagedEntityTypeRowId = MET.ManagedEntityTypeRowId
    INNER JOIN vManagedEntityTypeImage METIicon ON
        MET.ManagedEntityTypeRowId = METIicon.ManagedEntityTypeRowId
        AND METIicon.ImageCategory = 'u16x16Icon'
    INNER JOIN vManagedEntityTypeImage METIdiagram ON
        MET.ManagedEntityTypeRowId =
        METIdiagram.ManagedEntityTypeRowId
        AND METIdiagram.ImageCategory = 'DiagramIcon'
```

vManagementPack

You can use the `vManagementPack` view to filter objects in the `OperationsManagerDW` database by specific management packs. A typical example includes filtering managed entities by entity types that derive from a particular management pack, or you can filter rules and monitors by the management packs they are defined in.

The following query using the `OperationsManagerDW` database returns all the managed entities that are of a managed type, which is defined in the `Microsoft.Windows.Library` management pack:

```
SELECT
    ME.ManagedEntityRowId,
    ME.Name,
    ME.FullName,
    MET.ManagedEntityTypeSystemName
FROM
    vManagedEntity ME
    INNER JOIN vManagedEntityType MET ON
        ME.ManagedEntityTypeRowId = MET.ManagedEntityTypeRowId
    INNER JOIN vManagementPack MP ON
        MET.ManagementPackRowId = MP.ManagementPackRowId
WHERE
    MP.ManagementPackSystemName = 'Microsoft.Windows.Library'
```

vRelationship

You can use the `vRelationship` view to retrieve group membership information for managed entities. The `vRelationship` view contains `SourceManagedEntityRowId` and `TargetManagedEntityRowId`; both properties can be used to join to the `vManagedEntity` view.

You can also use the `vRelationshipType` view and join it to the `vRelationship` view on the `RelationshipTypeRowId` column to retrieve information about the type of relationship between two managed entities.

In the following example, we will retrieve a list of all entity types that relate to an instance of the `Microsoft.Windows.Computer` managed entity type. You can use the sample query from the `vManagedEntityType` section earlier in this recipe to retrieve a list of all the managed entities of the `Microsoft.Windows.Computer` type.

Pick any of these computers and replace the value for `SourceManagedEntityRowId` in the following script with `ManagedEntityRowId` of a computer in your environment:

```
SELECT
    ME.ManagedEntityRowId,
    ME.ManagedEntityDefaultName,
    MET.ManagedEntityTypeDefaultName,
    RT.RelationshipTypeDefaultName
FROM
    vRelationship R
```

```

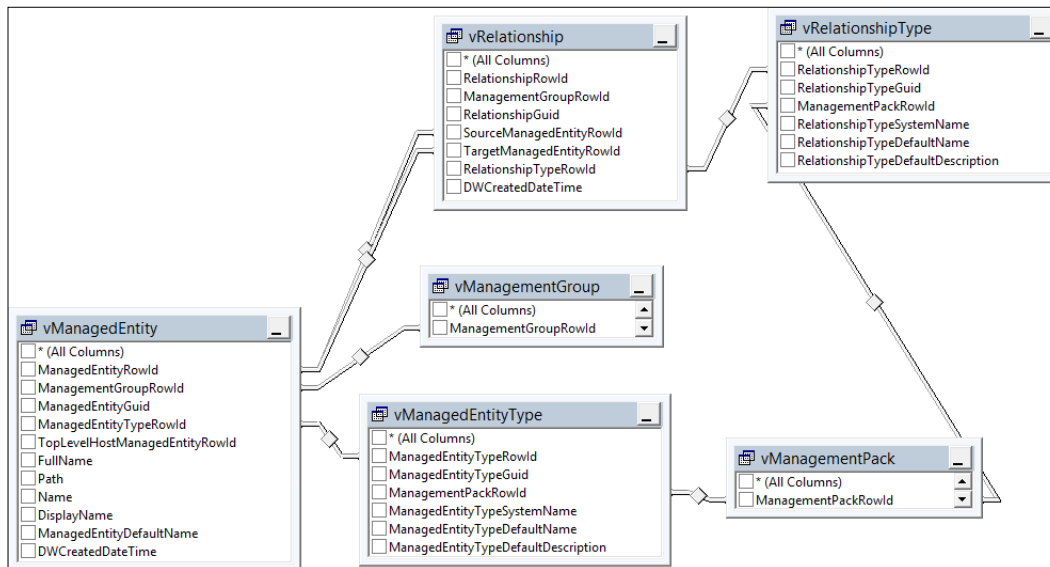
INNER JOIN vRelationshipType RT ON
    R.RelationshipTypeRowId = RT.RelationshipTypeRowId
INNER JOIN vManagedEntity ME ON
    R.TargetManagedEntityRowId = ME.ManagedEntityRowId
INNER JOIN vManagedEntityType MET ON
    ME.ManagedEntityTypeRowId = MET.ManagedEntityTypeRowId
WHERE
    R.SourceManagedEntityRowId = 1299

```

When you run the query, it returns several managed entities that relate to the particular computer, such as network adapters, logical disks, and operating system. Using the `vRelationship` and `vRelationshipType` views can be very helpful to filter the scope of your reports.

vManagementGroup

If your `OperationsManagerDW` database contains data of multiple Operations Manager management groups, it can be helpful to filter data based on managed groups. The `vManagementGroup` view contains all the management groups that report to the `OperationsManagerDW` database. You can join this view to views such as `vManagedEntity` or `vRelationship` on the `ManagementGroupRowId` column to establish a filter based on management groups, as shown in the following screenshot:



How it works...

The schema is similar to the map of a city. Using the city as an analogy, even if you have a map, you need to have the means to travel to your favorite destination. The mode of travel in the schema city is the use of SQL queries. The recipe discusses the schema and provides example queries on how to extract the information you need. The additional points to note in this recipe are the use of `Joins` and `aliases`. The joins in the queries connect views and aliases, provide short names to make the queries easier to read. *Appendix, Useful Websites, Chapter Code, and Community Resources*, provides links to additional resources, including the SQL query syntax.

There's more...

You have additional report data filter options available to you based on your requirements.

Rules and monitors

When authoring reports for Operations Manager, you might want to filter data in your report by rules or monitors. For instance, you might want to retrieve a list of alerts that were raised by a particular rule or monitor or filter performance data by a certain rule.

The `vRule` and `vMonitor` views contain information for all rules and monitors, respectively. You can join the `vRule` view on the `RuleRowId` column and the `vMonitor` view on the `MonitorRowId` column to the views in the different datasets available in the database.

If you want to further filter rules and monitors by the management packs they are defined in, you can establish a join between `vRule` and `vMonitor` to the `vManagementPack` view on the `ManagementPackRowId` column.

You can use the following query to retrieve a list of all the rules defined in Active Directory Server Common Library:

```
SELECT
    R.RuleRowId,
    R.RuleDefaultName
FROM
    vRule R
    INNER JOIN vManagementPack MP ON
        R.ManagementPackRowId = MP.ManagementPackRowId
WHERE
    MP.ManagementPackSystemName =
    'Microsoft.Windows.Server.AD.Library'
```

Preparing your environment to author reports

This recipe provides details on the steps you must take to prepare your environment to author reports.

Getting ready

You must have a deployed Operations Manager environment with the optional reporting components successfully configured. Additionally, you must have the appropriate delegation configured to allow access to the SQL Server Reporting Services instance for the Operations Manager environment.

How to do it...

Creating a shared data source

Before you can start authoring custom reports based on SQL Server Reporting Services, it is recommended that you create a shared data source that you will use for all your reports. It is recommended that you use the SCOM Data Reader account inside your data source, as appropriate permissions have already been granted in SQL servers. Let's take a look at the following steps that illustrate the process of creating a shared data source and the usage of SCOM Data Reader:

1. Open your web browser and navigate to `http://[SCOMRS]/Reports`. Replace [SCOMRS] with the Fully Qualified Domain Name of the SQL Server Reporting Server used for SCOM. If SQL Server Reporting Service is running as a named instance, the syntax of the URL will be `http://[SCOMRS]/Reports_[InstanceName]`.
2. Click on **New Data Source** and complete the form as follows:
 - Name: **OperationsManagerDW**
 - Description: Optionally type a description
 - Data source type: **Microsoft SQL Server**
 - Connection String: `data source=[SCOMSQL];initial catalog=OperationsManagerDW` (replace [SCOMSQL] with the Fully Qualified Domain Name of the SQL Server that hosts the OperationsManagerDW database)

Now perform the following steps:

1. Select **Credentials stored securely in the report server**.
2. Enter the username and password of the SCOM Data Reader account.
3. Select **Use as Windows credentials when connecting to the data source**.
3. Click on **Test Connection** to validate the connection.
4. Click on **OK** to save the new data source, as shown in the following screenshot:

Name: OperationsManagerDW

Description: Shared data source used for report authoring

Hide in tile view

Enable this data source

Data source type: Microsoft SQL Server

Connection string: Data Source=SEUKSCOM01;Initial Catalog=OperationsManagerDW

Connect using:

Credentials supplied by the user running the report

Display the following text to prompt user for a user name and password:
Type or enter a user name and password to access the data source

Use as Windows credentials when connecting to the data source

Credentials stored securely in the report server

User name: FW2RW\OMDWR

Password:

Use as Windows credentials when connecting to the data source

Impersonate the authenticated user after a connection has been made to the data source

Windows integrated security

Credentials are not required

Test Connection

Connection created successfully.

OK Cancel

How it works...

The recipe provides the steps you must perform to create a persistent connection to the data location for your reports. A shared data source allows authorized report authors to share the connection configuration. This differs from creating an embedded connection for each report, which keeps a separate configuration in each individual report.

Creating alert reports

This recipe provides steps on creating Operations Manager alert reports.

Getting ready

It is recommended that you read the *Understanding the OperationsManagerDW schema* recipe in this chapter before following this recipe. You will need to prepare your environment to author reports following the instructions in the *Preparing your environment to author reports* recipe.

Report Builder

In the following recipes, we will use Report Builder 3.0 to create custom reports for Operations Manager. You need to be familiar with using Report Builder to manage and author reports. Using Report Builder is explained in *Chapter 3, Unpacking System Center Report Building Tools*.

Before we start authoring our report, we will take a look at the views that are available in the alert dataset in the `OperationsManagerDW` database.

Alert dataset

The alert dataset can be used to retrieve information about the alerts generated by rules and monitors in Operations Manager. The alert dataset includes four views that represent the data. All views are stored in the `Alert` database schema.

Alert.vAlert

The `Alert.vAlert` view contains all the alerts available in the `OperationsManagerDW` database. You can find general information about the alerts, such as `Name`, `Description`, `Severity`, `Priority`, `Category`, `RaisedDateTime`, `RepeatCount`, triggering workflow, and the managed entity it is associated with.

The `Alert.vAlert` view can be joined to the other views in the alert dataset on the `AlertGuid` column.

To retrieve the rule or monitor that triggered the alert, you can join `Alert.vAlert` to `vRule` and `vMonitor` on the `Alert.vAlert.WorkflowRowId` column. The following rule applies:

- ▶ If `Alert.vAlert.MonitorAlertInd` equals 1, then join `vMonitor` to `vAlert.vAlert.WorkflowRowId = vMonitor.MonitorRowId`
- ▶ If `Alert.vAlert.MonitorAlertInd` equals 0, then join `vRule` to `vAlert.vAlert.WorkflowRowId = vRule.RuleRowId`

You can join `Alert.vAlert` to `vManagedEntity` on `Alert.vAlert.ManagedEntityRowId = vManagedEntity.ManagedEntityRowId` to retrieve information about the object that the alert is associated with.

The following sample query returns all the alerts in the `OperationsManagerDW` database, including the triggering rule or monitor and the associated managed entity:

```
SELECT
    A.AlertGuid,
    A.AlertName,
    A.RaisedDateTime,
    CASE A.MonitorAlertInd WHEN 1 THEN M.MonitorDefaultName WHEN 0
THEN R.RuleDefaultName END AS WorkflowDefaultName,
    E.FullName AS ManagedEntityFullName
FROM
    Alert.vAlert A
    LEFT OUTER JOIN vMonitor M ON
        A.WorkflowRowId = M.MonitorRowId
    LEFT OUTER JOIN vRule R ON
        A.WorkflowRowId = R.RuleRowId
    LEFT OUTER JOIN vManagedEntity E ON
        A.ManagedEntityRowId = E.ManagedEntityRowId
```

Alert.vAlertDetail

The `Alert.vAlertDetail` view contains advanced information on the individual alerts, such as `Owner`, `TicketId`, and all the custom fields. You can join this view to the `Alert.vAlert` view on the `AlertGuid` column, as shown in the following query:

```
SELECT
    A.AlertGuid,
    A.AlertName,
    A.RaisedDateTime,
    AD.Owner,
    AD.TicketId,
    AD.CustomField1
FROM
    Alert.vAlert A
    LEFT OUTER JOIN Alert.vAlertDetail AD ON
        A.AlertGuid = AD.AlertGuid
```

Alert.vAlertParameter

This view contains the value for each parameter in the alerts. You can join this view to the `Alert.vAlert` view on the `AlertGuid` column, as shown in the following query:

```
SELECT
  A.AlertGuid,
  A.AlertName,
  A.RaisedDateTime,
  AP.ParameterIndex,
  AP.ParameterValue
FROM
  Alert.vAlert A
  LEFT OUTER JOIN Alert.vAlertParameter AP ON
    A.AlertGuid = AP.AlertGuid
```



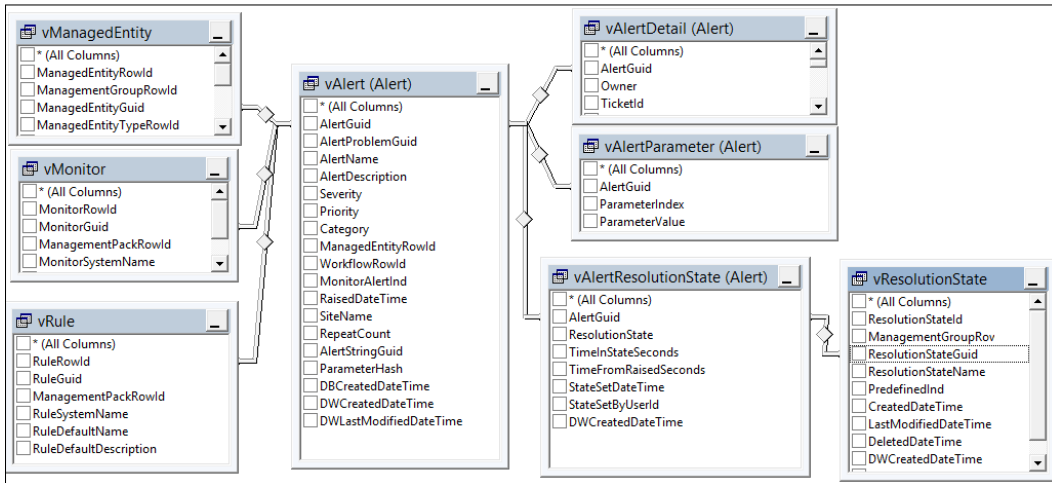
Note that this is a one-to-many relationship, that is, the join can return multiple rows per alert, one for each parameter.

Alert.vAlertResolutionState

The `Alert.vAlertResolutionState` view contains information about each resolution state the alerts were in, such as the user and the date/time it was set and how many seconds it was in this state. You can join this view to the `Alert.vAlert` view on the `AlertGuid` column. You could also join the `vResolutionState` view on `Alert.vAlertResolutionState.ResolutionState = vResolutionState.ResolutionStateId` if you want to resolve more information about the resolution state itself. Here is a sample query using the join to return resolution state information:

```
SELECT
  A.AlertGuid,
  A.AlertName,
  A.RaisedDateTime,
  RS.ResolutionStateName,
  AR.StateSetDateTime,
  AR.StateSetByUserId
FROM
  Alert.vAlert A
  LEFT OUTER JOIN Alert.vAlertResolutionState AR ON
    A.AlertGuid = AR.AlertGuid
  LEFT OUTER JOIN vResolutionState RS ON
    AR.ResolutionState = RS.ResolutionStateId
```

The following screenshot provides a visual view of the all the join relationships discussed in this section:



How to do it...

In this recipe, we will create a table report that shows the alerts from last month, ordered by the number of alerts per triggering rule/monitor. Perform the following steps:

1. Start Report Builder (use the Report Manager website or standalone option).
2. Under **New Report**, select **Table or Matrix Wizard**.
3. Select **Create a dataset** and click on **Next**.
4. Select the `OperationsManagerDW` data source connection. If it is not in the list, click on **Browse...** to browse to the data source in SQL Server Reporting Services. Click on **Next**.
5. If prompted for data source credentials, select **Use the current Windows user** and click on **OK**.
6. Click on **Edit as Text** and type the following SQL query in the query window:

```

DECLARE @DateTime datetime = GETUTCDATE()
DECLARE @StartOfPreviousMonth datetime = DATEADD(MONTH,
DATEDIFF(MONTH, '19000101', @DateTime) - 1, '19000101')
DECLARE @EndOfPreviousMonth datetime = DATEADD(MONTH,
DATEDIFF(MONTH, '19000101', @DateTime), '19000101')

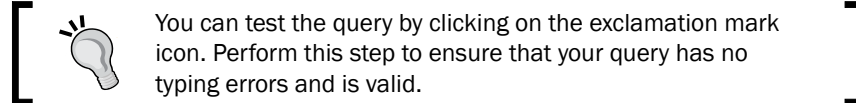
SELECT
    CASE A.MonitorAlertInd WHEN 1 THEN M.MonitorDefaultName
    WHEN 0 THEN R.RuleDefaultName END AS WorkflowName,
    A.AlertName,

```

```

A.RaisedDateTime,
ET.ManagedEntityTypeDefaultName AS EntityType,
E.FullName AS Entity,
ETI.Image AS EntityImage
FROM
Alert.vAlert A
LEFT OUTER JOIN vMonitor M ON
    A.WorkflowRowId = M.MonitorRowId
LEFT OUTER JOIN vRule R ON
    A.WorkflowRowId = R.RuleRowId
LEFT OUTER JOIN vManagedEntity E ON
    A.ManagedEntityRowId = E.ManagedEntityRowId
LEFT OUTER JOIN vManagedEntityType ET ON
    E.ManagedEntityTypeRowId = ET.ManagedEntityTypeRowId
LEFT OUTER JOIN vManagedEntityTypeImage ETI ON
    ET.ManagedEntityTypeRowId = ETI.ManagedEntityTypeRowId
    AND ETI.ImageCategory = 'u16x16Icon'
WHERE
    A.RaisedDateTime BETWEEN @StartOfPreviousMonth AND
@EndOfPreviousMonth
ORDER BY
    1 ASC, 3 DESC

```



7. Click on **Next**.
8. Drag **WorkflowName** from **Available fields** to the **Row groups** area.
9. Drag **AlertName**, **RaisedDateTime**, and **EntityType and Entity** from the **Available fields** area to the **Values** area.
10. Click on **Next** and choose **Show subtotals and grand totals, Blocked, subtotal above**, and **Expand/collapse groups**.
11. Click on **Next**, choose your preferred table style, and then click on **Finish**.

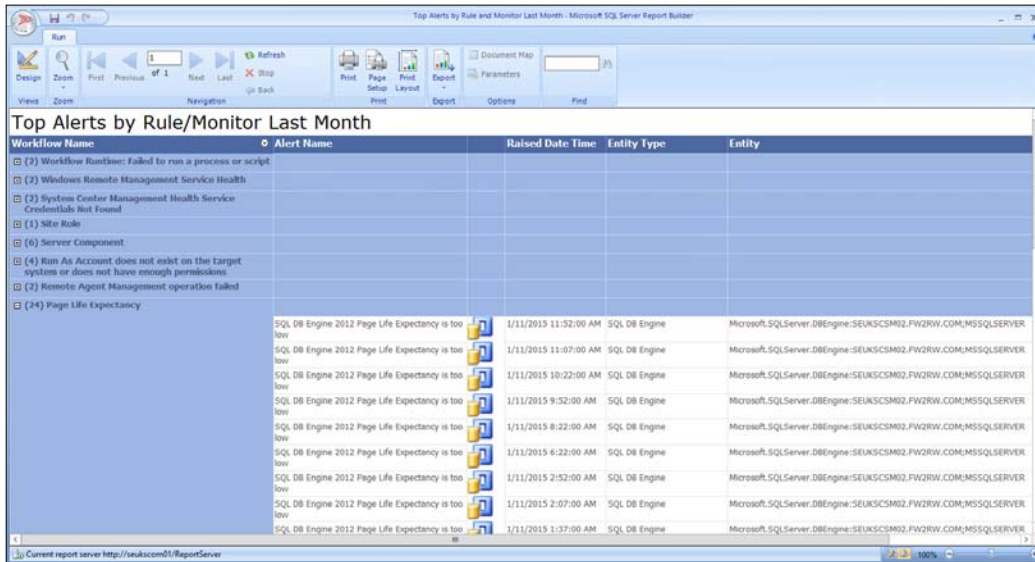
The wizard will now close, and Report Builder displays the table in your newly created report. You can now modify the report and the table as desired. Perform the following steps to modify the report:

1. Click on the **Click to add title** area and type **Top Alerts by Rule/Monitor Last Month**.
2. Change the font size and column width to accommodate the text length (click on the **Run** button to test the layout and click on **Design** to go back).

3. Select the **Alert Name** column. Right-click on the top gray part of the cell and select **Insert Column**.
4. Click on the **Insert** tab. Click on **Image**. Click on the cell to the right of **[AlertName]**.
5. In the **General** tab of **Image Properties**, select **Database** under **Select the image source**.
6. In **Use this field**:, select **[EntityImage]** and then select **image/jpeg** under **Use this MIME type**. Click on **OK**.
7. Right-click on a space in the **Workflow Name** cell (do not click on the text, just the space to the right of the text). Select **Text Box Properties**.
8. Click on **Interactive Sorting**. Select **Enable interactive sorting on this text box**. Under **Choose what to sort**:, select **Groups** and then select **WorkflowName**.
9. Under **Sort by**:, select **[WorkflowName]**. Click on **OK**.
10. Right-click on **[WorkflowName]**. Select **fx Expression....** Replace the formula in **Set expression for:Value** with the following code and click on **OK**:

```
= " ( "&CStr (RowNumber ( "WorkflowName" ) ) &" )
"&Fields!WorkflowName.Value
```

When you are done customizing your report, click on **Run**, as shown in the following screenshot:



Go back to the **Design** mode and click on **Save**. Navigate to the **Custom Reports** folder created when preparing your environment, enter **Top Alerts by Rule and Monitor Last Month** as the name, and then click on **Save**. Your report is now published to SQL Server Reporting Services.

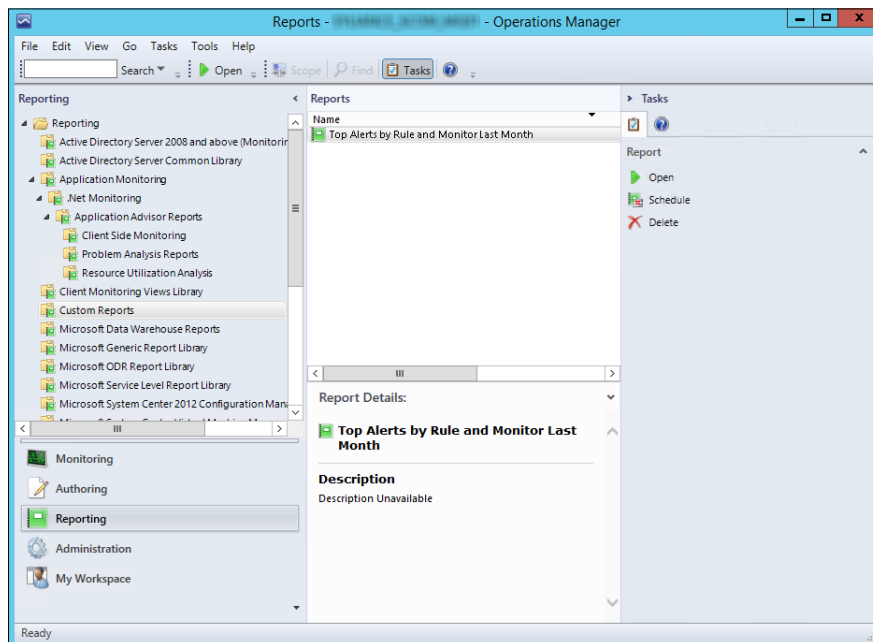
How it works...

The recipe provided the steps you can follow to create a report based on the **Alert** dataset. You must first select a report type, which, in this case, is Tablix. Next, you must select the target data source (`OperationsManagerDW`) that contains the alert information the report will retrieve. You use the SQL query provided to generate the dataset and organize the visualization using the Report Builder tool. The recipe finally suggests additional formatting that you can perform to improve the look and feel of your report. The specific formatting tasks suggested were as follows:

- ▶ Configure sorting on the `workflowName` row group
- ▶ Adding the group count to the workflow name
- ▶ Adding the entity image to the table
- ▶ Adding a title to the report
- ▶ Changing the font size and column width to accommodate text length

There's more...

Once your custom report has been published to SQL Server Reporting Services, it can be viewed from Operations Manager Console. The report will be available in the **Reporting** section under the folder that you created when going through the *Preparing your environment for authoring reports* recipe.



Creating event reports

This recipe provides steps on creating Operations Manager event type reports.

Getting ready

It is recommended that you read the *Understanding the OperationsManagerDW schema* recipe earlier in this chapter before following this recipe. You will need to prepare your environment to author reports by following the instructions in the *Preparing your environment to author reports* recipe.

Before we start authoring our report, we will take a look at the views that are available in the event dataset in the `OperationsManagerDW` database.

Event dataset

The event dataset can be used to retrieve information about the events collected from monitoring objects by **Operations Manager Event Collection Rules**. The event dataset includes four views that represent the data. All views are stored in the `Event` database schema.

Event.vEvent

The `Event.vEvent` view contains all the events available in the `OperationsManagerDW` database. Apart from information such as the event unique ID, event number, and event date/time, it also includes other unique IDs that can be joined to more views to retrieve additional information about the events.

The `Event.vEvent` view can be joined to the other views in the event dataset on the `EventOriginId` column.

You can join more views to the `Event.vEvent` view using the unique IDs available in the view, as outlined in the following table:

View	Join	Description
<code>vEventCategory</code>	<code>Event.vEvent.EventCategoryRowId = vEventCategory.EventCategoryRowId</code>	This is the event category information
<code>vEventChannel</code>	<code>Event.vEvent.EventChannelRowId = vEventChannel.EventChannelRowId</code>	This is the channel of the event, such as application, system, or the name of a custom event collection rule
<code>vEventLevel</code>	<code>Event.vEvent.EventLevelId = vEventLevel.EventLevelId</code>	This is the event level information, such as warning and error
<code>vEventLoggingComputer</code>	<code>Event.vEvent.LoggingComputerRowId = vEventLoggingComputer.EventLoggingComputerRowId</code>	This is the name of the computer that logged the event

View	Join	Description
vEventPublisher	Event.vEvent.EventPublisherRowId = vEventPublisher.EventPublisherRowId	This is the source in the event log or the name of a customer event collection rule
vEventUserName	Event.vEvent.UserNameRowId = vEventUserName.EventUserNameRowId	This is the user that was logged with the event

The following query returns all the events in the OperationsManagerDW database and resolves the detailed information from the views listed in the preceding table:

```

SELECT
    E.EventOriginId,
    E.EventNumber,
    E.DateTime,
    ECa.EventCategoryTitle,
    ECh.EventChannelTitle,
    ELe.EventLevelTitle,
    ELc.ComputerName,
    EPu.EventPublisherName,
    EUn.UserName
FROM
    Event.vEvent E
LEFT OUTER JOIN vEventCategory ECa ON
    E.EventCategoryRowId = ECa.EventCategoryId
LEFT OUTER JOIN vEventChannel ECh ON
    E.EventChannelRowId = ECh.EventChannelRowId
LEFT OUTER JOIN vEventLevel ELe ON
    E.EventLevelId = ELe.EventLevelId
LEFT OUTER JOIN vEventLoggingComputer ELc ON
    E.LoggingComputerRowId = ELc.EventLoggingComputerRowId
LEFT OUTER JOIN vEventPublisher EPu ON
    E.EventPublisherRowId = EPu.EventPublisherRowId
LEFT OUTER JOIN vEventUserName EUn ON
    E.UserNameRowId = EUn.EventUserNameRowId

```

Event.vEventDetail

The Event.vEventDetail view contains the description and event data of each individual event. You can join this view to the Event.vEvent view on the EventOriginId column, as shown in the following query:

```

SELECT
    E.EventOriginId,
    E.EventNumber,
    E.DateTime,

```

```

ED.RenderedDescription
FROM
Event.vEvent E
LEFT OUTER JOIN Event.vEventDetail ED ON
    E.EventOriginId = ED.EventOriginId
    
```

Event.vEventParameter

The `Event.vEventParameter` view contains all the parameters of an event and their corresponding values. This information is also available in the event description. You can join this view to the `Event.vEvent` view on the `EventOriginId` column, as shown in the following query:

```

SELECT
    A.AlertGuid,
    A.AlertName,
    A.RaisedDateTime,
    AP.ParameterIndex,
    AP.ParameterValue
FROM
    Alert.vAlert A
LEFT OUTER JOIN Alert.vAlertParameter AP ON
    A.AlertGuid = AP.AlertGuid
    
```



Note that this is a one-to-many relationship, that is, the join can return multiple rows per event, one for each parameter.

Event.vEventRule

The `Event.vEventRule` view contains the ID of the rule and the ID of the managed entity the event came from. You can join this view to the `Event.vEvent` view on the `EventOriginId` column. You can also join the `vRule` view to `Event.vEventRule`. `RuleRowId = vRule.RuleRowId` and the `vManagedEntity` view on `Event.vEventRule.ManagedEntityRowId = vManagedEntity.ManagedEntityRowId` if you want to resolve the rule and the object that the event came from:

```

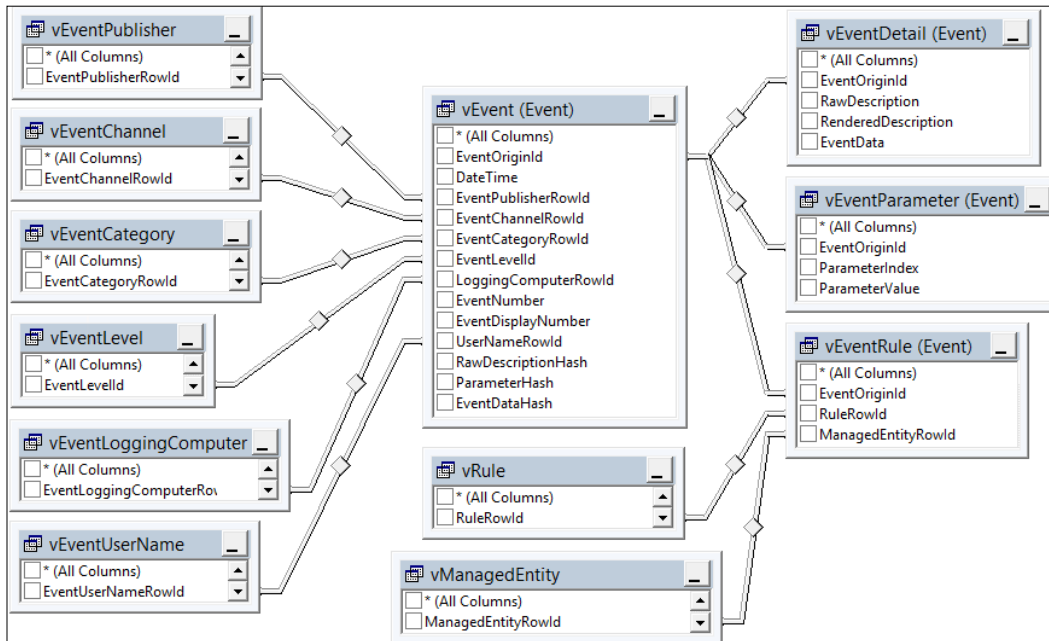
SELECT
    E.EventOriginId,
    E.EventNumber,
    E.DateTime,
    R.RuleDefaultName,
    M.FullName
FROM
    Event.vEvent E
LEFT OUTER JOIN Event.vEventRule ER ON
    E.EventOriginId = ER.EventOriginId
    
```

```

LEFT OUTER JOIN vRule R ON
    ER.RuleRowId = R.RuleRowId
LEFT OUTER JOIN vManagedEntity M ON
    ER.ManagedEntityRowId = M.ManagedEntityRowId

```

The following screenshot provides a visual view of the all the join relationships discussed in this section:



How to do it...

In this recipe, we will create a chart report that shows all the events by event level from the previous month. Perform the following steps:

1. Start Report Builder (use the Report Manager website or standalone option).
2. Under **New Report**, select **Chart Wizard**.
3. Select **Create a dataset** and click on **Next**.
4. Select the `OperationsManagerDW` data source connection. If it is not in the list, click on **Browse...** to browse to the data source in SQL Server Reporting Services. Click on **Next**.
5. If prompted for data source credentials, select **Use the current Windows user** and then click on **OK**.

6. Click on **Edit as Text** and type the following SQL query in the query window:

```
SELECT
    CONVERT(varchar(7), E.DateTime, 126) AS MonthRaised,
    ELe.EventLevelTitle,
    COUNT(*) AS EventCount
FROM
    Event.vEvent E
    LEFT OUTER JOIN vEventCategory ECa ON
        E.EventCategoryRowId = ECa.EventCategoryId
    LEFT OUTER JOIN vEventChannel ECh ON
        E.EventChannelRowId = ECh.EventChannelRowId
    LEFT OUTER JOIN vEventLevel ELe ON
        E.EventLevelId = ELe.EventLevelId
    LEFT OUTER JOIN vEventLoggingComputer ELc ON
        E.LoggingComputerRowId = ELc.EventLoggingComputerRowId
    LEFT OUTER JOIN vEventPublisher EPu ON
        E.EventPublisherRowId = EPu.EventPublisherRowId
    LEFT OUTER JOIN vEventUserName EUn ON
        E.UserNameRowId = EUn.EventUserNameRowId
GROUP BY
    CONVERT(varchar(7), E.DateTime, 126),
    ELe.EventLevelTitle
ORDER BY
    1
```

7. Click on **Next**, select the **Line** chart type, and then click on **Next**.
8. Drag **EventLevelTitle** from **Available fields** to the **Series** area.
9. Drag **MonthRaised** from **Available fields** to the **Categories** area.
10. Drag **EventCount** from **Available fields** to the **Values** area. Ensure that **Sum** is selected as the aggregation.
11. Click on **Next**, choose the desired chart style, and then click on **Finish**.
12. Click on the **Click to add title** area and type *Event by Level by Time*.

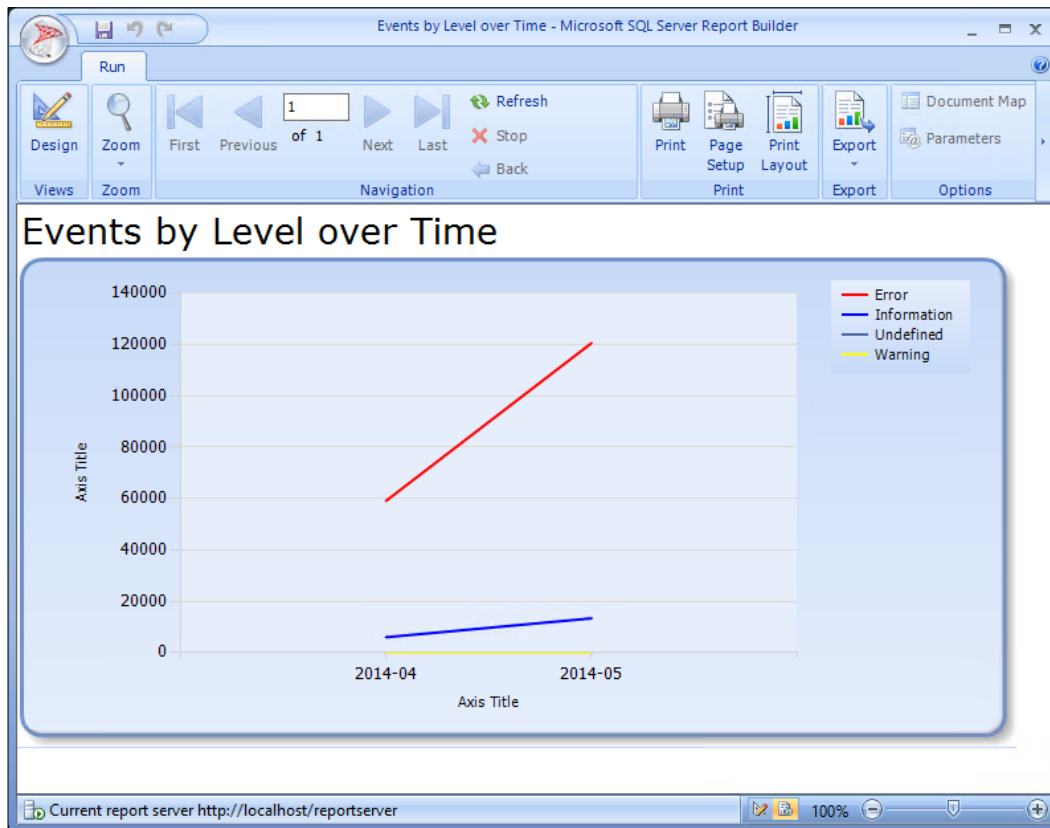
The wizard will now close, and Report Builder displays the chart in your newly created report. You can now modify the report and the chart as desired. You might want to change the line color to a color that represents the event level, such as red for errors and yellow for warnings. Perform the following steps:

1. Right-click the chart area. In Chart Data box on the right side of the chart area, under **Values**, click the arrow next to **EventCount**, and select **Series Properties**.
2. Under **Fill**, in the **Pick color** area, click on the formula icon (**fx**) to add an expression to the **Color** property.

3. Add the following expression to the **Set expression for: Color** textbox and click on **OK** twice:

```
=IIf (Fields!EventLevelTitle.Value="Error", "Red", IIf (Fields!EventLevelTitle.Value="Warning", "Yellow", IIf (Fields!EventLevelTitle.Value="Information", "Blue", "Automatic")))
```

When you are done customizing your report, click on **Run**, as shown in the following screenshot:



Go back to the **Design** mode and click on **Save**. Navigate to the **Custom Reports** folder created when preparing your environment, enter **Events by Level over Time** as the name, and click on **Save**. Your report is now published to SQL Server Reporting Services.

How it works...

The recipe provided the steps you follow to create a report based on Operations Manager Events data. You must first select a report type, which, in this case, is a chart. Next, you select the target data source (`OperationsManagerDW`) that contains the alert information the report will retrieve. You are provided with a primer on the four views available for creating event-based reports. The recipe discusses the typical view properties that are the best candidates for joins. You can use the example SQL query provided to generate the dataset and organize the chart report visualization using the Report Builder tool. The recipe finally suggested additional formatting and display optimization you can perform to improve the look and feel of your report.

Creating performance reports

This recipe provides steps on creating Operations Manager performance data type related reports.

Getting ready

It is recommended that you read the *Understanding the OperationsManagerDW schema* recipe earlier in this chapter before following this recipe. You will need to prepare your environment to author reports by following the instructions in the *Preparing your environment to author reports* recipe.

Before we start authoring our report, we will take a look at the views that are available in the **Performance** dataset in the `OperationsManagerDW` database.

You must use Microsoft Excel 2013 to complete some parts of this recipe.

Performance dataset

The performance dataset can be used to retrieve performance values collected from Operations Manager rules. Operations Manager stores both daily and hourly aggregates of performance values. These aggregates include the minimum, maximum, average, and standard deviation of the values. The performance dataset includes three views that represent the data. All views are stored in the `Perf` database schema.

Perf.vPerfDaily and Perf.vPerfHourly

The Perf.vPerfDaily and Perf.vPerfHourly views contain aggregated performance data by day and hour, respectively. The following aggregates are available in these views:

View column	Aggregate	Description
MinValue	Minimum value	This is the minimum performance rule value in the sample set
MaxValue	Maximum value	This is the maximum performance rule value in the sample set
AverageValue	Average value	This is the average of all the performance values in the sample set
StandardDeviation	Standard deviation	This is the standard deviation of the sample set

The number of values the aggregate is based on is represented in the SampleCount column.

You can join the vManagedEntity view on the ManagedEntityRowId column to retrieve information on the monitored object the performance data is associated with.

To retrieve information on the performance rules, you can join vPerformanceRuleInstance on the PerformanceRuleInstanceRowId column. Each counter that collected performance data is represented by a row in the vPerformanceRuleInstance view. You can then join the vPerformanceRule view on the RuleRowId column to get the counter and object names that were collected or join the vRule view on the RuleRowId column to get the rule that collected the performance counter.

The following query returns yesterday's hourly performance aggregates, including the counters, rules, and managed entities associated with the performance data:

```

SELECT
    ME.FullName,
    R.RuleDefaultName,
    PR.ObjectName,
    PR.CounterName,
    P.SampleCount,
    P.MinValue,
    P.MaxValue,
    P.AverageValue,
    P.StandardDeviation
FROM
    Perf.vPerfHourly P
    INNER JOIN vManagedEntity ME ON
        P.ManagedEntityRowId = ME.ManagedEntityRowId
    INNER JOIN vPerformanceRuleInstance PRI ON

```



```
P.PerformanceRuleInstanceId =
PRI.PerformanceRuleInstanceId
INNER JOIN vPerformanceRule PR ON
PRI.RuleRowId = PR.RuleRowId
INNER JOIN vRule R ON
PRI.RuleRowId = R.RuleRowId
WHERE
YEAR(P.DateTime) = YEAR (DATEADD(d, -1, GETDATE()))
AND MONTH(P.DateTime) = MONTH (DATEADD(d, -1, GETDATE()))
AND DAY(P.DateTime) = DAY (DATEADD(d, -1, GETDATE()))
```

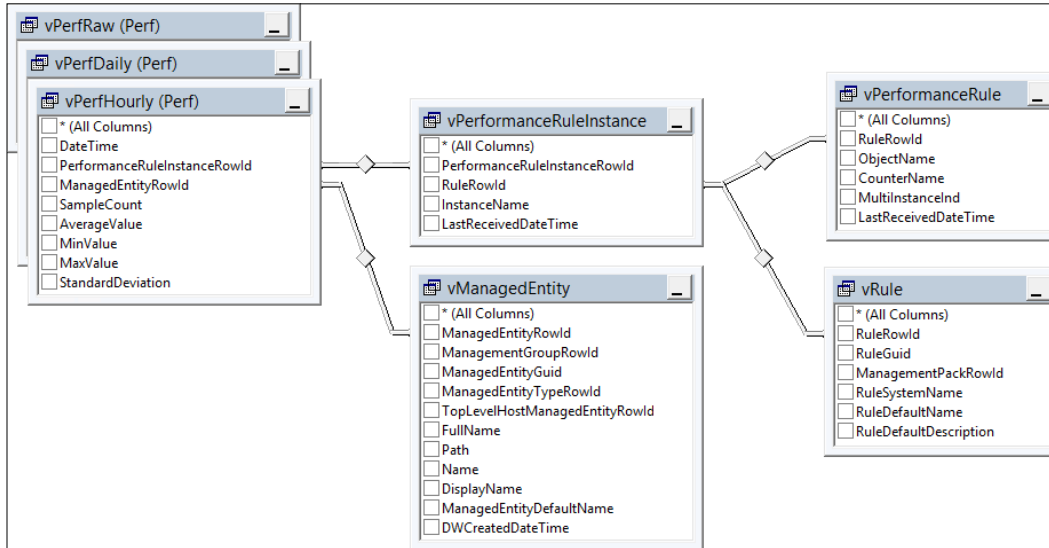
Perf.vPerfRaw

The `Perf.vPerfRaw` view contains the raw performance data without aggregates. Using this view is helpful when you want to create custom aggregates in your performance reports. This view can also be joined to the `vManagedEntity` and `vPerformanceRuleInstance` views to retrieve information on the counters, rules, and monitored objects.

The following query returns monthly aggregates of performance data. It also includes additional columns for other aggregate functions, such as standard deviation for the population for all values and variance:

```
SELECT
CONVERT (varchar(7), P.DateTime, 126) AS MonthlyAggregate,
P.ManagedEntityRowId,
P.PerformanceRuleInstanceId,
COUNT(P.SampleValue) AS SampleCount,
MIN(P.SampleValue) AS MinValue,
MAX(P.SampleValue) AS MaxValue,
AVG(P.SampleValue) AS AverageValue,
STDEV(P.SampleValue) AS StandardDeviation,
VAR(P.SampleValue) AS Variance,
STDEVP(P.SampleValue) AS StandardDeviationPopulation,
VARP(P.SampleValue) AS VariancePopulation
FROM
Perf.vPerfRaw P
GROUP BY
CONVERT (varchar(7), P.DateTime, 126),
P.ManagedEntityRowId,
P.PerformanceRuleInstanceId
```

The following screenshot shows a graphical view of the joins discussed in this section:



How to do it...

In this example, we will use Microsoft Excel to analyze performance data using ad hoc PivotTables and PivotCharts. We will use a view in the `OperationsManagerDW` database to provide an abstraction layer from the database model. The view will hold friendly names for the columns and will be limited to the data we need for our report.

To create the view, open SQL Server Management Studio and connect to the SQL server that hosts the `OperationsManagerDW` database. Then, execute the following query:

```
USE OperationsManagerDW
GO

CREATE VIEW v_Custom_r_PerformanceDataDaily AS

SELECT
    P.DateTime AS [Date],
    YEAR(P.DateTime) AS [Year],
    MONTH(P.DateTime) AS [Month],
    DAY(P.DateTime) AS [Day],
    MET.ManagedEntityTypeDefaultName AS [Managed Entity Type],
    ME.Name AS [Managed Entity Name],
    ME.FullName AS [Managed Entity Full Name],
    R.RuleDefaultName AS [Rule Name],
```

```
PR.ObjectName AS [Performance Object],
PR.CounterName AS [Performane Counter],
P.SampleCount AS [Sample Count],
P.MinValue AS [MIN],
P.MaxValue AS [MAX],
P.AverageValue AS [AVG],
P.StandardDeviation AS [STDEV]
FROM
Perf.vPerfDaily P
INNER JOIN vManagedEntity ME ON
P.ManagedEntityRowId = ME.ManagedEntityRowId
INNER JOIN vManagedEntityType MET ON
ME.ManagedEntityTypeRowId = MET.ManagedEntityTypeRowId
INNER JOIN vPerformanceRuleInstance PRI ON
P.PerformanceRuleInstanceRowId =
PRI.PerformanceRuleInstanceRowId
INNER JOIN vPerformanceRule PR ON
PRI.RuleRowId = PR.RuleRowId
INNER JOIN vRule R ON
PRI.RuleRowId = R.RuleRowId
```

Next, we need to grant permissions for the `OpsMgrReader` database role to the newly created view. This can be done by executing the following query:

```
USE OperationsManagerDW
GO
GRANT SELECT ON v_Custom_r_PerformanceDataDaily TO OpsMgrReader
GO
```



You need to make sure that the users who access the `OperationsManagerDW` database from Microsoft Excel are members of the `OpsMgrReader` database role.

It is recommended that you create an Active Directory group and add this group as a login to SQL Server. Add all users to whom you want to grant access to the data to this Active Directory group. Then, add the login as a user to the `OperationsManagerDW` database and assign it to the `OpsMgrReader` database role.

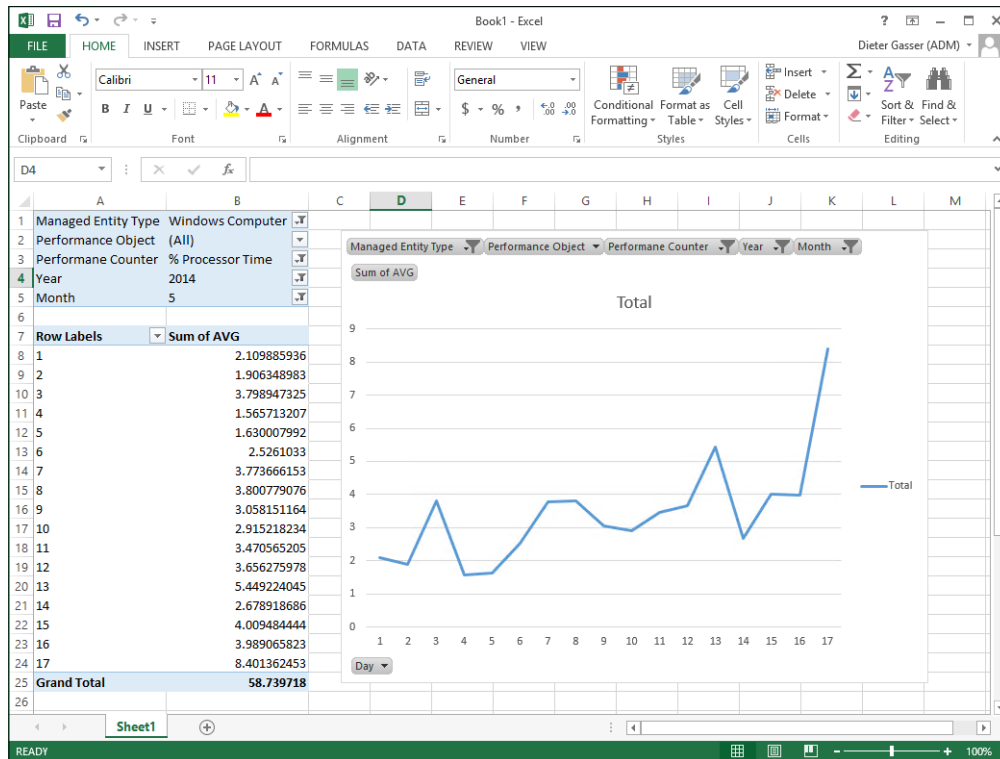
Perform the following steps:

1. Start Microsoft Excel and open a blank workbook.
2. On the **Data** ribbon, in the **Get External Data** area, click on **From Other Sources** and then choose **From SQL Server**.
3. In the **Server name** field, enter the name of the SQL Server that hosts the `OperationsManagerDW` database, select **Use Windows Authentication**, and then click on **Next**.

4. Select the `OperationsManagerDW` database, click on the **v_Custom_r_PerformanceDataDaily** view, and then click on **Finish**.
5. On the **Import Data** dialog, select **PivotChart** and then click on **OK**.

Microsoft Excel is now connecting to the data source and retrieving the data from the view created earlier in this recipe. You can now use common Excel features to analyze your data, such as the following:

1. In the **PivotTable Fields** pane, drag and drop the **Managed Entity Type**, **Performance Object**, **Performance Counter**, **Year**, and **Month** fields to the **Filters** area.
2. In the **PivotTable Fields** pane, drag and drop the **Day** field to the **AXIS (CATEGORIES)** area.
3. In the **PivotTable Fields** pane, drag and drop the **AVG** field to the **Values** area.
4. Right-click on the chart. Select **Change Chart Type...** and then **Line**. Click on **OK**.
5. Set filters by clicking on the arrow next to the **Filter** field in the chart. Select **Select Multiple items**. Uncheck **(All)** and select the fields you want to use as a filter, as shown in the following screenshot:



6. Save the Excel workbook.

When you want Excel to load the latest data from the database, right-click anywhere inside the data table and click on **Refresh**.

How it works...

The recipe provided you with the explanation of performance-related data available to you in the Operations Manager Data Warehouse database. You are provided with steps and illustrations as follows:

- ▶ **Performance dataset:** These are views that contain the information you need for this category and the specific property types you must use.
- ▶ **Create a custom view:** The steps to create a custom filter view is provided to reduce the amount of data you base your report on. Creating a view organizes the data by effectively saving the SQL query.
- ▶ **Security delegation:** You must assign the right permissions to the view you created to allow access to report authors.
- ▶ **Excel Pivot tables:** The final part of the recipe provides steps on using Microsoft Excel to retrieve the data using the custom view and present the desired report as a PivotTable.

Once you have completed these steps, you have the option to share the excel report with your report consumers.

See also

- ▶ There is a great performance visualization solution available from a company called Squared Up; visit <http://squaredup.com/product/scom-performance-reporting/> for additional details

Creating availability reports

This recipe provides steps on creating Operations Manager availability data type related reports.

Getting ready

It is recommended that you read the *Understanding the OperationsManagerDW schema* recipe earlier in this chapter before following this recipe. You will need to prepare your environment to author reports by following the instructions in the *Preparing your environment to author reports* recipe.

State dataset

The state dataset can be used to create availability reports. This data is collected by the monitors in Operations Manager. The views available in the state dataset contain information about the state of your managed entities. All views are stored in the `State` database schema.

A monitored object can be in different states. These states are known in Operations Manager and recorded in views. The views available in the state dataset include one column for the number of milliseconds the object has been in this state over a particular period. These columns are as follows:

- ▶ `InRedStateMilliseconds`: This means that object was in critical state
- ▶ `InYellowStateMilliseconds`: This means that object was in warning state
- ▶ `InDisabledStateMilliseconds`: This means that monitor was disabled
- ▶ `InPlannedMaintenanceMilliseconds`: This means that object was in planned maintenance
- ▶ `InUnplannedMaintenanceMilliseconds`: This means that object was in unplanned maintenance
- ▶ `HealthServiceUnavailableMilliseconds`: The health service of the object was unavailable

State.vStateHourly and State.vStateDaily

The `State.vStateDaily` and `State.vStateHourly` views contain aggregated state information by day and by hour, respectively.

You can join the `vManagedEntityMonitor` view on the `ManagedEntityMonitorRowId` column. From this view, you can join the `vManagedEntity` view on the `ManagedEntityRowId` column to retrieve information about the managed entity, the state that data is associated with it, and you can join the `vMonitor` view on the `MonitorRowId` column to retrieve information about the monitor that generated the state data.

The `IntervalEndHealthState` column in the `State.vStateHourly` view contains the state that the object was in at the end of the monitoring interval. You can join this column on the `HealthStateRowId` column of the `vHealthState` view to retrieve the name of the health state.

The following query returns yesterday's hourly state aggregates, including the monitors, managed entities, and end health states:

```
SELECT
    M.MonitorDefaultName,
    ME.FullName,
    HS.HealthStateDefaultName,
    S.DateTime,
    S.InRedStateMilliseconds,
```

```
S.InYellowStateMilliseconds,
S.InDisabledStateMilliseconds,
S.InPlannedMaintenanceMilliseconds,
S.InUnplannedMaintenanceMilliseconds,
S.HealthServiceUnavailableMilliseconds
FROM
State.vStateHourly S
INNER JOIN vManagedEntityMonitor MEM ON
    S.ManagedEntityMonitorRowId = MEM.ManagedEntityMonitorRowId
INNER JOIN vManagedEntity ME ON
    MEM.ManagedEntityRowId = ME.ManagedEntityRowId
INNER JOIN vMonitor M ON
    MEM.MonitorRowId = M.MonitorRowId
INNER JOIN vHealthState HS ON
    S.IntervalEndHealthState = HS.HealthStateRowId
WHERE
    YEAR(S.DateTime) = YEAR(DATEADD(d, -1, GETDATE()))
    AND MONTH(S.DateTime) = MONTH(DATEADD(d, -1, GETDATE()))
    AND DAY(S.DateTime) = DAY(DATEADD(d, -1, GETDATE()))
```

State.vStateRaw

The `State.vStateRaw` view contains the raw state data without aggregates. It includes one row per state change for each managed entity. You can find the date/time that the state change occurred, and you can also find the old and new state of the object. In other words, the monitored object transitioned from the health state indicated by the `OldHealthState` column to the health state indicated by the `NewHealthState` column at the date/time indicated by the `DateTime` column.

You can join `vManagedEntityMonitor` on `ManagedEntityMonitorRowId` to retrieve information about the managed entity and the monitor, and you can join `vHealthState` on the `OldHealthState` and `NewHealthState` columns to retrieve health state information.

In the following example, we will retrieve a list of all state changes that are associated with a specific monitored object and a specific monitor. You can use the `vManagedEntityMonitor`, `vManagedEntity`, and `vMonitor` views to select a sample managed entity and a sample monitor from your environment. You must replace the value for `ManagedEntityRowId` and `MonitorRowId` with the IDs from your environment, as shown in the following query:

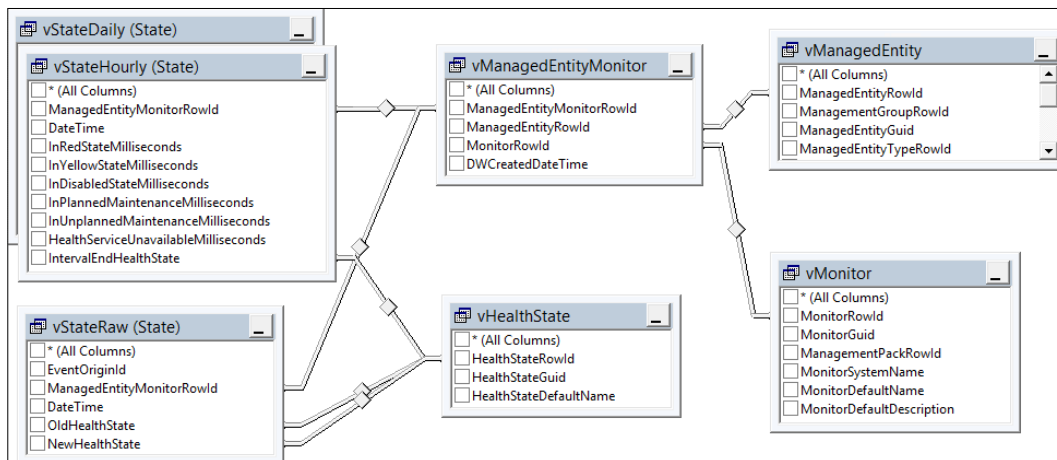
```
SELECT
    M.MonitorDefaultName,
    ME.FullName,
    S.DateTime,
    HSOld.HealthStateDefaultName AS OldState,
    HSNew.HealthStateDefaultName AS NewState
```

```

FROM
  State.vStateRaw S
  INNER JOIN vManagedEntityMonitor MEM ON
    S.ManagedEntityMonitorRowId = MEM.ManagedEntityMonitorRowId
  INNER JOIN vManagedEntity ME ON
    MEM.ManagedEntityRowId = ME.ManagedEntityRowId
  INNER JOIN vMonitor M ON
    MEM.MonitorRowId = M.MonitorRowId
  INNER JOIN vHealthState HSOld ON
    S.OldHealthState = HSOld.HealthStateRowId
  INNER JOIN vHealthState HSNew ON
    S.NewHealthState = HSNew.HealthStateRowId
WHERE
  ME.ManagedEntityRowId = 1299
  AND M.MonitorRowId = 1
ORDER BY
  S.DateTime

```

The following screenshot provides a visual view of the all the join relationships discussed in this section:



How to do it...

In this example, we will create a report using Report Builder; this report shows all state changes for a particular monitored object and monitor. Both the monitored object and the monitor will be made available as parameters in the report.

For this to work, we will use several datasets in our report; they are described in the following table:

Dataset	Description
PSTypes	This dataset is used as the source of a parameter in our report. This parameter allows us to pick the managed entity type for the report.
PSObjects	This dataset is used as the source of a parameter in our report. This parameter allows us to pick the managed entity for the report.
PSMonitors	This dataset is used as the source of a parameter in our report. This parameter allows us to pick the monitor for the report.
DSState	This dataset returns state changes of the selected managed entity and monitor.

Perform the following steps:

1. Start Report Builder (use the Report Manager website or run from a standalone installation).
2. Under **New Report**, select **Blank Report**.
3. In the **Report Data** area, right-click on **Data Sources** and add the OperationsManagerDW shared data source.
4. Click on **Test Connection**. Then, click on **OK**.
5. In the **Report Data** area, right-click on **Datasets** and add a dataset named PSTypes (type this in the **Name:** field). Select the **Use a dataset embedded in my report** option, select the OperationsManagerDW data source, select **Text** for the query type, and type the following text. Then, click on **OK**:

```
SELECT DISTINCT
    MET.ManagedEntityTypeRowId AS [Id],
    MET.ManagedEntityTypeDefaultName AS [Name]
FROM
    vManagedEntityMonitor MEM
    INNER JOIN vManagedEntity ME ON
        MEM.ManagedEntityRowId = ME.ManagedEntityRowId
    INNER JOIN vManagedEntityType MET ON
        ME.ManagedEntityTypeRowId = MET.ManagedEntityTypeRowId
ORDER BY
    MET.ManagedEntityTypeDefaultName
```

6. In **Enter Data Source Credentials**, select **Use the current Windows user**, and then click on **OK**.
7. In the **Report Data** area, right-click on **Parameters** and add a parameter named **Type**. Select the **Text** data type and ensure that all options regarding blank, null and allow multiple values are unchecked. Also, set the visibility to **Visible**.

8. Under **Available Values**, choose **Get values from a query**, pick the **PSTypes** dataset and pick **Id** as **Value field** and **Name** as **Label field**. Then, click on **OK**.
9. In the **Report Data** area, right-click on **Datasets** and add a dataset named **PSObjects**. Select the **Use a dataset embedded in my report** option, select the **OperationsManagerDW** data source, select the **Text** query type, and paste the following query. Then, click on **OK**:

```
SELECT DISTINCT
    ME.ManagedEntityRowId AS [Id],
    ME.FullName AS [Name]
FROM
    vManagedEntityMonitor MEM
    INNER JOIN vManagedEntity ME ON
        MEM.ManagedEntityRowId = ME.ManagedEntityRowId
WHERE
    ME.ManagedEntityTypeRowId = @Type
ORDER BY
    ME.FullName
```

10. In the **Report Data** area, right-click on **Parameters** and add a parameter named **Object**. Select the **Text** data type and ensure that all options regarding blank, null, and allow multiple values are unchecked. Also, set the visibility to **Visible**.
11. Under **Available Values**, choose **Get values from a query**, pick the **PSObjects** dataset, and pick **Id** as **Value field** and **Name** as **Label field**. Then, click **OK**.
12. In the **Report Data** area, right-click on **Datasets** and add a dataset named **PSMonitors**. Select the **Use a dataset embedded in my report** option, select the **OperationsManagerDW** data source, select **Text** query type, and type the following query. Then, click on **OK**:

```
SELECT DISTINCT
    M.MonitorRowId AS [Id],
    M.MonitorDefaultName AS [Name]
FROM
    vManagedEntityMonitor MEM
    INNER JOIN vMonitor M ON
        MEM.MonitorRowId = M.MonitorRowId
WHERE
    MEM.ManagedEntityRowId = @Object
ORDER BY
    M.MonitorDefaultName
```

13. In the **Report Data** area, right-click on **Parameters** and add a parameter named **Monitor**. Select the **Text** data type and disable all the options regarding blank, null, and multiple values. Also, set the visibility to **Visible**.

14. Under **Available Values**, choose **Get values from a query**, pick the **PSMonitors** dataset, and pick **Id** as **Value field** and **Name** as **Label field**. Then, click on **OK**.
15. In the **Report Data** area, right-click on **Datasets** and add a dataset named **DSState**. Select the **Use a dataset embedded in my report** option, select the **OperationsManagerDW** data source, select **Text** query type, and type the following query. Then, click on **OK**:

```

SELECT
    S.DateTime,
    HSOld.HealthStateDefaultName AS OldState,
    HSNew.HealthStateDefaultName AS NewState
FROM
    State.vStateRaw S
    INNER JOIN vManagedEntityMonitor MEM ON
        S.ManagedEntityMonitorRowId =
MEM.ManagedEntityMonitorRowId
    INNER JOIN vHealthState HSOld ON
        S.OldHealthState = HSOld.HealthStateRowId
    INNER JOIN vHealthState HSNew ON
        S.NewHealthState = HSNew.HealthStateRowId
WHERE
    MEM.ManagedEntityRowId = @Object
    AND MEM.MonitorRowId = @Monitor
ORDER BY
    S.DateTime
    
```

Now, it's time to design our report by performing the following steps:

1. In **Click to add title**, type `State changes for object/monitor`.
2. Click on the **Insert** tab. Select **Text Box**. Draw a box below the title and type `Object` as the text.
3. Click on the **Insert** tab. Select **Text Box**. Draw a box below the **Object** text box and type `Monitor` as the text.
4. Go to **Datasets | PSObjects**, click on **Name**, and drag it to the space next to the **Object** text box (a new box with `<<Expr>>` is placed in the report pane next to the text box).
5. Go to **Datasets | PSMonitors**, click on **Name**, and drag it to the space next to the **Object** text box (a new box with `<<Expr>>` is placed in the report pane next to the text box).
6. Click on the **Insert** tab. Select **Table**. Then, select **Insert Table**. Draw a table below the **Monitor** box.
7. Type `Date Time`, `Old State`, and `New State`, respectively, in the three header cells.

8. In the table **Data** area, click on the **Table** icon and select the respective fields for the headers using the `DSState` dataset.
9. You can apply formatting to the table and text boxes using the formatting control tools (refer to the following screenshot for layout and design recommendations).
10. Use the **Run** button to test the layout and report. You will be presented with the parameter selections first. When you complete the three-prompt selection options, click on **View Report** (on the right-hand side of the screen).
11. Save the report with `State changes for object and monitor` as the name in the `Custom Reports` folder.
12. When you run your reports, choose a type, and then a monitored object of this type. Pick a monitor, and the report will show all state changes that the monitored object went through.

When you run your reports, choose a type, and then a monitored object of this type. Pick a monitor, and the report will show all state changes that the monitored object went through.

The screenshot shows the Microsoft SQL Server Report Builder interface. The title bar reads "State changes for object and monitor - Microsoft SQL Server Report Builder". The interface includes a toolbar with options like Run, Design, Zoom, Navigation, Refresh, Print, Page Setup, Print Layout, Export, and Parameters. Below the toolbar, there are dropdown menus for "Select Type" (set to "ConfigMgr") and "Select Object" (set to "Microsoft.SystemCenter2012.ConfigurationManagement"). A "View Report" button is visible on the right. The main content area displays the report title "State changes for object/monitor" and the following data:

State changes for object/monitor		
Object	Microsoft.SystemCenter2012.ConfigurationManagement	
Monitor	All Contained Objects	
Date Time	Old State	New State
1/10/2015 12:18:24 PM	Unmonitored	Unmonitored
1/11/2015 2:01:26 AM	Error	Error
1/11/2015 2:02:26 AM	Unmonitored	Unmonitored
1/11/2015 3:19:17 PM	Error	Error
1/11/2015 3:26:17 PM	Unmonitored	Unmonitored
1/11/2015 3:42:18 PM	Error	Error
1/11/2015 3:43:17 PM	Unmonitored	Unmonitored
1/13/2015 2:08:04 AM	Success	Success
1/17/2015 3:16:12 PM	Error	Error

How it works...

The recipe provided you with the explanation of availability-related data available to you in the Operations Manager Data Warehouse database. You are provided with steps and illustrations as follows:

- ▶ **State dataset:** This has the views that contain the information you need for this category and the specific property types you must use.
- ▶ **Create datasets with parameters:** The steps to create four datasets (predefined queries) with parameters. The parameters provide you with the option to specify different criterion before executing the report.

- ▶ **Design the report layout:** The final part of the recipe provides steps on using Report Builder to present the desired report as a table. The parameters are made visible when you run the report, and you are also provided with recommendations on the layout design using labels.

Once you have completed these steps, you can publish (save) the report and share it with your report consumers.

Using Virtual Machine Manager reports

In this recipe, we will show you how to prepare for and access System Center Virtual Machine Manager built-in reports using the SCOM console.

Getting ready

Before you can connect **Virtual Machine Manager (VMM)** with Operations Manager so that Operations Manager monitors and reports on your virtual environment, you need to ensure that your environment is supported. These are the supported combinations of versions of System Center Virtual Machine Manager and System Center Operations Manager:

- ▶ If you are running System Center 2012 Virtual Machine Manager, you can use either Operations Manager 2007 R2 or System Center 2012 Operations Manager
- ▶ If you are running System Center 2012 SP1 Virtual Machine Manager, you must use System Center 2012 SP1 Operations Manager
- ▶ If you are running System Center 2012 R2 Virtual Machine Manager, you must use System Center 2012 R2 Operations Manager

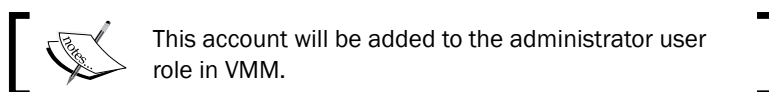
You must also ensure that the following prerequisites are met before you can connect VMM with Operations Manager:

- ▶ For System Center 2012, Windows PowerShell 2.0 must be installed on all Operations Manager management servers.
- ▶ For System Center 2012 SP1 or System Center 2012 R2, Windows PowerShell 3.0 must be installed on all Operations Manager management servers.
- ▶ Port 5724 must be open between VMM and Operations Manager.
- ▶ You must install an Operations Manager Console on the VMM management server.
- ▶ You must install Operations Manager agents on all virtualization hosts under management by VMM. This includes the VMM management server.

- ▶ The managed hosts on which you installed Operations Manager agents must show up in the Operations Manager Console; you can view these agents by going to **Administration | Device Manager | Agent Managed**.
- ▶ If you are running System Center 2012 SP1 or System Center 2012 R2, double-click on a host in the list, click on the **Security** tab, and then ensure that **Allow this agent to act as a proxy and discover managed objects on other computers** has been selected. Repeat this step for each of the hosts.
- ▶ Import the following management packs in Operations Manager:
 - ❑ Windows Server Internet Information Services 2003
 - ❑ Windows Server 2008 Operating System (Discovery)
 - ❑ Windows Server Operating System Library
 - ❑ Windows Server 2008 Internet Information Services 7
 - ❑ SQL Server Core Library

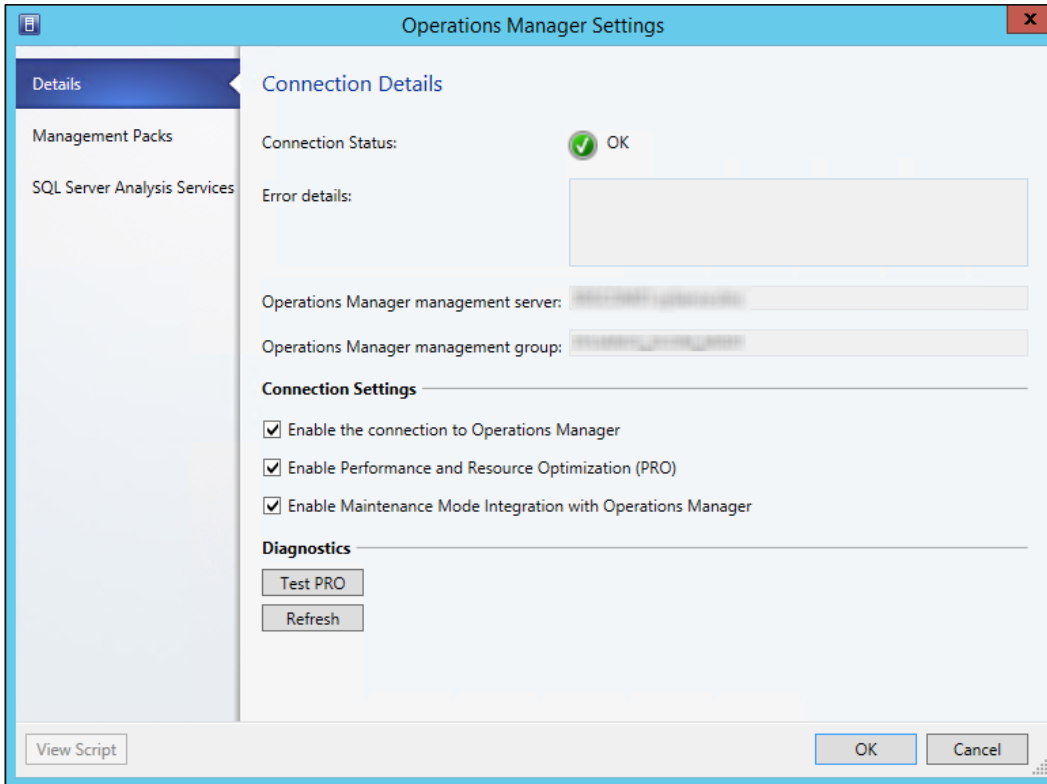
We will now walk you through the required steps to enable integration between Virtual Machine Manager and Operations Manager:

1. Start the Virtual Machine Manager console.
2. Under **Settings**, select **System Center Settings** and then double-click on **Operations Manager Server**.
3. Click on **Next**. Enter the server name of one of your Operations Manager management servers and select an account to use to connect to the server. You can use the VMM server service account or specify a **Run As** account. The account you specify must be a member of the Operations Manager Administrator role.
4. Optionally, select **Enable Performance and Resource Optimization (PRO)**.
5. Optionally, select **Enable maintenance mode integration with Operations Manager**.
6. Click on **Next**, and then enter the credentials for Operations Manager to connect with the VMM management server.



7. Click on **Next** and then click on **Finish**.

You can verify that the connection status is OK by opening the Operations Manager Server dialog box again and reviewing the status next to **Connection Status**, as shown in the following screenshot:

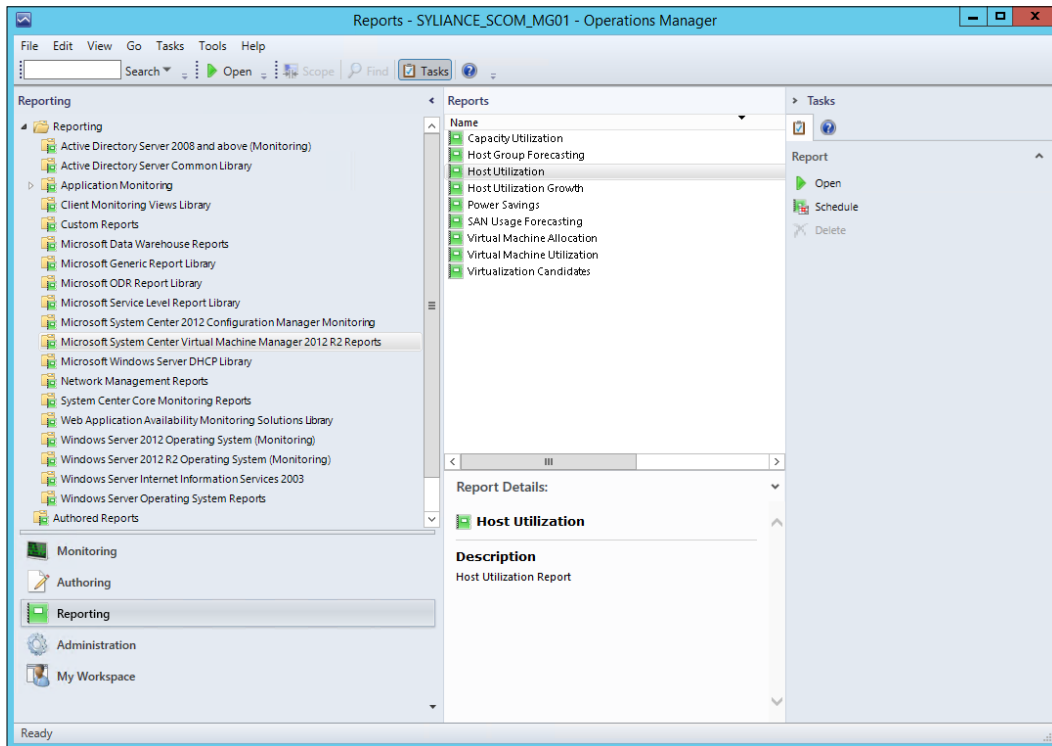


Once the synchronization between Virtual Machine Manager and Operations Manager is complete, you can verify the successful connection in Operations Manager Console in the **Monitoring** section. Make sure that the Microsoft System Center Virtual Machine Manager and Microsoft System Center Virtual Machine Manager Views folders exist.

How to do it...

Once Virtual Machine Manager and Operations Manager are connected and monitoring data is available from Operations Manager, you can use the reports that ship as part of the Virtual Machine Manager reporting management pack in Operations Manager.

You can find the reports in the **Microsoft System Center Virtual Machine Manager 2012 Reports** or **Microsoft System Center Virtual Machine Manager 2012 R2 Reports** folder in the **Reporting** section in Operations Manager Console, as shown in the following screenshot:



Refer to the *Using out-of-box Operations Manager reports* recipe earlier in this chapter for instructions on how to work with built-in Operations Manager reports.

How it works...

The recipe provides you with the steps that you must follow to run out-of-the-box Virtual Machine Manager reports. You can access the reports in the Operations Manager console using a user account with the appropriate rights to perform the actions.

6

Creating Reports for System Center Data Protection Manager

In this chapter, we will cover the following recipes:

- ▶ Preparing the DPM reporting environment
- ▶ Preparing a dataset query for a DPM agent status report
- ▶ Creating a DPM agent version and a disk space report
- ▶ Creating a backup status report with a community template

Introduction

This chapter focuses on creating and modifying reports using the **Data Protection Manager (DPM)** database. The installation of DPM includes reporting using **Microsoft SQL Server Reporting Services (SSRS)**. The recipes in this chapter use DPM 2012 R2.

For all recipes in this chapter, the requirements are the following:

- ▶ A deployed System Center 2012 R2 Data Protection Manager environment.
- ▶ Agents deployed and protection enabled.
- ▶ Protection configured for the disk. In the cookbook environment, there are three protection groups: clients, domain controllers, and member servers.

Preparing the DPM reporting environment

Shortcuts lead to long delays! There is no shortcut for preparation. This recipe is focused on preparing your environment for your DPM custom reports.

Getting ready

You must plan to review *Chapter 3, Unpacking System Center Report Building Tools*, as a primer to this recipe.

How to do it...

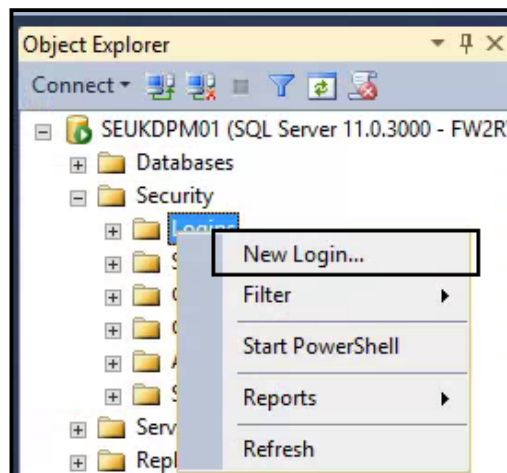
The tasks discussed in this recipe are as follows:

- ▶ Creating and delegating read access to an Active Directory user account to access the DPM database
- ▶ Creating DPM Report Manager folders and performing delegation
- ▶ Creating a shared data source for DPM reports

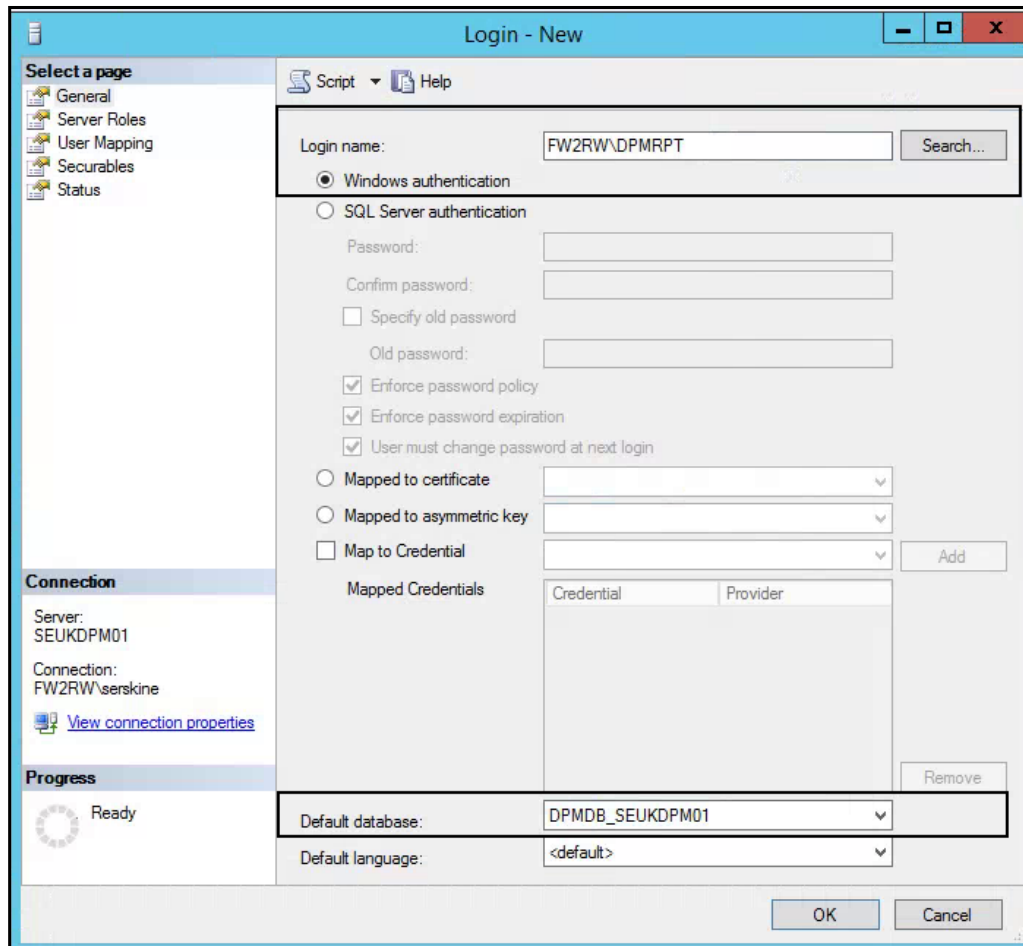
Creating and delegating read access to an Active Directory account user

Perform the following steps:

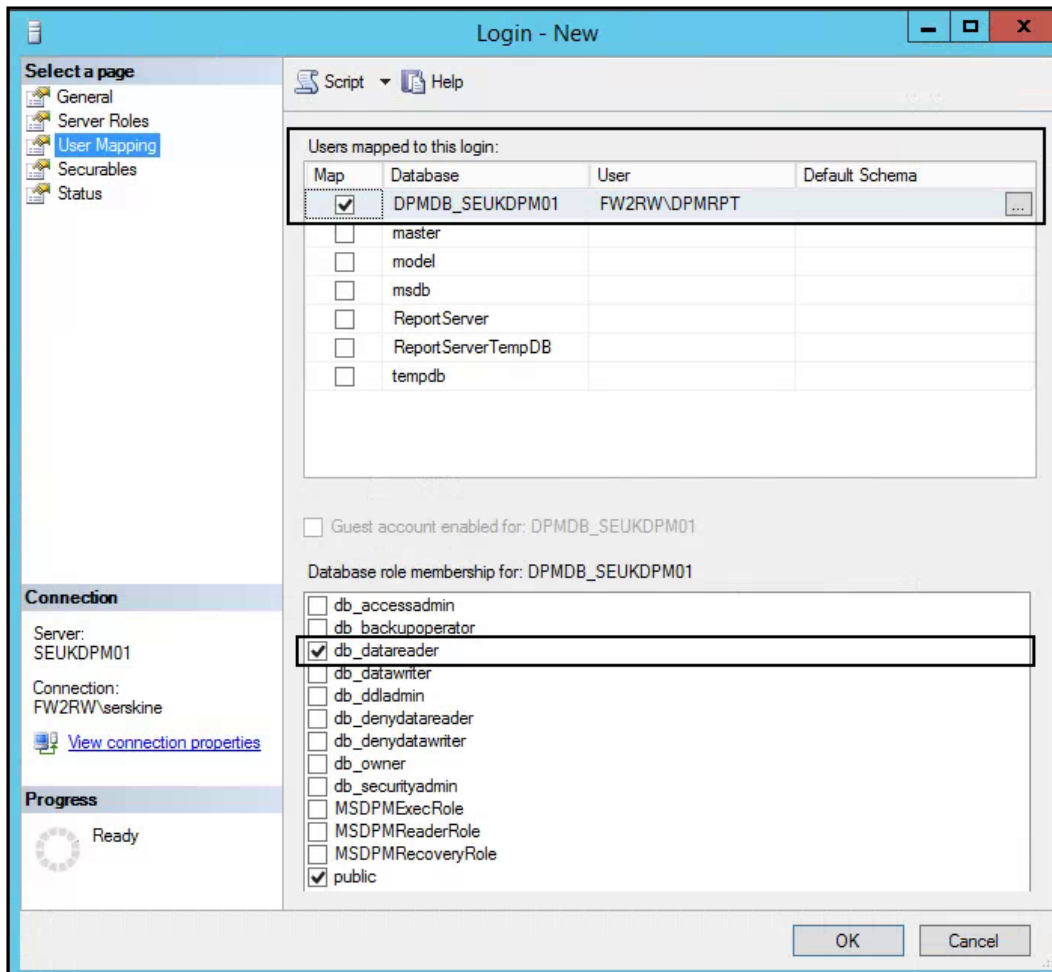
1. Create a user called, for example, `DPMRPT` in Active Directory.
2. Connect to the DPM SQL Server instance using SQL Server Management Studio.
3. Create a new Windows login. Expand **Security**. Right-click on **Logins** and select **New Login...**, as shown in this screenshot:



4. Ensure that **Windows authentication** is selected. Go to the AD account created for reporting access using the **Search...** button (<Domain Name>\DPMRPT).
5. Select DPMDDB_<Server Name> as the default database (replace <Server Name> with your DPM server name), as shown in the following screenshot:



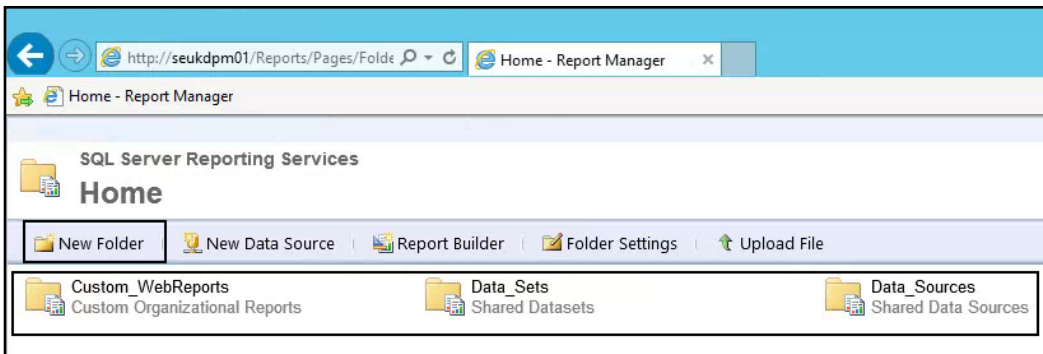
- Click on **User Mapping**. Check the DPM database and check **db_datareader**, as shown in this screenshot:



- Click on **OK** to complete the delegation.

Creating DPM Report Manager folders and performing delegation

The first recommended action you must perform to organize your reports is to create a folder structure in Report Manager. You must connect to the DPM SSRS instance to perform this task. *Chapter 3, Unpacking System Center Report Building Tools*, provides the steps you need to follow to create folders in Report Manager. The screenshot that follows shows an example of the structure used in the environment for the recipes in this chapter:



You can delegate access to the folders as discussed in *Chapter 3, Unpacking System Center Report Building Tools*.

Creating a shared data source for DPM reports

The final step in the preparation is to create a shared data source. Follow the data source creation steps in *Chapter 3, Unpacking System Center Report Building Tools*. Assuming that DPM is installed using the default settings, configure the data source as follows:

- ▶ Data source name: DPM_DATASOURCE
- ▶ Folder: Data_Sources
- ▶ Description: Shared Data source for DPM reports
- ▶ Connection string: Data Source=<DPM Server Name>;Initial Catalog=<DPM Database>

You are now ready to create your custom DPM reports.

How it works...

The reports you create depend on a data source and the report author's or consumer's access to this data source. The steps discussed in this recipe prepare your environment for authoring reports using the DPM database as the data source.

These steps work as follows:

- ▶ **Creating and delegating access to an Active Directory user account:** This is a proactive step that prepares a specific user account dedicated to retrieving information from the source database. Performing this step provides you with the option to centralize and standardize this process for all reports.
- ▶ **Creating DPM Report Manager folders and delegating access:** Perform this step to include organization in managing reports from the outset. This prevents creating reports and their dependents such as data sources in a disorganized environment.
- ▶ **Creating a shared data source** for DPM reports: A shared data source is recommended for report authoring when you intend to create multiple reports from the same database.

The DPM environment when deployed successfully will include a data source and a default folder. The steps in this recipe are recommended because they provide you with organization and control outside the default configuration.

See also

- ▶ You can find additional information on the steps discussed in this recipe in *Chapter 3, Unpacking System Center Report Building Tools*.

Preparing a dataset query for a DPM agent status report

The *Preparing the DPM reporting environment* recipe discussed the first steps you need to perform for DPM reports.

This recipe will show you how to create a dataset for the reports you create, based on the DPM data in your database.

Getting ready

You must plan to review *Chapter 3, Unpacking System Center Report Building Tools*, as a primer to this recipe.

Additionally, you must have performed the steps required to create a shared data source as discussed in the *Preparing the DPM reporting environment* recipe. You will need access to a functional DPM database with active agents to complete the steps discussed.

The following table lists the details required to complete this recipe:

Requirement or information	Description
The data source has been created and is accessible. It is stored in a folder on the reporting server called <code>Data_Sources</code> .	Use a pre-created shared data source
The reporting server URL is <code>http://<SSRS Instance>/Reports</code> .	The reporting server runs Report Manager
The DPM views with the required data are <code>VW_DPM_Agents</code> , <code>VW_DPM_ProtectedDataSource</code> , and <code>VW_DPM_ProtectedGroup</code> .	DPM views containing the required report data
The required fields are <code>ServerName</code> , <code>AgentVersion</code> , <code>AllocatedSpace</code> , <code>UsedSpace</code> , <code>DPMServer</code> , and <code>ProtectionGroupName</code> .	The required columns in the report
You need a column connecting the <code>ServerName</code> and <code>ProtectionGroupID</code> views.	The column is common to the views and used as the SQL JOIN
You need a user with the ability to connect to the DPM database using Microsoft SQL Server Management Studio (for example, <code><Domain Name\DPMRPT></code>).	The tool and the user required to connect and create the dataset query

This table will serve as the input for the tasks required to complete this recipe.

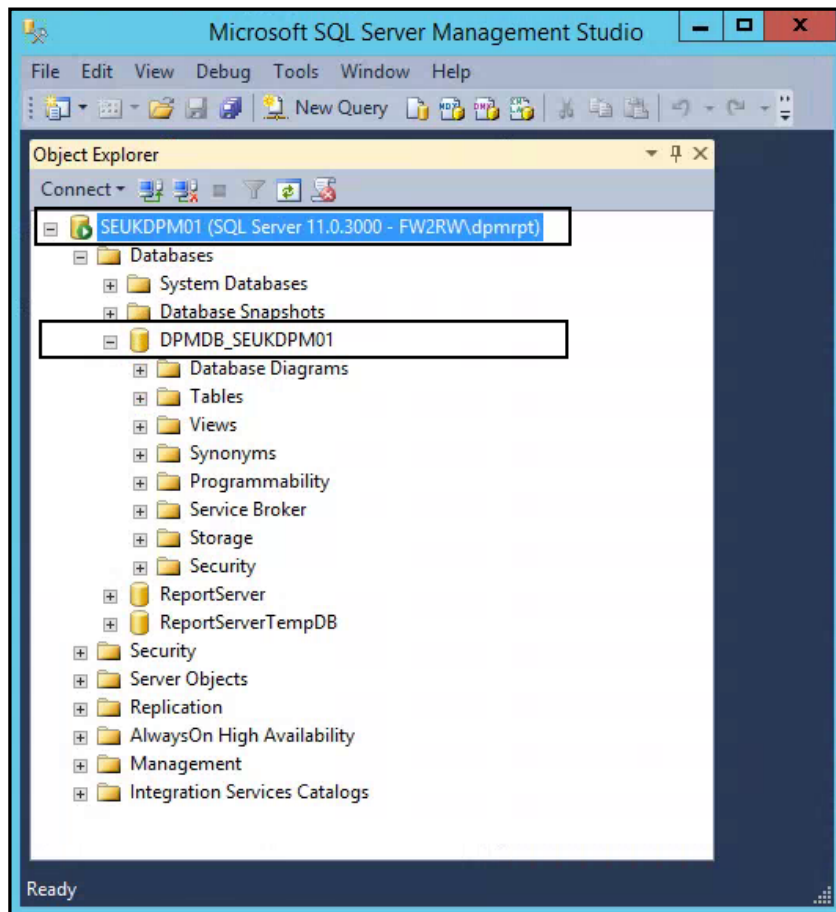
How to do it...

The steps to complete this recipe are as follows.

Creating a dataset

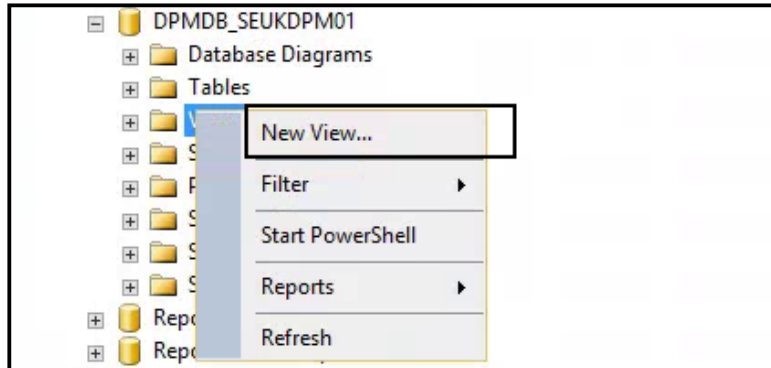
The dataset is the result of a query to the data source (the DPM database in this example), and in this case, it is created in the **Microsoft SQL Server Management Studio (MSSMS)** application. Perform the following steps:

1. Launch the MSSMS application and connect to the DPM SQL Server instance using the report user account (you must have created and delegated this account as discussed in the previous recipe), as shown in this screenshot:

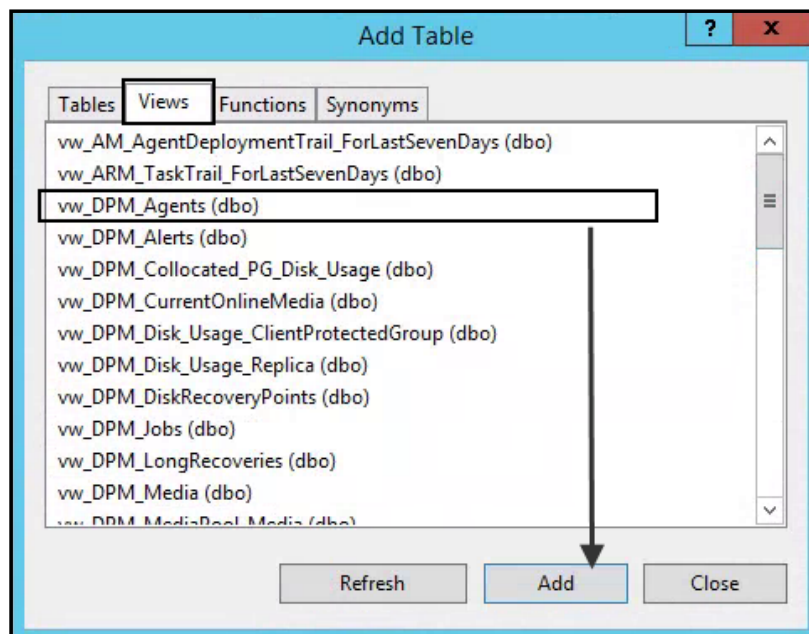


2. Expand **Databases**. Then expand `DPMDB_<DPMSERVERNAME>` (replace `<DPMSERVERNAME>` with your DPM server name).

3. Right-click on **Views** and select **New View...**, as shown in the following screenshot:

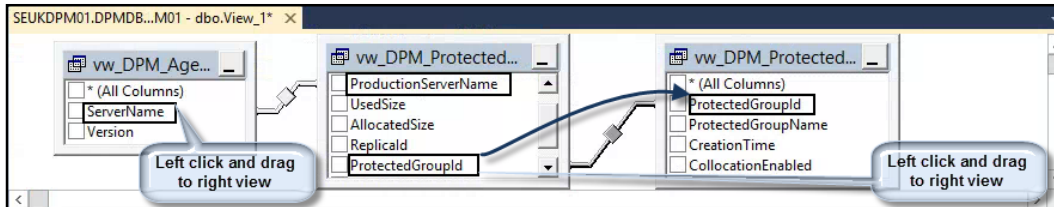


4. The **Add Table** dialog box is launched. Select the following views one at a time with the **Add** button: **vw_DPM_Agents (dbo)**, **vw_DPM_ProtectedDataSource(dbo)**, and **vw_DPM_ProtectedGroup (dbo)**. Click on **Close** after completing all three view selections, as shown in this screenshot:

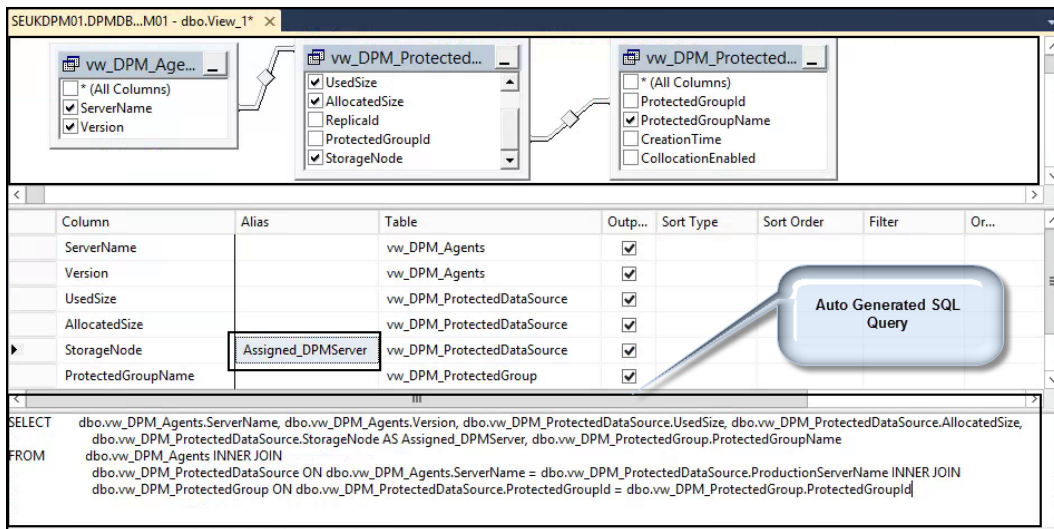


Creating Reports for System Center Data Protection Manager

5. In the middle pane of MSSMS, ensure that you can see the three views (you may have to drag the objects and resize the windows). Click and hold the **ServerName** column from the **vw_DPM_Agents(dbo)** view, and drag it onto the **ProductionServerName** column in the **vw_DPM_ProtectedDataSource** view.
6. Click and hold the **ProtectedGroupId** column from the **vw_DPM_ProtectedDataSource** view, and drag it onto the **ProtectedGroupId** column in the **vw_DPM_ProtectedGroup** view, as shown in the following screenshot:



7. Check the column names from the three views, as follows:
 - ❑ **vw_DPM_Agent(dbo): ServerName and Version**
 - ❑ **vw_DPM_ProtectedDataSource (dbo): UsedSize, AllocatedSize, and StorageNode**
 - ❑ **vw_DPM_ProtectedGroup (dbo): ProtectedGroupName**
8. In the middle pane, in the **Alias** column, type **Assigned_DPMServer** next to the **StorageNode** column, as shown in this screenshot:



9. Test the query using the execute icon (red exclamation mark), as shown in the following screenshot:

The screenshot displays the SQL Server Enterprise Manager interface. At the top, there are three view selection panes. The first pane shows columns for 'vw_DPM_Age...' including 'ServerName' and 'Version'. The second pane shows columns for 'vw_DPM_Protected...' including 'UsedSize', 'AllocatedSize', 'Replicad', 'ProtectedGroupd', and 'StorageNode'. The third pane shows columns for 'vw_DPM_Protected...' including 'ProtectedGroupd', 'ProtectedGroupName', 'CreationTime', and 'CollocationEnabled'. Below these panes is a query editor with the following SQL query:

```

FROM
  dbo.vw_DPM_ProtectedDataSource.StorageNode AS Assigned_DPMServer,
  dbo.vw_DPM_ProtectedGroup.ProtectedGroupName]
  dbo.vw_DPM_Agents INNER JOIN
  dbo.vw_DPM_ProtectedDataSource ON dbo.vw_DPM_Agents.ServerName = dbo.vw_DPM_ProtectedDataSource.ProductionServerName INNER JOIN
  dbo.vw_DPM_ProtectedGroup ON dbo.vw_DPM_ProtectedDataSource.ProtectedGroupd = dbo.vw_DPM_ProtectedGroup.ProtectedGroupd

```

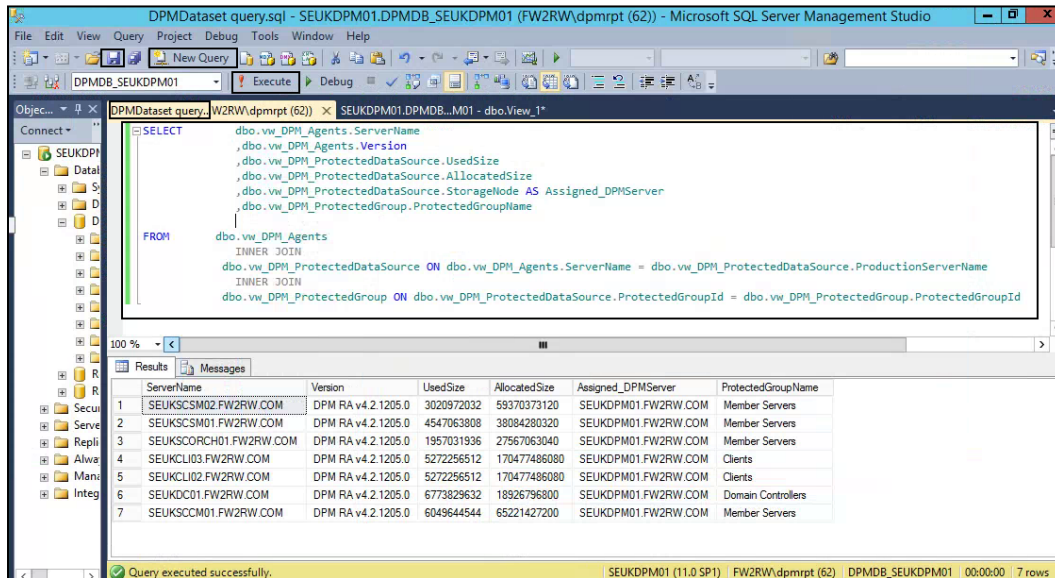
Below the query editor is a results grid with the following columns: ServerName, Version, UsedSize, AllocatedSize, Assigned_DPMServer, and ProtectedGroupName. The grid contains seven rows of data:

ServerName	Version	UsedSize	AllocatedSize	Assigned_DPMServer	ProtectedGroupName
SEUKSCSM02.FW2R...	DPM RA v4.2.1205.0	3020972032	59370373120	SEUKDPM01.FW2RW...	Member Servers
SEUKSCSM01.FW2R...	DPM RA v4.2.1205.0	4547063808	38084280320	SEUKDPM01.FW2RW...	Member Servers
SEUKSCORCH01.FW2...	DPM RA v4.2.1205.0	1957031936	27567063040	SEUKDPM01.FW2RW...	Member Servers
SEUKCLI03.FW2RW.C...	DPM RA v4.2.1205.0	5272256512	170477486080	SEUKDPM01.FW2RW...	Clients
SEUKCLI02.FW2RW.C...	DPM RA v4.2.1205.0	5272256512	170477486080	SEUKDPM01.FW2RW...	Clients
SEUKDC01.FW2RW.C...	DPM RA v4.2.1205.0	6773829632	18926796800	SEUKDPM01.FW2RW...	Domain Controllers
SEUKSCCM01.FW2R...	DPM RA v4.2.1205.0	6049644544	65221427200	SEUKDPM01.FW2RW...	Member Servers

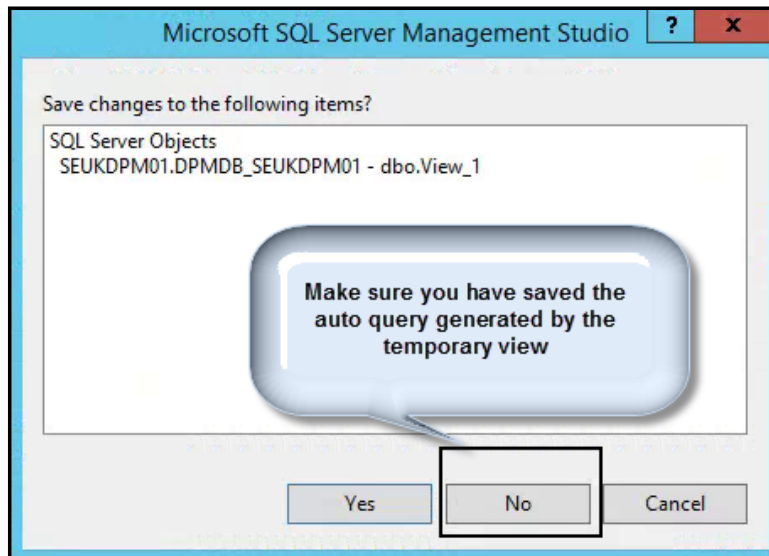
The status bar at the bottom indicates '3 of 7' rows and 'Cell is Read Only'.

Creating Reports for System Center Data Protection Manager

10. In the middle pane, select the autogenerated SQL query. Copy the details. While still in MSSMS, click on the **New Query** icon and paste the SQL query you copied in the new query window. Tidy up the text as shown in the following screenshot, taking care not to delete any of the syntax. Test the query and save it in a local folder.



11. Make sure you have saved the query. Close MSSMS, and when prompted, do not save the view, as shown in this screenshot:



This completes the steps you need to create the dataset query.

How it works...

The steps you followed in this recipe are a simplified approach to creating a SQL query to read information from the DPM database. The steps require that you plan for the data you need and document the views and columns you need for your query.

When you create a report using a report-authoring tool, you need to supply a query to retrieve the data (dataset). When you use MSSMS to create a view, the SQL query is automatically generated. You can use this approach to create your simple queries for the purpose of reports.

In tools such as Report Builder, you can create a query graphically similar to the steps you followed in MSSMS. The difference here is you create and test the query outside the authoring tool.

There's more...

The query has two disk-sized columns that return a result set in bytes. The values are easier to read and manage when read in megabytes (MB) or gigabytes (GB).

Converting byte values into MB or GB values

You can perform the conversion of byte values into MB or GB in the query using the SQL ROUND function. When this function is applied to the SQL query to convert the sizes into GB, it will change the SQL query to the following:

```
SELECT    vw_DPM_Agents.ServerName
          ,vw_DPM_Agents.Version
          ,ROUND
          (vw_DPM_ProtectedDataSource.UsedSize /1024 /1024 /1024 ,2) AS
          UsedSpaceGB
          ,ROUND
          (vw_DPM_ProtectedDataSource.AllocatedSize /1024 /1024 /1024 ,2) AS
          AllocatedSizeGB
          , vw_DPM_ProtectedDataSource.StorageNode AS Assigned_DPMServer
          ,dbo.vw_DPM_ProtectedGroup.ProtectedGroupName

FROM      vw_DPM_Agents INNER JOIN
          dbo.vw_DPM_ProtectedDataSource ON dbo.vw_DPM_Agents.ServerName =
          dbo.vw_DPM_ProtectedDataSource.ProductionServerName
          INNER JOIN  dbo.vw_DPM_ProtectedGroup ON
          dbo.vw_DPM_ProtectedDataSource.ProtectedGroupId =
          dbo.vw_DPM_ProtectedGroup.ProtectedGroupId
```

You can find additional information on the ROUND function at [http://technet.microsoft.com/en-us/library/ms175003\(v=sql.110\).aspx](http://technet.microsoft.com/en-us/library/ms175003(v=sql.110).aspx).

See also

- ▶ Detailed information on the DPM database views can be found in the official online documentation (Microsoft TechNet) at <http://technet.microsoft.com/en-us/library/hh757766.aspx>

Creating a DPM agent version and a disk space report

This recipe will show you how to use Report Builder to create a report using DPM data. The report will show the agent version and the amount of disk space utilized for replicas versus the allocated disk space.

Getting ready

You must plan to review *Chapter 3, Unpacking System Center Report Building Tools*, as a primer to this recipe. Additionally, you must have performed the steps required to create a shared data source and the dataset query discussed in the *Preparing the DPM reporting environment* and *Preparing a dataset query for a DPM agent status report* recipes in this chapter.

The following table lists the details required to complete this recipe:

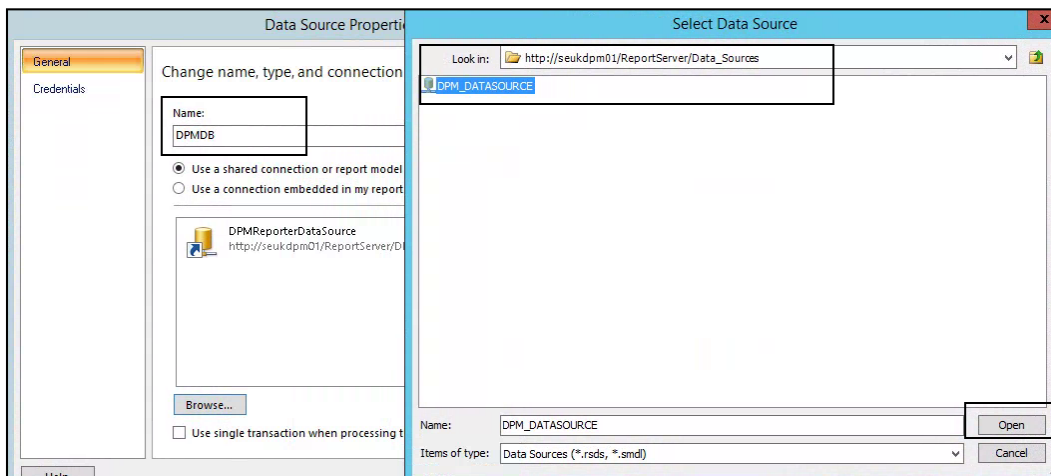
Requirement or information	Description
You need to make sure a data source has been created and should be accessible. It is stored in a folder on the reporting server called <code>Data_Sources</code> .	Use a pre-created shared data source
Ensure the reporting server URL is <code>http://<SSRS Instance>/Reports</code> .	The reporting server runs Report Manager
Ensure the dataset query is prepared and tested.	See the steps for creating the required dataset in the <i>Preparing a dataset query for a DPM agent status report</i> recipe
You need a user with the ability to connect to the DPM database using Report Builder.	Tool and user to connect and create the report
You will need the report name.	The name of the report is <code>DPM Agent Details</code>

This table will serve as the input for the tasks required to complete this recipe.

How to do it...

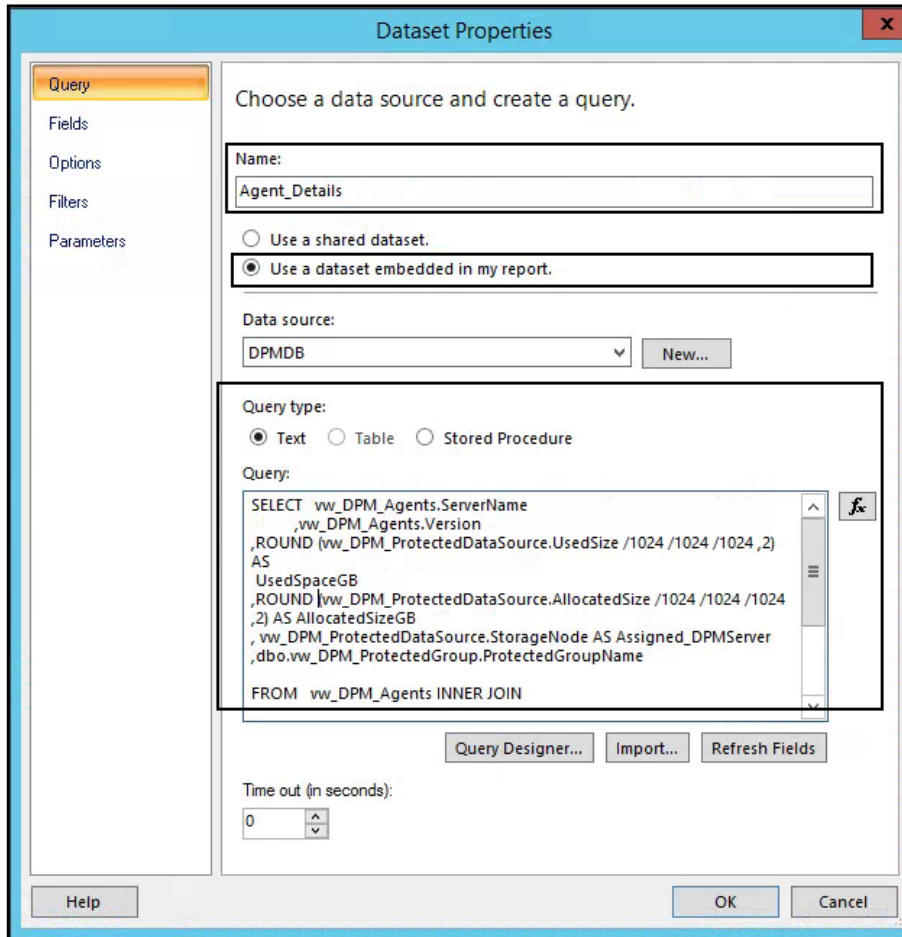
The following steps will show you how to create the report for this recipe:

1. Launch the Report Builder application using the Report Manager website option by navigating to `http://<ServerName>/Reports` (substitute the URL with your specific environment values) and click on **Report Builder**.
2. Close the **Getting Started** page. Ensure that the **Home** tab is selected. Under **Report Data**, click on **New** and **Select Data Source....**
3. On the **General** tab, type `DPMDB` for the data source in the **Name:** field. Select **Use a shared connection or report model**.
4. Click on the **Browse** button. Navigate to the shared data source in the **Data_Sources** folder. Select **DPM_DATASOURCE** and click on **Open** (you must have created the data source in the *Preparing the DPM reporting environment* recipe of this chapter), as shown in the following screenshot:



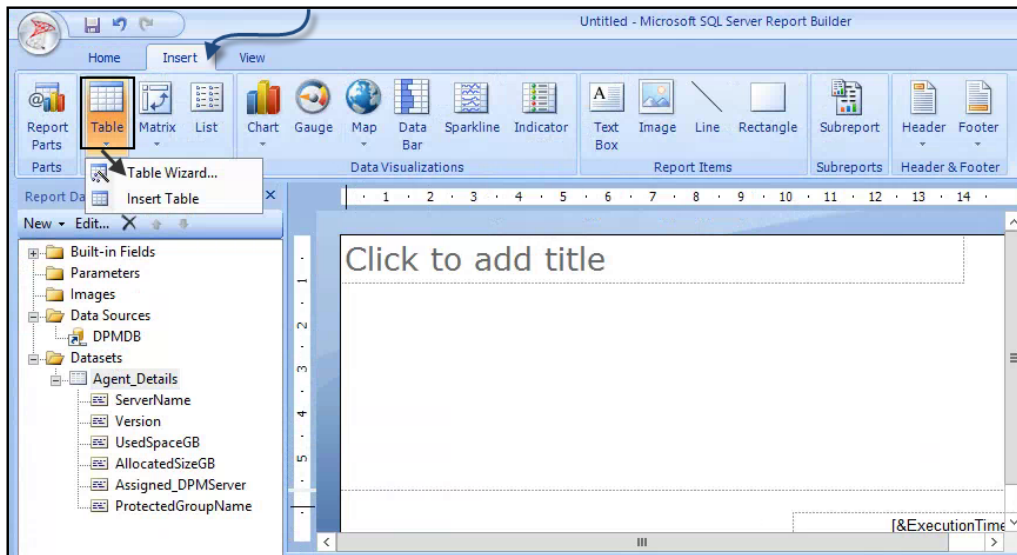
5. Click on **Test Connection**, and if it succeeds, click on **OK**.
6. Under **Report Data**, click on **New** and select **Dataset....**
7. On the **Query** tab, type `Agent_Details` for the dataset in the **Name:** field. Select **Use a dataset embedded in my report**.
8. Under **Data source:**, select the **DPMDB** data source you created.

9. In **Query type:**, paste the dataset query (from the file you saved in the previous recipe) in the textbox, as shown in the following screenshot:

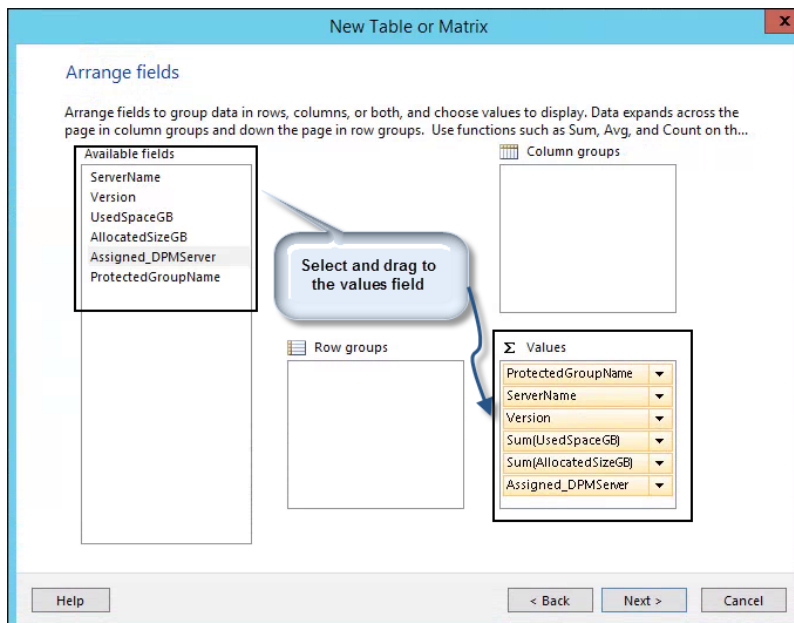


10. Click on **OK**. Verify that you have a data source called **DPMDB** and a dataset called **Agent_Details**.

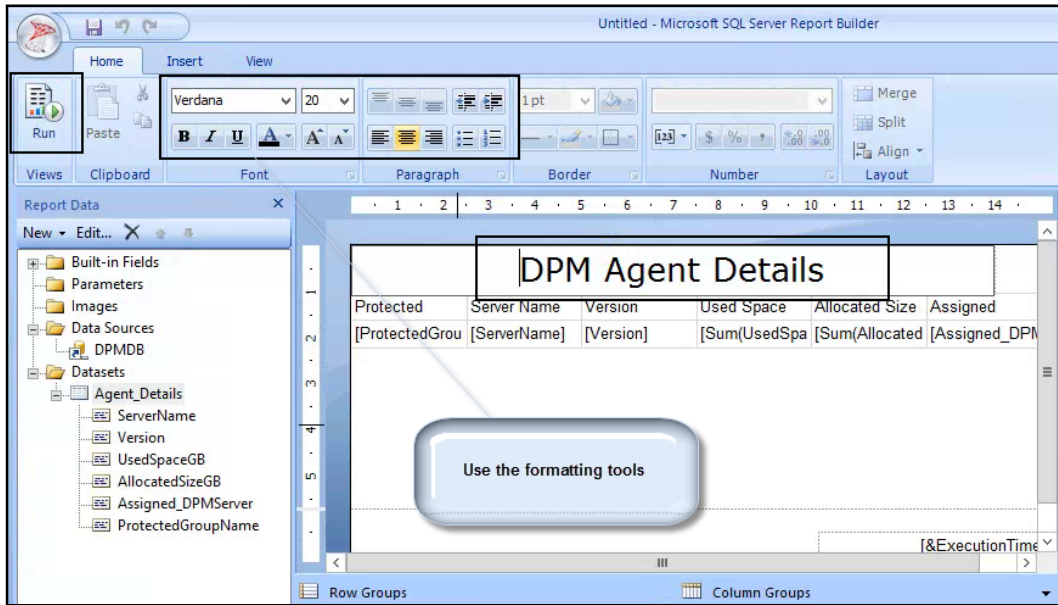
- Click on the **Insert** tab and select **Table**. Click on **Table Wizard**, as shown in this screenshot:



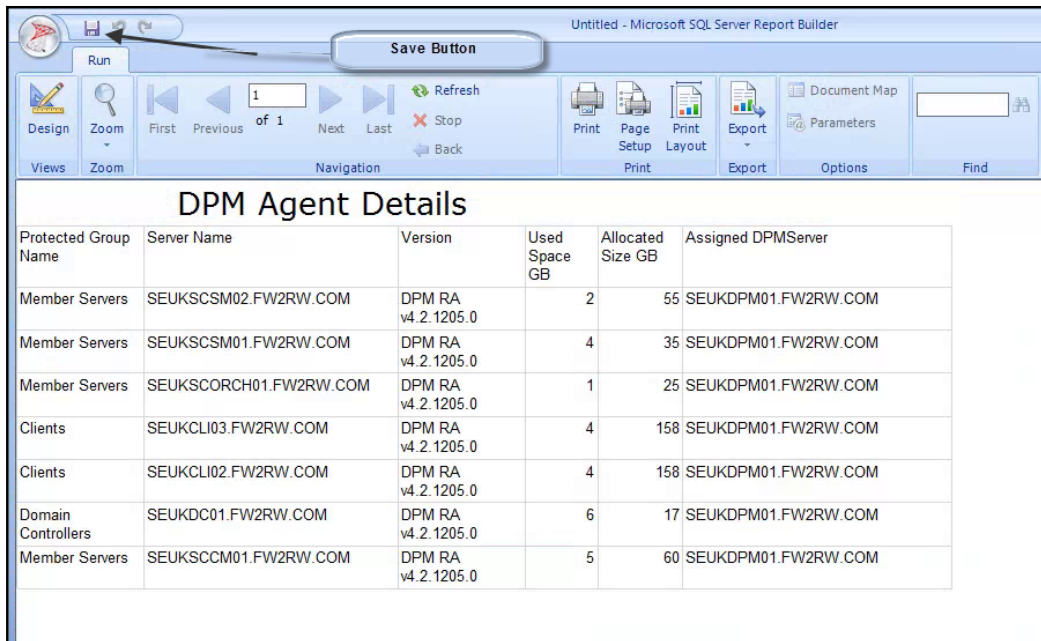
- Select **Agent_Details** under the **Choose dataset** option and click on **Next**.
- Select each of the **Available fields** fields in turn and drag them into the **Values** box, as shown in the following screenshot:



14. Click on **Next**. Again, click on **Next** on the **Choose the Layout** page.
15. Select **Generic** and click on **Finish**.
16. Click on the title field in the middle pane and type a name for the report, for example, **DPM Agent Details**, as shown in this screenshot:



17. Click on the **Run** button to test the report.
18. Click on the **Design** button to go back and edit the layout and column sizes, as shown in the following screenshot:



19. Click on the **Save** button. Navigate to the required folder in Report Manager, provide a name for the report (for example, **DPM_Agent_Details**), and click on **Save**.

The new report is saved to the SSRS Server you are connected to. It can be run and edited by connecting to the Report Manager website, as shown in this screenshot:



How it works...

The recipe shows you how to create a report using the dataset you created by connecting to a DPM database data source. The steps demonstrate the connections between the preparation, delegation, data source creation, and dataset queries that are required to create the report.

The Report Builder interface provides you with formatting tools and wizards to transform the data into your desired visual representation.

The final part of the recipe discusses how to save the report. Saved reports can be edited, downloaded, and even saved with a different name to create a new version of an existing report.

There's more...

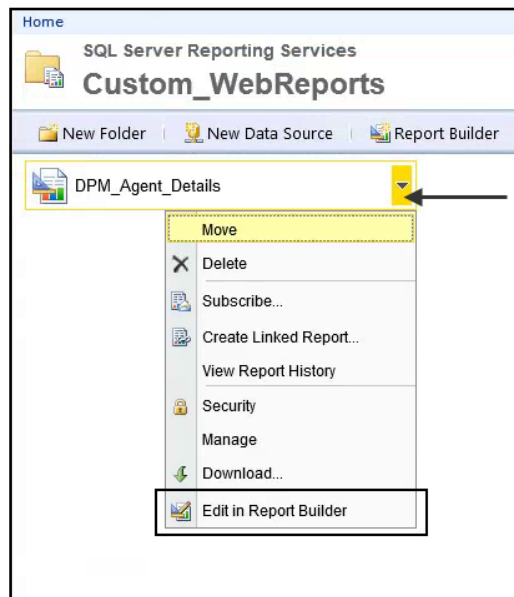
What we covered was a basic report using the generic template (no formatting). You may want to enhance the report and add a bit of style and color. Here are three examples of formatting you may want to apply:

- ▶ Change the color and font of the title
- ▶ Change the background color of the headers
- ▶ Add a size indicator bar to the used space column

Changing the color and font of the title

Perform the following steps:

1. Navigate to the report location on the Report Manager website. Click on the arrow to the right of the report name and select **Edit in Report Builder**, as shown in the following screenshot:

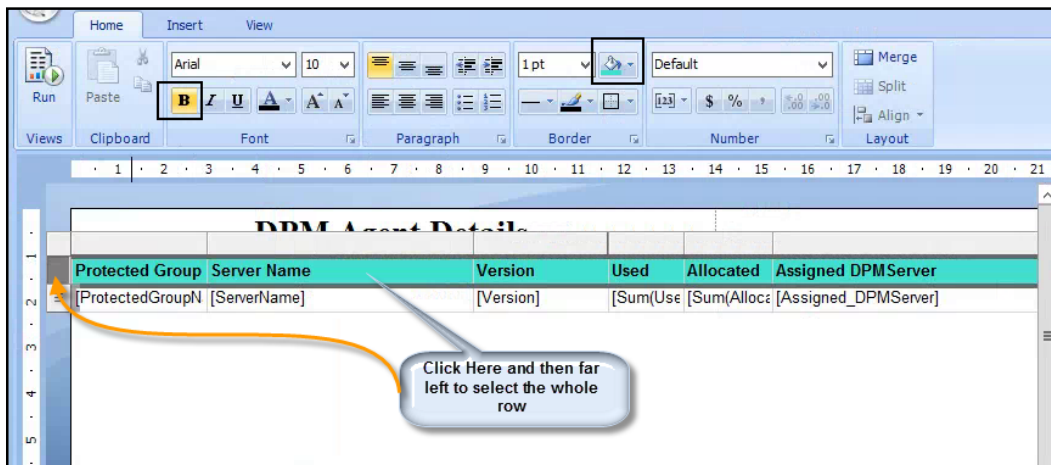


2. Click on the report title. Highlight the text and change the font to **Times New Roman**.
3. Click on the bold button (**B**).
4. Click on the **Save** button.

Changing the background color of the headers

Perform the following steps:

1. Click on the space next to one of the column headers (for example, **Server Name**).
2. Click on the far-left row tile to select the entire row; select a different fill color using the formatting tools. Click on the bold button and change the background fill, as shown in the following screenshot:



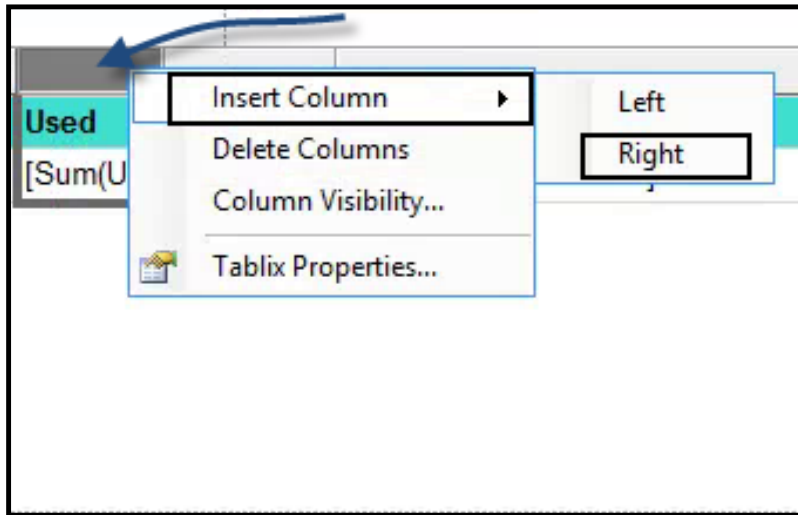
3. Click on the **Save** button.

Adding a size indicator bar to the used space column

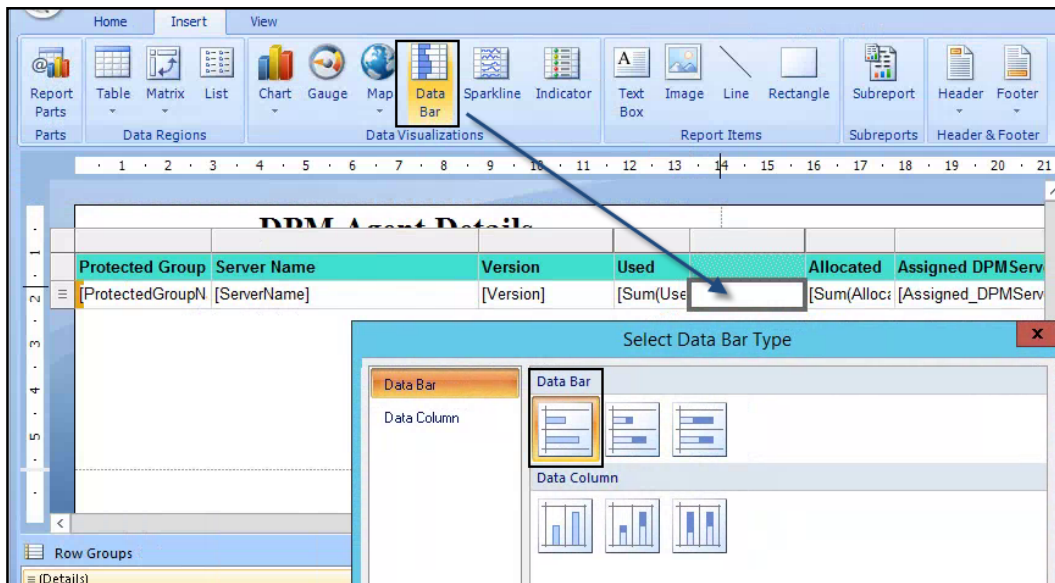
Perform the following steps:

1. Click on the space next to the **UsedSizeGB** column headers. Then, click on the top of the column to highlight the whole column.

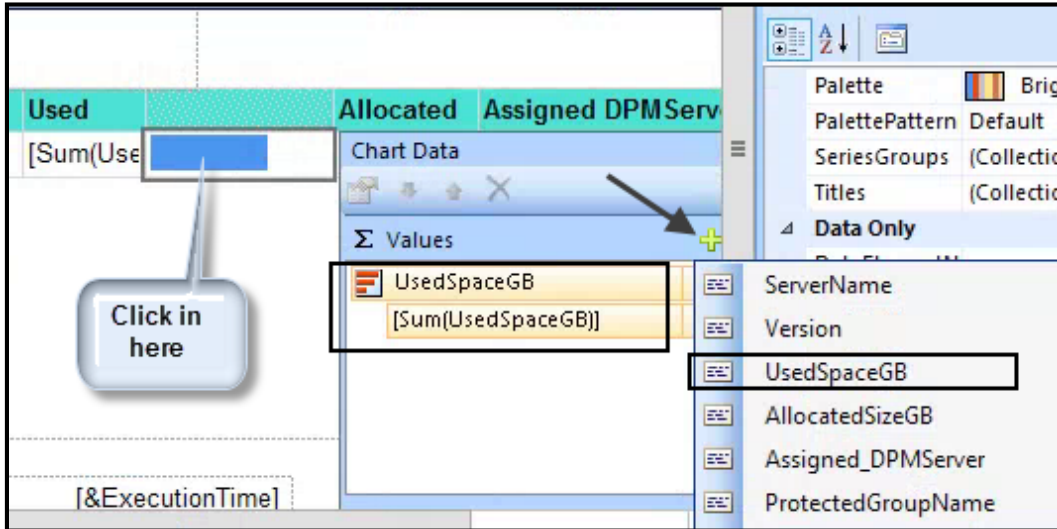
2. Right-click above **Used** and select **Insert Column**. Select **Right**, as shown in this screenshot:



3. Click on the **Insert** tab. Select **Data Bar**. Click on the block next to the **UsedSizeGB** value (the property with "[Sum...]"), as shown in the following screenshot:



- Click on the blue indicator bar that is inserted into the column. Click on the + sign in the **Values** area of the **Chart** data properties box and select **UsedSpaceGB**, as shown in this screenshot:



- Click on the **Save** button.
- Run the report to validate the three modifications, as shown in the following screenshot:

Home > Custom_WebReports > DPM_Agent_Details Home | My

1 of 1 100% Find | Next

DPM Agent Details

Protected Group Name	Server Name	Version	Used Space GB	Allocated Size GB	Assigned DPM Server
Member Servers	SEUKSCSM02.FW2RW.COM	DPM RA v4.2.1205.0	2	55	SEUKDPM01.FW2RW.COM
Member Servers	SEUKSCSM01.FW2RW.COM	DPM RA v4.2.1205.0	4	35	SEUKDPM01.FW2RW.COM
Member Servers	SEUKSCORCH01.FW2RW.COM	DPM RA v4.2.1205.0	1	25	SEUKDPM01.FW2RW.COM
Clients	SEUKCLI03.FW2RW.COM	DPM RA v4.2.1205.0	4	158	SEUKDPM01.FW2RW.COM
Clients	SEUKCLI02.FW2RW.COM	DPM RA v4.2.1205.0	4	158	SEUKDPM01.FW2RW.COM
Domain Controllers	SEUKDC01.FW2RW.COM	DPM RA v4.2.1205.0	6	17	SEUKDPM01.FW2RW.COM
Member Servers	SEUKSCCM01.FW2RW.COM	DPM RA v4.2.1205.0	5	60	SEUKDPM01.FW2RW.COM

Creating a backup status report with a community template

The recommended practice for creating reports is to use vendor-supplied views instead of database tables. The use of tables is not officially supported by Microsoft. DPM, however, is an exception to this recommendation, based on the versions leading up to 2012 R2. This exception is due to the limited amount of views available to create the right kind of reports.

Two community gurus have combined their talent to create and provide a reporting solution that targets the DPM tables with valuable data for organization reports. Steve Buchanan (System Center MVP) and Brooks Lindall (business intelligence consultant) are the creators and authors of the **DPM_BackupSummary** report.

The focus of this recipe is to use this report as a basis for discussing modifications to an existing report.

Getting ready

You must plan to review *Chapter 3, Unpacking System Center Report Building Tools*, as a primer to this recipe. Additionally, you must have downloaded and followed the instructions to upload the community report to your reporting server.

The following table lists the details required to complete this recipe:

Requirement or information	Description
You must make sure a data source has been created and is accessible. It should be stored in a folder on the reporting server called <code>Data_Sources</code> , or you can use the default DPM data source.	Use a pre-created shared data source
Ensure the reporting server URL is <code>http://<SSRS Instance>/Reports</code> .	The reporting server runs Report Manager
The <code>DPM_BackupSummary</code> report must run successfully.	See the steps for installing the report in the user guide, including the solution, at http://gallery.technet.microsoft.com/DPM-Backup-Summary-Report-4533bcc6
You need to have a user with access to connect to the DPM database using Report Builder.	Tool and user required to connect and create the report

Requirement or information	Description
Color codes: <ul style="list-style-type: none"> ▶ Red: #f80000 ▶ Green: #0fab0f 	The proper red and green color codes required to modify the community report

This table will serve as the input for the tasks required to complete this recipe.


How to do it...

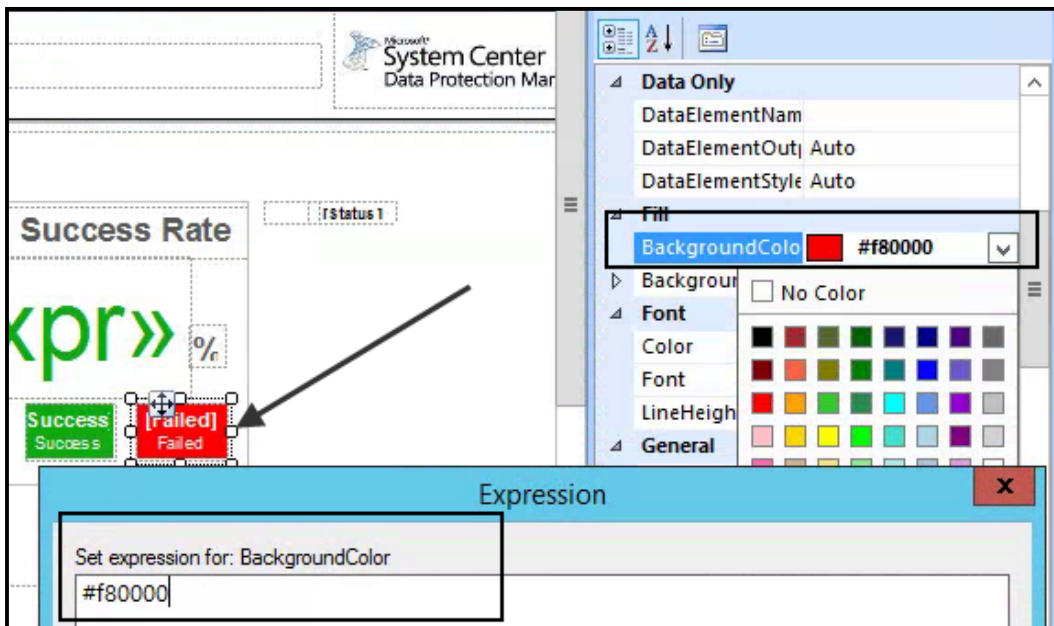
The following steps will show you how to create a new version of an existing report using a community-created report definition:

1. Launch the Report Builder application using the Report Manager website option by navigating to `http://<ServerName>/Reports` (substitute the URL with your specific environment values) and clicking on **Report Builder**.
2. Navigate to the **DPM Summary Report** location in Report Manager. Click on the arrow to the right of the report name and select **Edit in Report Builder**.
3. Click on the top-left **Report Builder** icon (next to the **Save** button). Select **Save As**.
4. Give the report a new name, for example, `MYOrg_DPMBackupSummary`.
5. In the middle pane, click on the MVP logo and then on **delete**. Select the footer textbox and delete it.
6. Click on the blue bar of the **Number of Backups by Server** chart. In the properties area, ensure that **TaskID** is displayed, and click on the little arrow next to **Color**, as illustrated in this screenshot:

The screenshot shows the Report Builder interface. The main chart area displays a horizontal bar chart titled "Number of Backups by Server". The chart has six rows representing servers (A-F) and two data series. The values for each server are: Server F (79, 68), Server E (61, 12), Server D (18, 37), Server C (24, 87), Server B (85, 22), and Server A (98, 55). The x-axis ranges from 0 to 150. The properties pane on the right is open to the "TaskID" section. The "Color" property is set to "#0fab0f". A color palette is visible below the properties pane, and an arrow points to the "Expression..." link.

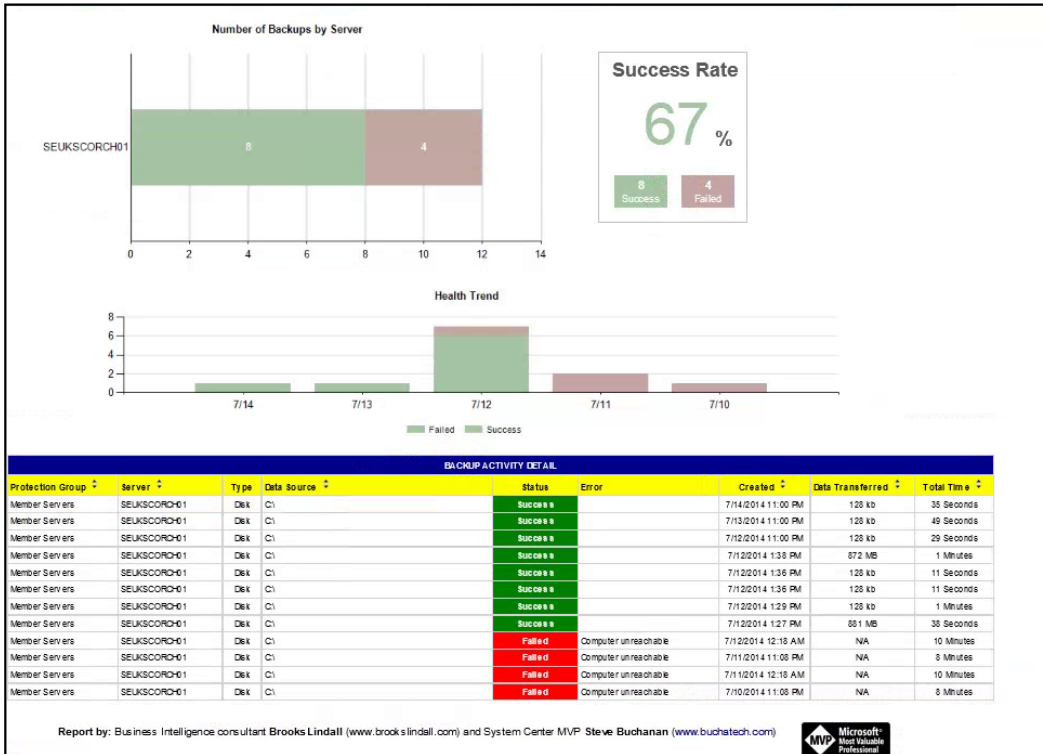
7. Click on **Expression**. Change the values to #f80000 and #0fab0f to represent the shades of red and green respectively, as shown in the following screenshot. Click on **OK**.
8. Click on the values in the **Success Rate** textbox next to the chart in turn, and change the values to the correct shades of red and green.

 To select the text properties, click on the **ESC** option to toggle the property selection.



9. Repeat the color change steps you performed with the first chart on health trends.

Here is a screenshot that shows the community report before the modification:



The following screenshot shows the community report after the modification:



How it works...

This recipe shows how to modify an existing report. The report definition was supplied by two community experts. The objective of this recipe is to show how you can make visualization changes to existing reports without the need to recreate the datasets.

The Report Builder interface provides you with formatting tools and multiple property editing options.

You must practice working with these settings on copies of existing reports until you get the desired result for your needs.

See also

- ▶ Microsoft released a new System Center Operations Manager Management Pack to improve the DPM report in February 2015. You can download the management pack from <http://www.microsoft.com/en-us/download/details.aspx?id=45525>.

7

Creating Reports for System Center Service Manager and Orchestrator

In this chapter, we will cover the following recipes:

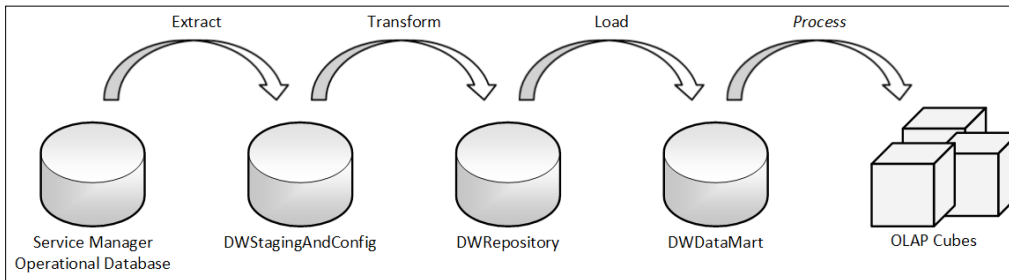
- ▶ Using out-of-the-box Service Manager reports and cubes
- ▶ Understanding the Service Manager Data Warehouse data mart
- ▶ Creating Work Item and Configuration Item reports
- ▶ Creating reports using OLAP cubes
- ▶ Creating reports using the operational database
- ▶ Accessing data using Microsoft Excel
- ▶ Extending the Service Manager Data Warehouse
- ▶ Creating Orchestrator runbook reports

Introduction

This chapter will walk you through the various options of creating reports for System Center Service Manager and System Center Orchestrator.

System Center Service Manager

System Center Service Manager contains a Data Warehouse database that you can use to gain insight into the data stored in Service Manager. Service Manager periodically transfers data that is stored in the operational database to the Service Manager Data Warehouse databases through a process called **extract, transform, and load (ETL)**. The processes required to keep the data in the Data Warehouse in sync with the operational database are controlled by the Service Manager Data Warehouse Management Server role, as shown in the following image:



The reasons to transfer data to the Service Manager Data Warehouse databases are as follows:

- ▶ Offload data from the Service Manager operational database to improve performance
- ▶ Provide long-term storage of historical data stored in Service Manager
- ▶ Optimize and provide data for reporting

The `DWDataMart` database is the database used for all reporting needs. In addition, Service Manager comes with a set of predefined OLAP cubes to facilitate advanced analysis of data. An OLAP cube is a multidimensional data structure that is optimized to aggregate and analyze a large amount of data while also allowing access to the most granular information. The SCSM Data Warehouse leverages Microsoft SQL Server Analysis Services to provide OLAP cubes to end users.

With both the `DWDataMart` database and the OLAP cubes, Service Manager provides powerful means to create simple reports, as well as gain advanced insight into the data by analyzing it from multiple perspectives.

System Center Orchestrator

System Center Orchestrator does not offer any out-of-the-box reporting capabilities. However, the data stored in the Orchestrator SQL database contains useful data about runbooks and jobs that you can use in custom reports.

Using out-of-the-box Service Manager reports and cubes

In this recipe, we will show you how to navigate through the reporting section of the Service Manager console and how to work with the built-in reports that are shipped with Service Manager. We will also show you how you can access the OLAP cubes using Microsoft Excel for ad hoc reporting.

Getting ready

Before working with Service Manager reports, you must have already installed a Service Manager Data Warehouse Management Server and registered it with the Service Manager Server installation. You also need to ensure that the initial synchronization of the management packs is complete and the ETL jobs have run. This process normally takes several hours to complete. You also need to wait until all data warehouse jobs of the **cube processing** type have finished. This ensures that the cubes have processed all the data.

You must have Microsoft Excel installed on the computer that you use to perform these steps. The computer can either be the Service Manager Management server or a client with the Service Manager console installed.

How to do it...

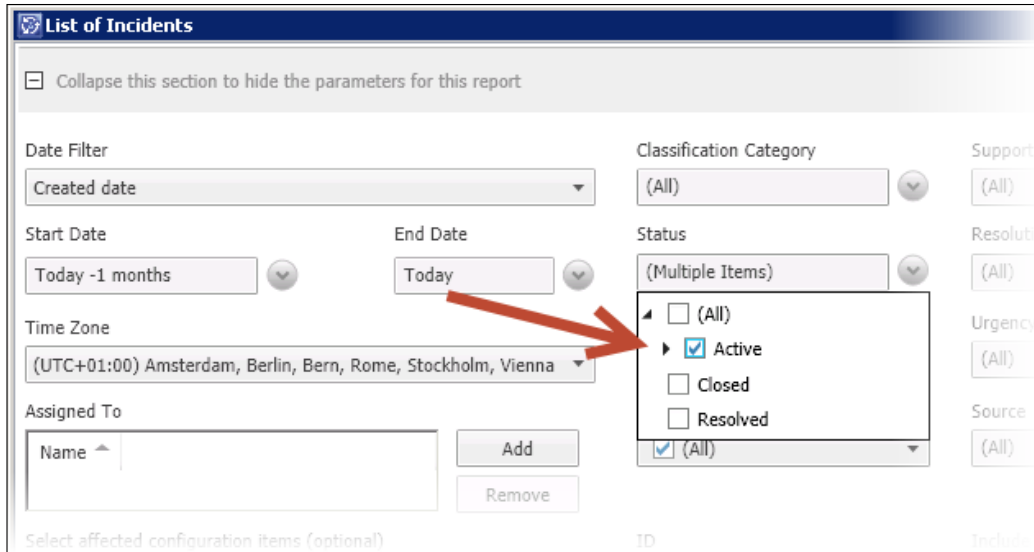
In this section, we will demonstrate how you can access the out-of-the-box Service Manager reports and OLAP cubes.

Using out-of-the-box Service Manager reports

Perform the following steps:

1. Navigate to the **Reporting** section of the Service Manager console.
2. Under the **Reports** node, you will see folders that contain predefined reports for different types of Work Items and Configuration Items. Click on the **Incident Management** folder.

3. You will see the built-in reports for incident management listed in the view. In this example, we want to retrieve a list of all the incidents with a classification of hardware problems. Click on the **List of Incidents** report and then on **Run Report** from the taskbar.
4. In the **Parameters** section, define any filters you would like to apply to the report. For instance, under **Status**, uncheck the **(All)** option and select **Active** to display only the active incidents, as shown in the following screenshot:



5. Click on the **Run Report** option from the taskbar to generate and view the report in the console.
6. You can export reports to various file formats. Click on the **disk** icon and choose the desired format to export the reports.

Using out-of-the-box Service Manager OLAP cubes

Perform the following steps:

1. Navigate to the **Data Warehouse** section of the Service Manager console.
2. Click on **Cubes**. You will see the list of all OLAP cubes available in Service Manager.
3. Click on **Service Manager WorkItems Cube** and then click on the **Analyze Cube in Excel** option from the taskbar. This will start Microsoft Excel and connect to the OLAP cube automatically.

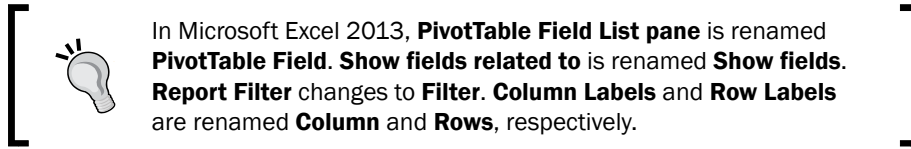


Manually connecting to the OLAP cubes

If you do not have the Service Manager console installed on the computer, you can manually configure the connection to the OLAP cubes. Start Microsoft Excel. On the **Data** ribbon, click on **From Other Sources** and choose **From Analysis Services**. Type the name of the computer that hosts SQL Server Analysis Services, click on **Next**, and choose the desired cube from the list of available cubes in DWASDataBase. Click on **Finish** and then on **OK**.

4. Make the following configuration in **PivotTable Field List** pane:
 1. Select **IncidentDim** under **Show fields related to**.
 2. Drag the **IncidentDimCount** element from the **IncidentDim** entry and drop it under **Values**.
 3. Scroll down to the **IncidentDim_IncidentImpact** entry in the **Show fields related to** list.
 4. Expand **More Fields** and drag the **IncidentImpactValue** element and drop it under **Column Labels**.
 5. Scroll down to the **IncidentDim_IncidentStatus** entry in the **Show fields related to** list.
 6. Expand **More Fields**, drag the **IncidentStatusValue** element, and drop it under **Row Labels**, as shown in the following screenshot:

IncidentDimCount	Column Labels	C	D	E	F	G	H	I
Row Labels	High	Low	Medium	Grand Total				
Active	1	23	6	30				
Closed	107	18904	2348	21359				
Pending		42	9	51				
Resolved	2	475	49	526				
Grand Total	110	19444	2412	21966				



How it works...

Service Manager uses SQL Server Reporting Services to provide its reporting capabilities. The folders and reports that are listed under the **Reporting** section of the Service Manager console represent how the reports are organized in SQL Server Reporting Services.

When a report is opened in the console, Service Manager loads the report definition and displays its parameters to allow for filtering data in the report. The options and controls available to filter data are defined using management packs.

The Service Manager OLAP cubes are stored in SQL Server Analysis Services. You can use tools such as Microsoft Excel or other Business Intelligence tools to access and analyze the data stored in the cubes.

There's more...

The in-built Service Manager reports and the custom reports you create can be accessed without using the console.

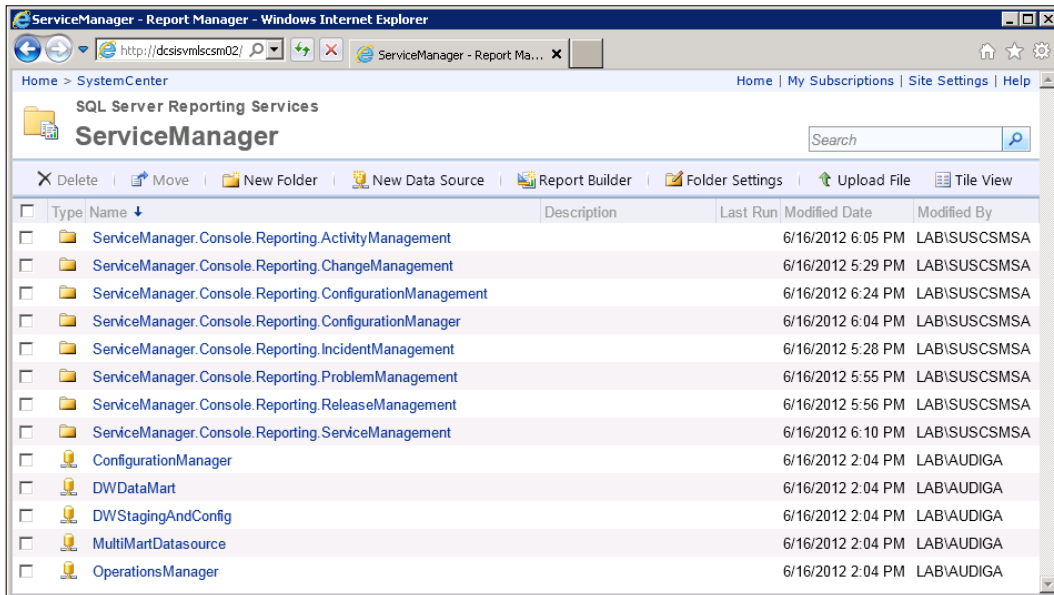
Accessing reports from a browser

The Service Manager reports can also be viewed through a web browser. This allows people who do not work with the Service Manager console to access reports.

Open your web browser and navigate to `http://[SCSMDWSQL]/Reports`.

Replace `[SCSMDWSQL]` with the fully qualified domain name of the SQL server that is used for reporting. If SQL Server Reporting Service is running as a named instance, the syntax of the URL will be `http://[SCSMDWSQL]/Reports_[InstanceName]`.

When you click on the `SystemCenter` folder and then on the `ServiceManager` folder, you will see the same folder structure that was previously displayed in the Service Manager console. You might want to switch to **Tile View** to get a better overview of the folder and file structure, as shown in the following screenshot:



You will notice that the folder and file names as well as the controls used to define your report parameters are much less comprehensive than when the reports are consumed through the Service Manager console.

When you use the web browser to navigate the reports, SQL Server Reporting Services shows the actual file and folder names, and also renders the default controls for the report parameters.

This behavior can be overridden when accessing reports through the Service Manager console, using directives in management packs.

If you want to provide reports to users through the web browser, we recommend that you create custom reports for your end users who do not make use of advanced management pack features but are instead tailored to be consumed through the web browser. You can create your own folder and report file structure in SQL Server Reporting Services and limit access to only the folders you want your users to see.

Creating favorite and linked reports

Service Manager allows you to save the selections you make in the **Parameters** section of the report window for future use. You can save settings for just yourself or create a linked report that will also be available to other users. Perform the following steps:

1. Open the Service Manager console, navigate to the **Reporting** node, and open the report of your choice.
2. Make the desired selections in the **Parameters** section of the report window.
3. On the right-hand side tasks pane, click on **Save as Favorite**, enter a name for the report, and then click on **OK**.
4. The report will now appear under the **Favorite Reports** folder in the **Reporting** section of the Service Manager console.

A linked report is a favorite report that can be accessed by other users. You can create linked reports only from reports that were imported through management packs. Follow these instructions to create a linked report:

1. Open the report of your choice in the Service Manager console.
2. Make the desired selections in the **Parameters** section of the report window.
3. On the right-hand side **Tasks** pane, click on **Save as Linked Report**.
4. Enter a name for the report and choose the management pack where you would like to save the linked report definition and optionally add a description.
5. Under **Select Folder**, choose the folder where you would like your report to show up. Click on **OK**. You may need to close and reopen the console for the link report to appear in your selected folder.
6. All users with access to Service Manager reports and the destination folder can now access the linked report by navigating to the respective location in the **Reporting** section of the Service Manager console.



Note that you require **Publisher** or **Content Manager** permissions in SQL Server Reporting Services to be able to create linked reports.

Understanding the Service Manager Data Warehouse data mart

The purpose of this recipe is to describe the database schema of the Service Manager Data Warehouse data mart. Knowing the anatomy and design of the data mart is key to successfully creating custom reports for System Center Service Manager.

Getting ready

You need to have a fully deployed Service Manager 2012 SP1/R2 environment that includes a registered Data Warehouse Management server with its prerequisite databases.

You will also need SQL Server connection rights to the Data Warehouse database server that allows you to view and read objects using **Microsoft SQL Server Management Studio (MSSMS)**.

A basic knowledge of SQL query syntax is recommended.

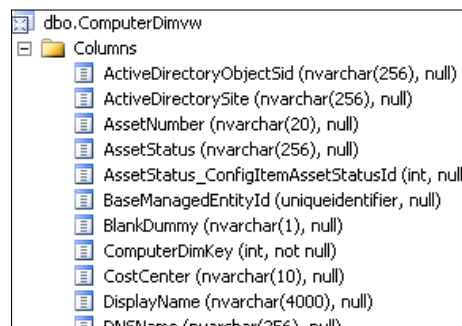
How to do it...

All data relevant for reporting is exposed in views inside the `DWDDataMart` database. You should not access the tables directly in your report queries. `DWDDataMart` is a data warehouse that uses three types of entities: dimensions, facts, and outriggers.

Understanding dimensions

A dimension in the database is roughly analogous to a class in Service Manager. Each class in Service Manager has a list of properties, while each dimension contains a list of attributes. Each dimension attribute will map to one property in a class.

Dimensions are exposed in views that normally end with the `...Dimvw` string, such as `ComputerDimvw`, as shown in the following screenshot:



To get a list of all computers, you can execute the following query:

```
SELECT * FROM ComputerDimvw
```

Note that data is never deleted from the data warehouse. Instead, deleted rows are marked accordingly. Each dimension has an attribute named `IsDeleted`. To get a list of all undeleted computers, you can execute the following query:

```
SELECT * FROM ComputerDimvw WHERE IsDeleted = 0
```

Furthermore, each dimension includes an attribute that represents the primary key for each row in the dimension. This attribute is named `[DimensionName]DimKey`. The primary key attribute for the `Computer` dimension is named `ComputerDimKey`:

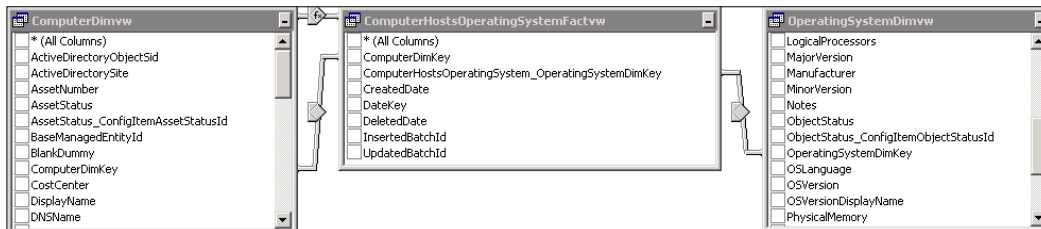
```
SELECT ComputerDimKey, * FROM ComputerDimvw WHERE IsDeleted = 0
```

Understanding facts

A fact in the data warehouse is analogous to a relationship in Service Manager. Each relationship allows for a source instance and a target instance (both are typically represented by dimensions) to be joined together.

An example of a fact is the relationship between a computer and the operating system running on the computer. This relationship is exposed in a view named `ComputerHostsOperatingSystemFactvw`.

To create a query that returns data from both the computer and its operating system, you need to join the two dimensions together using the view that represents the relationship fact, as shown in the following screenshot:



The join between the source dimension and the fact needs to be established using the dimension key attribute. To establish the join between the fact and the target dimension, use the corresponding target dimension key attribute. Note that you can exclude the deleted relationships by filtering the `DeletedDate` attribute by `IS NULL`:

```
SELECT
    co.PrincipalName,
    os.PhysicalMemory
```

```

FROM
  ComputerDimvw co
  LEFT OUTER JOIN ComputerHostsOperatingSystemFactvw coHos ON
    co.ComputerDimKey = coHos.ComputerDimKey
    AND coHos.DeletedDate IS NULL
  LEFT OUTER JOIN OperatingSystemDimvw os ON
    coHos.ComputerHostsOperatingSystem_OperatingSystemDimKey =
os.OperatingSystemDimKey
WHERE
  co.IsDeleted = 0

```

Understanding outriggers

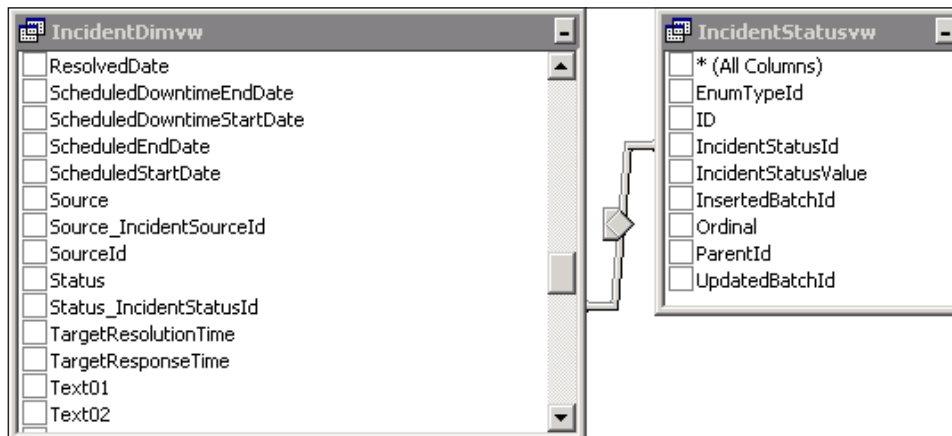
An outrigger is a list that can logically group together a set of values. Outriggers in the data warehouse can target one or more class properties and consolidate them into a single set of discrete values. These class properties are typically enumerations (aka **lists** in Service Manager).

An example of an outrigger is the `Status` property of the `Incident` class. This property is represented as a list in Service Manager.

If you execute the following query, you will notice that the values in the `Status` attribute contain internal names only:

```
SELECT Id, Status FROM IncidentDimvw
```

In order to retrieve the actual string values for the status, you need to join the corresponding view that represents the outrigger, which is `IncidentStatusvw` in this case. The join needs to be established between the `[DimensionAttributeName]_ [OuttriggerName] Id` attribute of the dimension and the `[OuttriggerName] Id` attribute of the outrigger, as shown in the following screenshot:



The string value for the outrigger is represented by the [OutriggerName] Value attribute, which is IncidentStatusValue in this case:

```
SELECT
    i.Id,
    i.Status,
    s.IncidentStatusValue
FROM
    IncidentDimvw i
    LEFT OUTER JOIN IncidentStatusvw s ON
        i.Status_IncidentStatusId = s.IncidentStatusId
WHERE
    i.IsDeleted = 0
```

How it works...

The central repository that stores all data available for Service Manager reporting is the `DWDataMart` database. This database is located in the SQL Server that hosts the Service Manager Data Warehouse databases and is also referred to as the Service Manager Data Mart.

System Center Service Manager makes the data from the operational **CMDB** database available in the `DWDataMart` database. The schema of the data mart is optimized for reporting.

The translation of entities in the CMDB to their respective representation in the data mart is defined in management packs. These management packs are synchronized from the operational database to the data mart as part of the `MPSyncJob` data warehouse job.

Each entity type in the CMDB has its own representation in the data mart. Classes are represented as dimensions, relationships are represented as facts, and enumerations (aka lists) are represented as outriggers. Service Manager creates views in the SQL database that you can use to write queries for your reports.

SQL Server Report Builder and other tools used to author reports offer visual wizards to create the queries you need to fuel your System Center reports. These tools, however, do not remove the requirement for you to have at least a basic knowledge of the **Structured Query Language (SQL)** used to query relational databases. A basic knowledge of SQL is essential to create and edit Service Manager reports.

The Service Manager Data Warehouse data mart is documented and is part of the SM Job Aids, which can be downloaded from the Microsoft website at <http://www.microsoft.com/en-us/download/details.aspx?id=27850>.

There's more...

There is a special condition when dealing with the Service Manager `DWDataMart` database.

Creating custom database objects for reporting

Although the `DWDataMart` database ships with System Center Service Manager, its intended purpose is to act as the source for reporting; hence, you are allowed to create database objects such as views, stored procedures, and user-defined functions to accommodate your reporting needs.

Whenever possible, use stored procedures for your reports. Stored procedures have advantages over views when it comes to optimizing the query and the ability to do complex queries that wouldn't be possible in views.

As a recommended practice, try to come up with a naming standard for all custom objects that you create in the `DWDataMart` database. This allows you to distinguish them from the out-of-the-box objects.

A sample naming standard would be `[ObjectTypeAbbreviation]_[Company]_r_[Name]`.

Some examples would be as follows:

- ▶ `usp_MyCompany_r_IncidentSLA` for a user-defined stored procedure
- ▶ `udf_MyComany_r_GetStatusDisplayName` for a user-defined function
- ▶ `uv_MyCompany_r_ListOfComputers` for a user-defined view

It is recommended that you do not use white spaces in the name of database objects.

Database permissions required for reporting

During the installation of the Service Manager Data Warehouse management server, you have to specify a reporting service account. The setup process automatically added this account as a SQL Server Login and added it as a member of the **reportuser** database role in the `DWDataMart` database.

In order for your custom-created reports to render successfully, you will need to grant explicit permissions to the `reportuser` database role for the objects you created in the `DWDataMart` database, such as views, stored procedures, or user-defined functions.

Creating Work Item and Configuration Item reports

This recipe will walk you through the steps for creating new reports for Work Items and Configuration Items stored in Service Manager.

Getting ready

We will use Report Builder 3.0 to create the reports in this recipe. You need to be familiar with using Report Builder to manage reports. Using Report Builder is explained in *Chapter 3, Unpacking System Center Report Building Tools*.

Work Item and Configuration Item data is stored in the Service Manager Data Warehouse data mart database. It is recommended that you read the *Understanding the Service Manager Data Warehouse data mart* recipe earlier in this chapter before following this recipe.

It is recommended that you store all your reports in a custom folder structure in SQL Server Reporting Services. Follow these steps to create a custom folder in the SQL Server Reporting Services used by Service Manager:

1. Open your web browser and navigate to `http://[SCSMDWSQL]/Reports`. Replace `[SCSMDWSQL]` with the fully qualified domain name of the SQL server that is used for reporting. If SQL Server Reporting Service is running as a named instance, the syntax of the URL will be `http://[SCSMDWSQL]/Reports_[InstanceName]`.
2. Click on the **SystemCenter** folder and then on the **ServiceManager** folder.
3. Click on **New Folder**, enter a name for the folder, such as `Custom Reports`, and optionally enter a description. Then, click on **OK**.



Note that the permissions applied to the newly created folder will be inherited from the parent **ServiceManager** folder. You can break inheritance and apply custom permissions to your folders if required.

How to do it...

In this recipe, we will create a report that shows a chart of the number of incidents created over time. We will also create a report that lists the computers that we have in our Service Manager CMDB.

Creating a Work Item report

Perform the following steps:

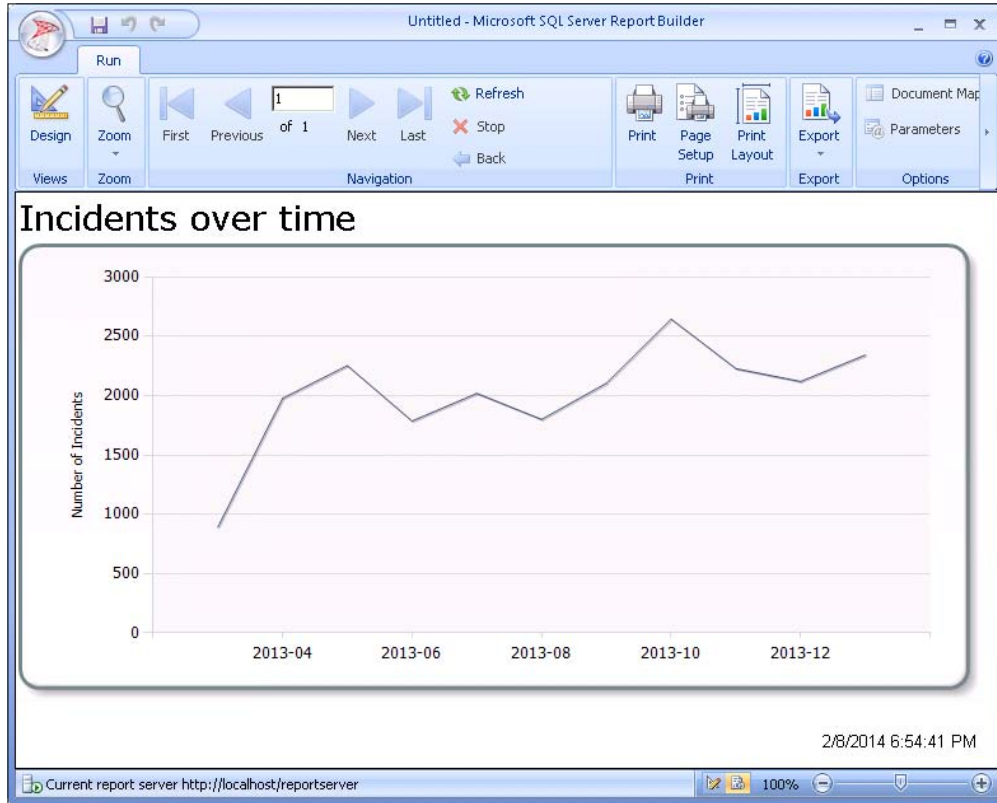
1. Start Report Builder.
2. Under **New Report**, select **Chart Wizard**.
3. Select **Create a dataset** and click on **Next**.
4. Select the `DWDataMart` data source connection. If it is not in the list, click on **Browse...** to browse to the data source in SQL Server Reporting Services. Click on **Next**.
5. If prompted for data source credentials, select **Use the current Windows user** and then click on **OK**.
6. Click on **Edit as Text** and type the following SQL query in the query window:


```
SELECT
    CONVERT(varchar(7), CreatedDate, 126) AS MonthCreated,
    COUNT(*) AS IncidentCount
FROM
    IncidentDimvw
WHERE
    IsDeleted = 0
GROUP BY
    CONVERT(varchar(7), CreatedDate, 126)
ORDER BY
    1
```

7. Click on **Next**, select the **Line** chart type, and then click on **Next**.
8. Drag the **MonthCreated** field and drop it in the **Categories** area.
9. Drag the **IncidentCount** field and drop it in the **Values** area.
10. Click on **Next**, choose the desired chart style, and then click on **Finish**.

The wizard will now close, and Report Builder displays the chart in your newly created report. You can now modify the report and the chart as desired. For example, add a **Title** named `Incidents over time` and rename the **Axis** titles to `Number of Incidents` and `Date Created`.

When you are done customizing your report, click on **Run**, as shown in the following screenshot:



 Go to **View | Properties** to show the report properties in the Report Builder application. It is simpler to make changes in the **Properties** area.

Go back to the **Design** mode and click on **Save**. Navigate to the custom folder created earlier in this recipe, enter `Incidents over time` as the name, and click on **Save**. Your report is now published to SQL Server Reporting Services and made available to users using the Service Manager console or a web browser.

Creating a Configuration Item report

To create a Configuration Item report perform the following steps:

1. Start Report Builder.
2. Under **New Report**, select **Table or Matrix Wizard**.
3. Select **Create a dataset** and click on **Next**.
4. Select the `DWDataMart` data source connection. If it is not in the list, click on **Browse...** to browse to the data source in SQL Server Reporting Services. Then, click on **Next**.
5. If prompted for data source credentials, select **Use the current Windows user** and click on **OK**.
6. Click on **Edit as Text** and type the following SQL query in the query window:

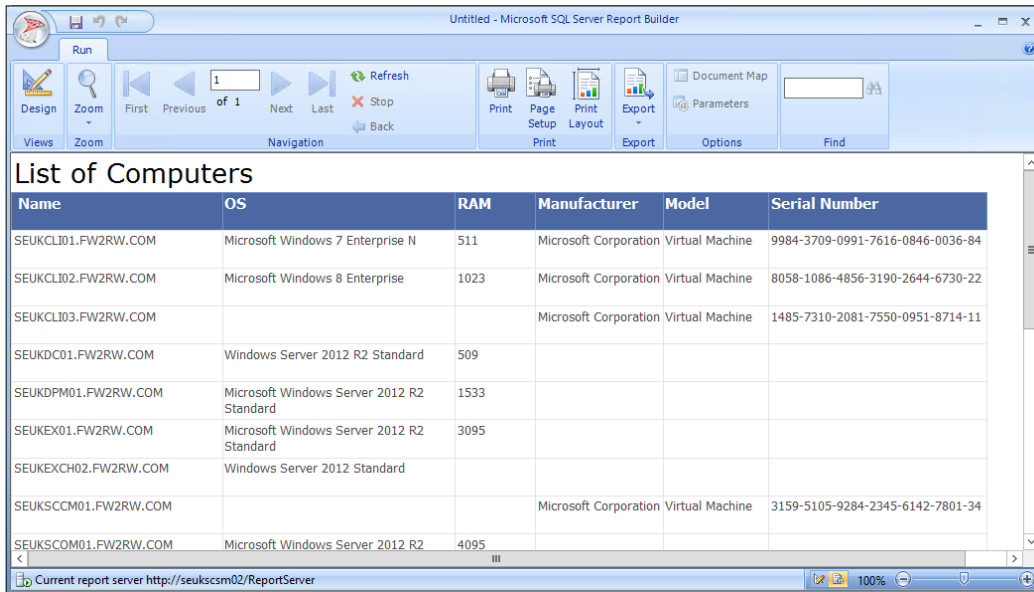
```

SELECT co.PrincipalName,
       os.OSVersionDisplayName,
       os.PhysicalMemory,
       dc.Manufacturer,
       dc.Model,
       dc.SerialNumber
FROM
  ComputerDimvw co
  LEFT OUTER JOIN ComputerHostsOperatingSystemFactvw coHos
ON
  co.ComputerDimKey = coHos.ComputerDimKey
  AND coHos.DeletedDate IS NULL
  LEFT OUTER JOIN OperatingSystemDimvw os ON
  coHos.ComputerHostsOperatingSystem_OperatingSystemDimKey =
os.OperatingSystemDimKey
  LEFT OUTER JOIN DeployedComputerRunsWindowsComputerFactvw
dcRco ON
  co.ComputerDimKey =
dcRco.DeployedComputerRunsWindowsComputer_ComputerDimKey
  AND dcRco.DeletedDate IS NULL
  LEFT OUTER JOIN DeployedComputerDimvw dc ON
  dcRco.DeployedComputerDimKey =
dc.DeployedComputerDimKey
WHERE
  co.IsDeleted = 0
ORDER BY
  1

```

7. Click on **Next**.
8. Drag all fields and drop them in the **Values** area.
9. Click on **Next** twice, choose the desired table style (for example, **Ocean**), and then click on **Finish**.

The wizard will now close, and Report Builder displays the table in your newly created report. You can now modify the report and the table as desired (for example, type `List of Computer` as the title and change the column headings). When you are done customizing your report, click on **Run**, as shown in the following screenshot:



Go back to the **Design** mode and click on **Save**. Navigate to the custom folder created earlier in this recipe, enter `List of Computer` as the name, and click on **Save**. Your report is now published to SQL Server Reporting Services and made available to users using the Service Manager console or a web browser.

How it works...

Service Manager leverages SQL Server Reporting Services to provide rich reporting functionality to end users. As an administrator, you can create your own reports that access the `DWDataMart` database provided by the Service Manager Data Warehouse.

The recipe steps showed two options you have to present the data in reports. The first option showed how to create a chart type report in Report Builder using the Incident Work Items class. The second option showed you how to create a Tablix-type report using the Configuration Items class. The fuel for both of these options is the SQL query you use to create the datasets. You can find additional resources on SQL queries in *Appendix, Useful Websites, Chapter Code, and Community Resources*.

There's more...

How about if you want to create your own custom reports for valuable data introduced by the new Service Level Management feature in Service Manager 2012?

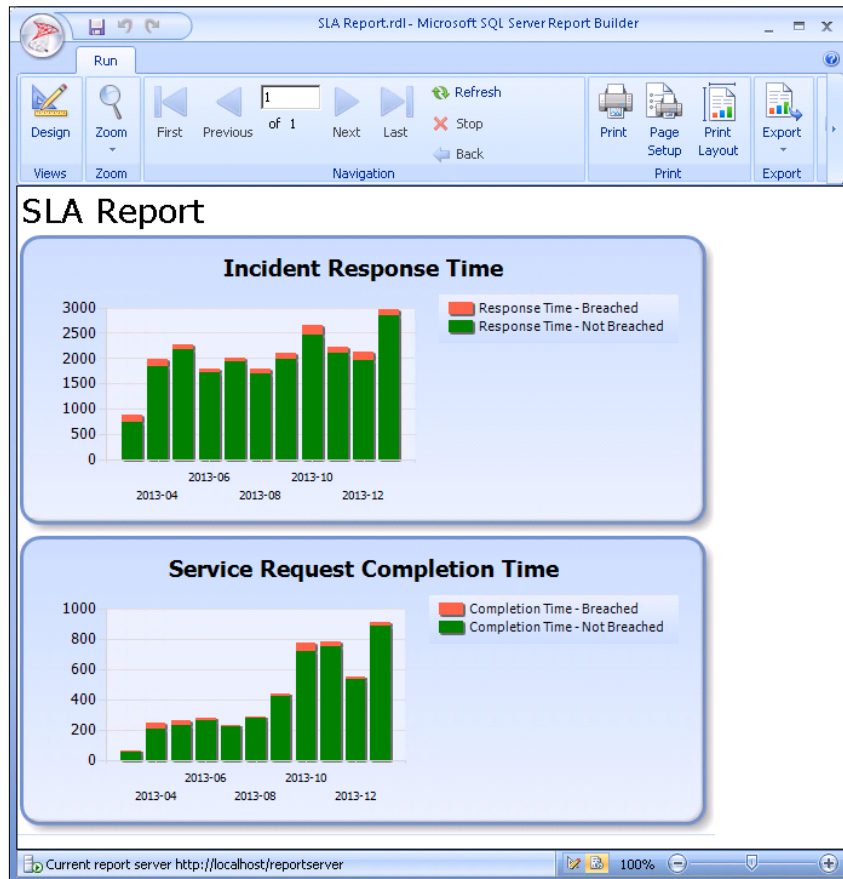
Creating Service Level reports

Service Manager 2012 introduced Service Level Management that allows you to monitor key metrics, such as Incident Response Time and Service Request Completion Time. Unfortunately, there are no out-of-the-box reports available to report on Service Level breaches.

The instructions earlier in this recipe should have given you a basic understanding of how to use tables and charts to display Service Manager Data in reports. You can use the following query as an example to create a custom Service Level report:

```
SELECT
    m.DisplayName AS Metric,
    CONVERT(varchar(7), StartDate, 126) AS StartDate,
    SUM(CONVERT(tinyint, s.IsBreached)) AS Breached,
    COUNT(*) - SUM(CONVERT(tinyint, s.IsBreached)) AS NotBreached,
    COUNT(*) AS Total
FROM
    SLAInstanceInformationFactvw s
    INNER JOIN SLAConfigurationDimvw c ON
        s.SLAConfigurationDimKey = c.SLAConfigurationDimKey
    INNER JOIN SLAConfigurationHasMetricFactvw cm ON
        c.SLAConfigurationDimKey = cm.SLAConfigurationDimKey
    INNER JOIN SLAMetricDimvw m ON
        cm.ConfigurationRefersToMetric_SLAMetricDimKey =
m.SLAMetricDimKey
WHERE
    s.IsCancelled = 0
    AND c.IsDeleted = 0
    AND m.IsDeleted = 0
GROUP BY
    m.DisplayName,
    CONVERT(varchar(7), StartDate, 126)
ORDER BY
    1, 2
```


You can use this query as a starting point to establish additional joins to your work items based on service level metrics. You could create a report showing the number of breached and compliant (not breached) work items over a period of time as shown in the screenshot.



See also

- ▶ Refer to the following recipes for more information on using Report Builder:
 - ❑ The *Installing Report Builder* recipe in *Chapter 3, Unpacking System Center Report Building Tools*
 - ❑ The *Understanding Report Builder basics* recipe in *Chapter 3, Unpacking System Center Report Building Tools*
 - ❑ The *Building basic reports with Report Builder* recipe in *Chapter 3, Unpacking System Center Report Building Tools*

Creating reports using OLAP cubes

In the *Using out-of-the-box Service Manager Reports and cubes* recipe, we showed you how to use Microsoft Excel to access the OLAP cubes that are shipped with Service Manager.

Under certain circumstances, you might want to use these OLAP cubes to create reports hosted in SQL Server Reporting Services and make them available to your end users through the console or web browser.

In this recipe, we will use Report Builder to create and publish a report using the data stored in the Service Manager OLAP cubes.

Getting ready

Before we can start using SQL Server Reporting Services with Service Manager OLAP cubes, we need to create a data source that points to the OLAP cubes. The cubes are hosted in `DWASDataBase` on the instance of SQL Server Analysis Services used by Service Manager. Perform the following steps:

1. Open your web browser and navigate to `http://[SCSMDWSQL]/Reports`. Replace `[SCSMDWSQL]` with the fully qualified domain name of the SQL server that is used for reporting. If SQL Server Reporting Service is running as a named instance, the syntax of the URL will be `http://[SCSMDWSQL]/Reports_[InstanceName]`.
2. Click on the **SystemCenter** folder, then click on the **ServiceManager** folder.
3. Click on **New Data Source** and complete the form as follows:
 - Name: `DWASDataBase`
 - Data source type: **Microsoft SQL Server Analysis Services**
 - Connection string: `Data Source=[SCSMDWAS];Catalog=DWASDataBase` (replace `[SCSMDWAS]` with the fully qualified domain name of the SQL Server Analysis Services server that hosts the Service Manager OLAP cubes)



Do not use the *Enter* key to move to the next line when typing the connection string. The text will automatically word wrap. Using the *Enter* key to move to the next line will generate a syntax error.

4. Select **Credentials stored securely in the report server**
5. Enter the username and password of an account that has read access to the cubes in the `DWASDataBase` Analysis Services database.
6. Select **Use as Windows credentials when connecting to the data source**

7. Click on **Test Connection** to validate the connection.
8. Click on **OK** to save the new data source, as shown in the following screenshot:

Name: DWASDataBase

Description: Service Manager OLAP Cubes data source

Hide in tile view

Enable this data source

Data source type: Microsoft SQL Server Analysis Services

Connection string: DataSource=SEUKSCSM02.FW2RW.COM; Catalog=DWASDataBase

Connect using:

Credentials supplied by the user running the report

Display the following text to prompt user for a user name and password:
Type or enter a user name and password to access the data source

Use as Windows credentials when connecting to the data source

Credentials stored securely in the report server

User name: FW2RW\SCSM_RPT

Password: [Masked]

Use as Windows credentials when connecting to the data source

Impersonate the authenticated user after a connection has been made to the data source

Windows integrated security

Credentials are not required

Test Connection

Connection created successfully.

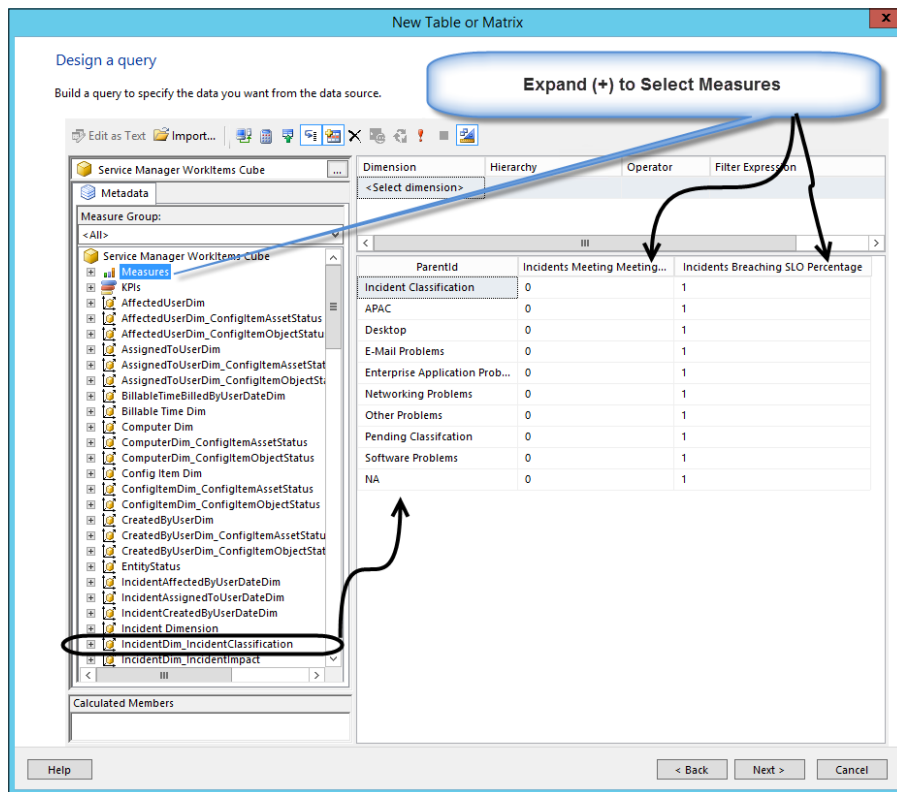
OK Cancel

How to do it...

In this example, we will create a report that shows the percentage of incidents meeting and breaching the SLA by incident classification category. Perform the following steps:

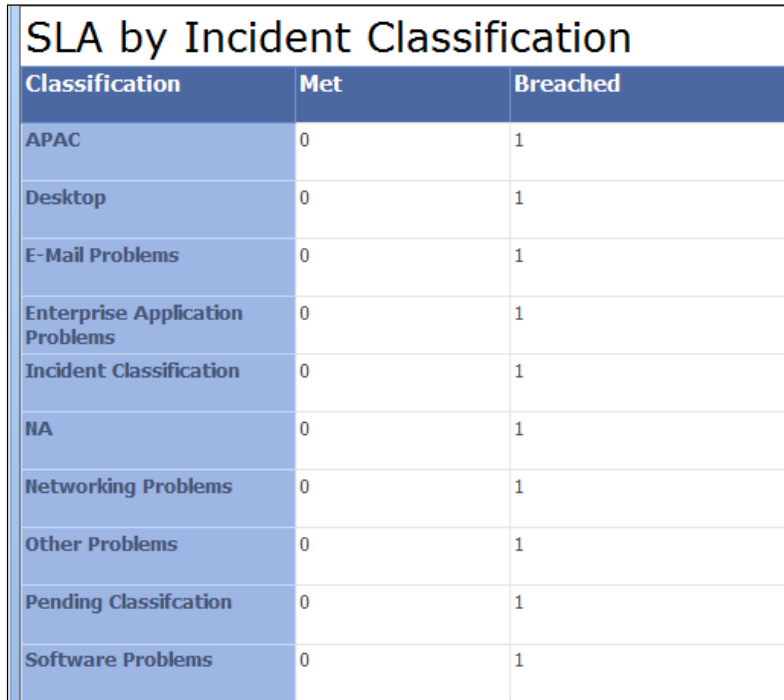
1. Start Report Builder.
2. Under **New Report**, select **Table or Matrix Wizard**.
3. Select **Create a dataset** and click on **Next**.
4. Click on **Browse...** to browse to the DWASDataBase data source in SQL Server Reporting Services. Click on **Next**.

5. If prompted for data source credentials, select **Use the current Windows user** and click on **OK**.
6. The **Design a query** wizard starts. In the upper-left corner, change to **Service Manager WorkItems Cube**.
7. Drag and drop the IncidentDim_IncidentClassification dimension in the **Drag levels or measures here to add to the query** middle pane area.
8. Under **Measure Group**, expand **Measures**, expand **IncidentDim**, and drag **Incidents Meeting Meeting All SLOs Percentage** and drop it in the middle pane area next to **IncidentDim_IncidentClassification**.
9. Drag **Incidents Breaching SLO Percentage** and drop it in the middle pane area next to **Incidents Meeting Meeting All SLOs Percentage**, as shown in the following screenshot:



10. Click on **Next**, drag and drop the **ParentId** field into the **Row groups** area, drag and drop the other two measure fields into the **Values** area, and then click on **Next**.
11. Choose the desired table style and then click on **Finish**.

The wizard will now close, and Report Builder displays the table in your newly created report. You can now modify the report and the table as desired. When you are done customizing your report, click on **Run**, as shown in the following screenshot:



Classification	Met	Breached
APAC	0	1
Desktop	0	1
E-Mail Problems	0	1
Enterprise Application Problems	0	1
Incident Classification	0	1
NA	0	1
Networking Problems	0	1
Other Problems	0	1
Pending Classification	0	1
Software Problems	0	1

Go back to the **Design** mode and click on **Save**. Navigate to a custom folder in SQL Server Reporting Services, enter *SLA by Incident Classification* as the name, and then click on **Save**. Your report is now published to SQL Server Reporting Services and made available to users using the Service Manager console or a web browser.

How it works...

SQL Server Analysis Services ships with a data source provider that allows you to access the data stored in OLAP databases and cubes using reporting tools such as Report Builder. Before you can use Report Builder to access the cubes provided by Service Manager, you have to manually configure a data source in SQL Server Reporting Services.

As cubes store data in a multidimensional schema, SQL Analysis Services implements the **MDX (Multidimensional Expression)** query language. MDX queries are used in your reports to define the datasets. The MDX used in this example is as follows:

```
SELECT
    NON EMPTY
    {
        [Measures].[Incidents Breaching SLO Percentage],
        [Measures].[Incidents Meeting Meeting All SLOs Percentage]
    }

    ON COLUMNS,
    NON EMPTY
    {
        (DESCENDANTS ([IncidentDim_IncidentClassification].[ParentId].
[Level 02].ALLMEMBERS))
    }
    ON ROWS
FROM
    [SystemCenterWorkItemsCube]
```

Creating reports using the operational database

Service Manager transfers the data to the data mart using an ETL process as described in the introduction of this chapter. The individual steps of the ETL process are implemented as data warehouse jobs that are run by the Service Manager Data Warehouse management server.

The entire ETL process cycle to transfer new data or data changes from the operational database to the data mart can take up to 1 hour to complete.

While this is okay for most common reporting needs, it can be an issue when you would like to get real-time information from Service Manager, such as the number of currently open and unassigned incidents.

In this recipe, we will access the Service Manager operational database to retrieve real-time information from the CMDB.



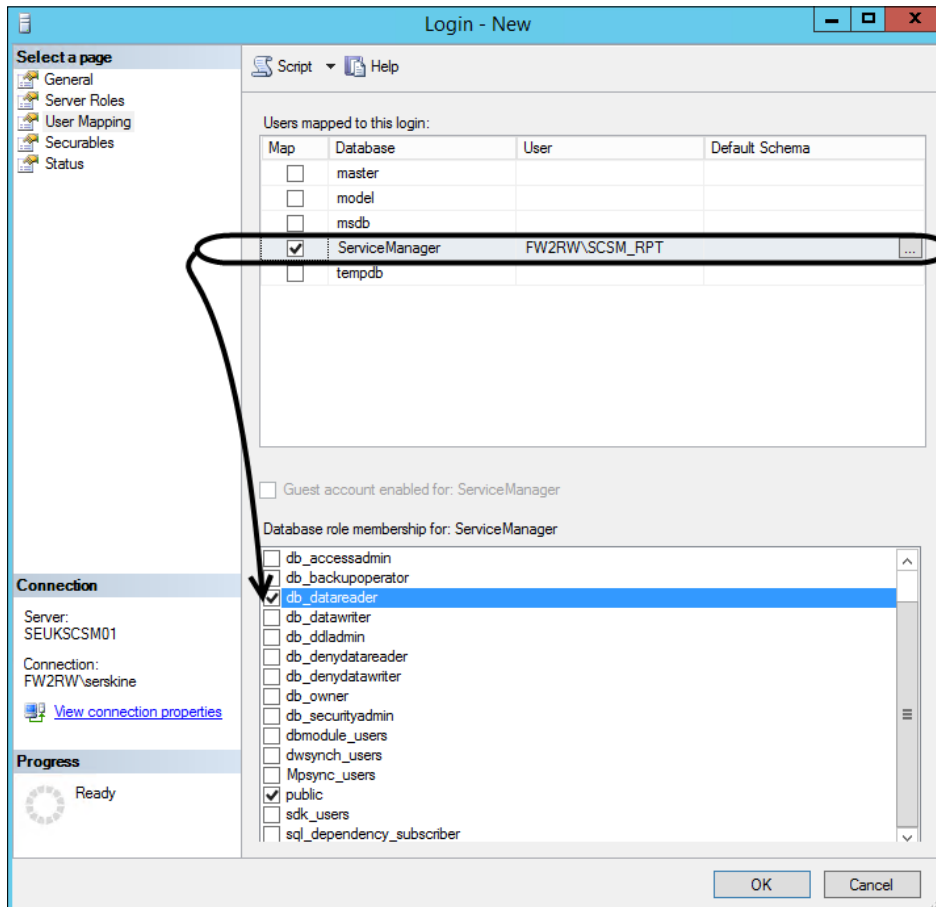
Accessing the Service Manager operational database

Note that accessing the operational database is not supported by Microsoft. It is possible that the database schema might change between release cycles. Also, in order to prevent locks from occurring when accessing the database, it is recommended that you always use the `NOLOCK` query option.

Getting ready

Before we can access the operational database, we need to grant read permissions to a user account for reporting access to the data. We will use the SCSM Reporting service account for this purpose. Perform the following steps:

1. Start SQL Server Management Studio and connect to the SQL server that hosts the `ServiceManager` database.
2. Expand **Security** and **Logins**, and check whether the SCSM Reporting Account is present in the list of logins. If not, add the login.
3. Double-click on the SCSM Reporting Account login. Then, under **User Mapping**, select the **ServiceManager** database and assign the `db_datareader` role (you can leave the `public` role enabled).
4. Click on **OK**, as shown in the following screenshot:



Next, we have to create a data source for the *ServiceManager* database in SQL Server Reporting Services. Perform the following steps:

1. Open your web browser and navigate to `http://[SCSMDWSQL]/Reports`. Replace `[SCSMDWSQL]` with the fully qualified domain name of the SQL server that is used for reporting. If SQL Server Reporting Service is running as a named instance, the syntax of the URL will be `http://[SCSMDWSQL]/Reports_[InstanceName]`.
2. Click on the **SystemCenter** folder and then on the **ServiceManager** folder.
3. Click on **New Data Source** and complete the form as follows:
 - ❑ Name: **ServiceManager**
 - ❑ Data source type: **Microsoft SQL Server**
 - ❑ Connection string: `data source=[SCSMSQL];initial catalog=ServiceManager` (replace `[SCSMSQL]` with the fully qualified domain name of the SQL Server hosts the *ServiceManager* database)
4. Select **Credentials stored securely in the report server**
5. Enter the username and password of the SCSM Reporting Account
6. Select **Use as Windows credentials when connecting to the data source**
7. Click on **Test Connection** to validate the connection.
8. Click on **OK** to save the new data source, as shown in the following screenshot:

The screenshot displays the 'New Data Source' configuration window. Key elements include:

- Name:** ServiceManager
- Description:** Service Manager Operation Data source
- Enable this data source:**
- Data source type:** Microsoft SQL Server
- Connection string:** Data Source=SEUKSCSM01.FW2RW.COM; Initial Catalog=ServiceManager
- Connect using:**
 - Credentials supplied by the user running the report
 - Credentials stored securely in the report server
 - User name: FW2RWSCSM_RPT
 - Password: [masked]
 - Use as Windows credentials when connecting to the data source
 - Impersonate the authenticated user after a connection has been made to the data source
 - Windows integrated security
 - Credentials are not required
- Test Connection:** [button]
- Message:** Connection created successfully.
- Buttons:** OK, Cancel

How to do it...

We will now create a dashboard-like report that displays the number of incidents created and resolved today, this week, and this month. Also, we want to display the number of open and unassigned incidents. Perform the following steps:

1. Start Report Builder.
2. Under **New Report**, select **Blank Report**.
3. Under **Data Sources**, add the newly created `ServiceManager` data source.
4. Under **Datasets**, add a dataset named `DSCreatedResolved` (this is shown in the next screenshot). Select the **Use a dataset embedded in my report** option, select the `ServiceManager` data source, select **Text** query type, type the following query and click on **OK**:

```

DECLARE @DateTime datetime = GETUTCDATE()
DECLARE @StartOfToday datetime = DATEADD(DAY, DATEDIFF(DAY,
'19000101', @DateTime), '19000101')
DECLARE @EndOfToday datetime = DATEADD(DAY, DATEDIFF(DAY,
'19000101', @DateTime) + 1, '19000101')
DECLARE @StartOfWeek datetime = DATEADD(WEEK,
DATEDIFF(WEEK, '19000101', @DateTime), '19000101') - 1
DECLARE @EndOfWeek datetime = DATEADD(WEEK, DATEDIFF(WEEK,
'19000101', @DateTime) + 1, '19000101') - 1
DECLARE @StartOfMonth datetime = DATEADD(MONTH,
DATEDIFF(MONTH, '19000101', @DateTime), '19000101')
DECLARE @EndOfMonth datetime = DATEADD(MONTH,
DATEDIFF(MONTH, '19000101', @DateTime) + 1, '19000101')

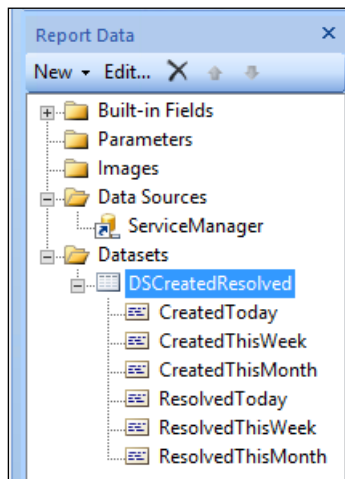
SELECT
    ISNULL(SUM(CASE WHEN
CreatedDate_6258638D_B885_AB3C_E316_D00782B8F688 BETWEEN
@StartOfToday AND @EndOfToday THEN 1 END),0) AS
CreatedToday,
    ISNULL(SUM(CASE WHEN
CreatedDate_6258638D_B885_AB3C_E316_D00782B8F688 BETWEEN
@StartOfWeek AND @EndOfWeek THEN 1 END),0) AS
CreatedThisWeek,
    ISNULL(SUM(CASE WHEN
CreatedDate_6258638D_B885_AB3C_E316_D00782B8F688 BETWEEN
@StartOfMonth AND @EndOfMonth THEN 1 END),0) AS
CreatedThisMonth,
    ISNULL(SUM(CASE WHEN
ResolvedDate_D2A4C73F_01B8_29C5_895B_5BE4C3DFAC4E BETWEEN
@StartOfToday AND @EndOfToday THEN 1 END),0) AS
ResolvedToday,

```

```

ISNULL (SUM (CASE WHEN
ResolvedDate_D2A4C73F_01B8_29C5_895B_5BE4C3DFAC4E BETWEEN
@StartOfWeek AND @EndOfWeek THEN 1 END), 0) AS
ResolvedThisWeek,
ISNULL (SUM (CASE WHEN
ResolvedDate_D2A4C73F_01B8_29C5_895B_5BE4C3DFAC4E BETWEEN
@StartOfMonth AND @EndOfMonth THEN 1 END), 0) AS
ResolvedThisMonth
FROM
MT_System$WorkItem$Incident WITH (NOLOCK)

```



- Under **Datasets**, add a dataset named `DSCreatedResolved` (this is shown in the next screenshot). Select the **Use a dataset embedded in my report** option, select the `ServiceManager` data source, select **Text** query type, and type the following query. Then, click on **OK**:

```

SELECT
    'Open Incidents' AS [Title],
    COUNT(*) AS [Count]
FROM
    MT_System$WorkItem$Incident WITH (NOLOCK)
WHERE
    Status_785407A9_729D_3A74_A383_575DB0CD50ED NOT IN
    ('2B8830B6-59F0-F574-9C2A-F4B4682F1681', 'BD0AE7C4-
3315-2EB3-7933-82DFC482DBAF')

UNION

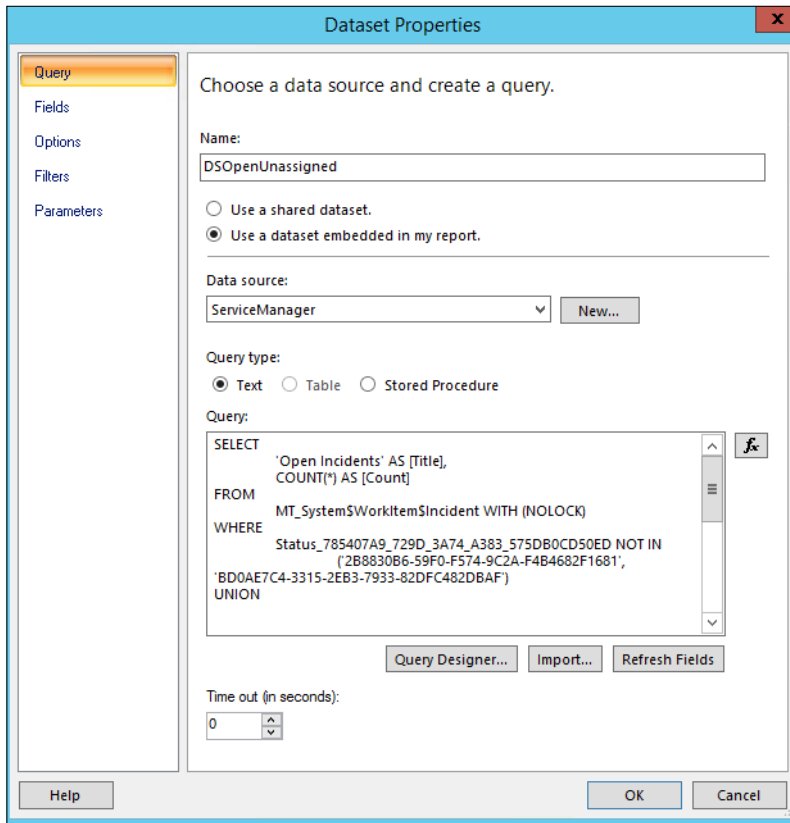
SELECT

```

```

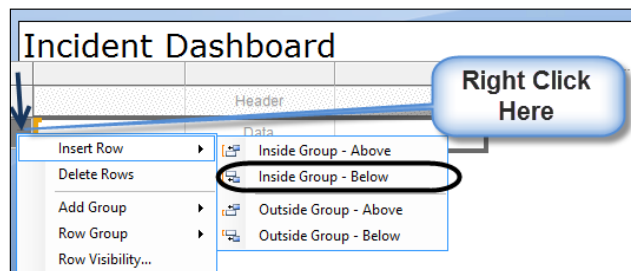
'Unassigned Incidents' AS [Title],
COUNT(*) AS [Count]
FROM
    MT_System$WorkItem$Incident WITH (NOLOCK)
WHERE
    Status_785407A9_729D_3A74_A383_575DB0CD50ED NOT IN
    ('2B8830B6-59F0-F574-9C2A-F4B4682F1681', 'BD0AE7C4-
3315-2EB3-7933-82DFC482DBAF')
    AND NOT EXISTS
    (SELECT * FROM Relationship WITH (NOLOCK)
    WHERE
        SourceEntityId = BaseManagedEntityId
        AND RelationshipTypeId = '15E577A3-6BF9-6713-4EAC-
BA5A5B7C4722'
        AND IsDeleted = 0)

```

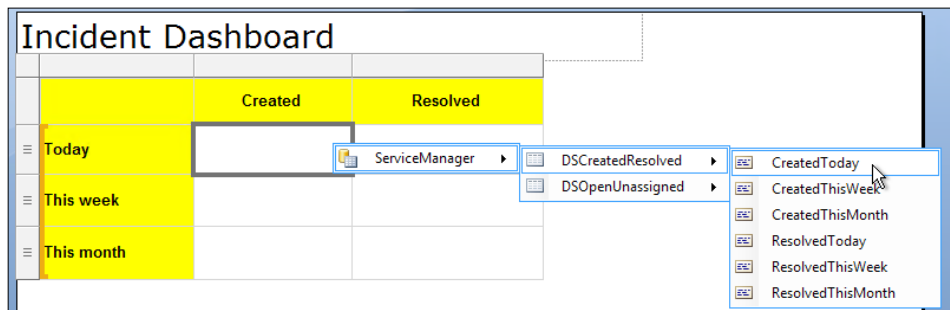


Now, it's time to design our report. Place two tables in the report to hold the data in the two datasets: `DSCreatedResolved` and `DSOpenUnassigned`. Follow these steps to complete the dashboard design:

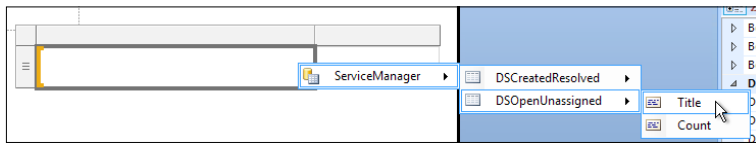
1. Double-click on and replace the text `Click to add title` with `Incident Dashboard`.
2. Under the **Insert** tab, click on **Table** and then on **Insert Table**.
3. Drag the icon and draw a table on the left-hand side of the report.
4. Right-click on the **Data** row in the table (click on the left-hand side gray outer part) and then go to **Insert Row | Inside Group - Below**.
5. Right-click on the last row in the table (click on the left-hand gray outer part) and then go to **Insert Row | Inside Group - Below**, as shown in the following screenshot:



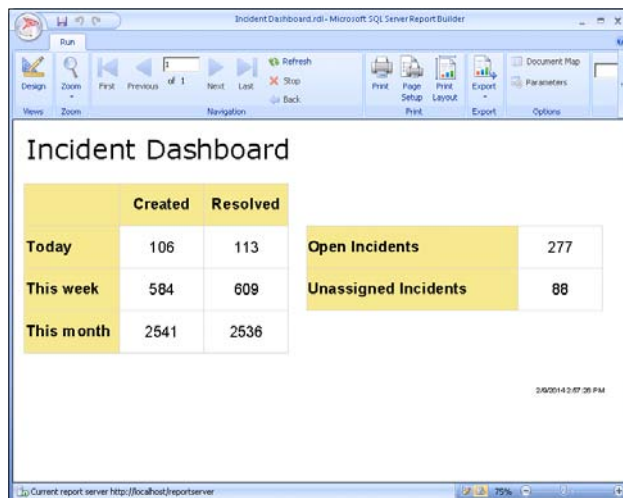
6. Type `Created` and `Resolved` in the second and third column headings of the table as shown in the screenshot (leave the first cell blank).
7. Skip the first row cell and type `Today`, `This week`, and `This month` in the second, third and fourth row cells, as illustrated in the screenshot.
8. Change the font style, size, and background color of the cells that hold the column and row headings according to the screenshot.
9. Hover the mouse over the cell that intersects **Created** and **Today**. Click on the **Table** icon. Go to **ServiceManager | DSCreatedResolved | CreatedToday**, as shown in the following screenshot:



10. Repeat step 9 for the other five data cells (under **Created** and **Resolved**).
11. Apply style and formatting to the six data cells (for example, change the font size and format as bold).
12. Under **Insert** tab, click on **Table** and then on **Insert Table**.
13. Drag the **Draw Table** icon and draw a table on the right-hand side of the first table.
14. Select the last column in the table, right-click on it, and click on **Delete Columns**.
15. Select the **Header** row (first row) in the table, right-click on it, and click on **Delete Rows**.
16. Hover the mouse over the first cell (left-hand side cell), click on the **Table** icon, and go to **ServiceManager | DSOpenUnassigned | Title**, as shown in the following screenshot:



17. Hover the mouse over the right-hand-side cell, click on the **Table** icon, and select **Count**.
18. Apply style and formatting to the cells to suit your needs (for example, make the cell bold and change the font size to 12 pts).
19. Click on **Run** to preview the report.
20. Save the report with the name of Incident Dashboard.rdl, as shown in the following screenshot:



How it works...

You can use Report Server to access any accessible data source using the supported data source providers. Before you can use the Service Manager operational database, you need to configure permissions in SQL Server and then create a data source in SQL Server Reporting Services pointing to the `ServiceManager` database.

There's more...

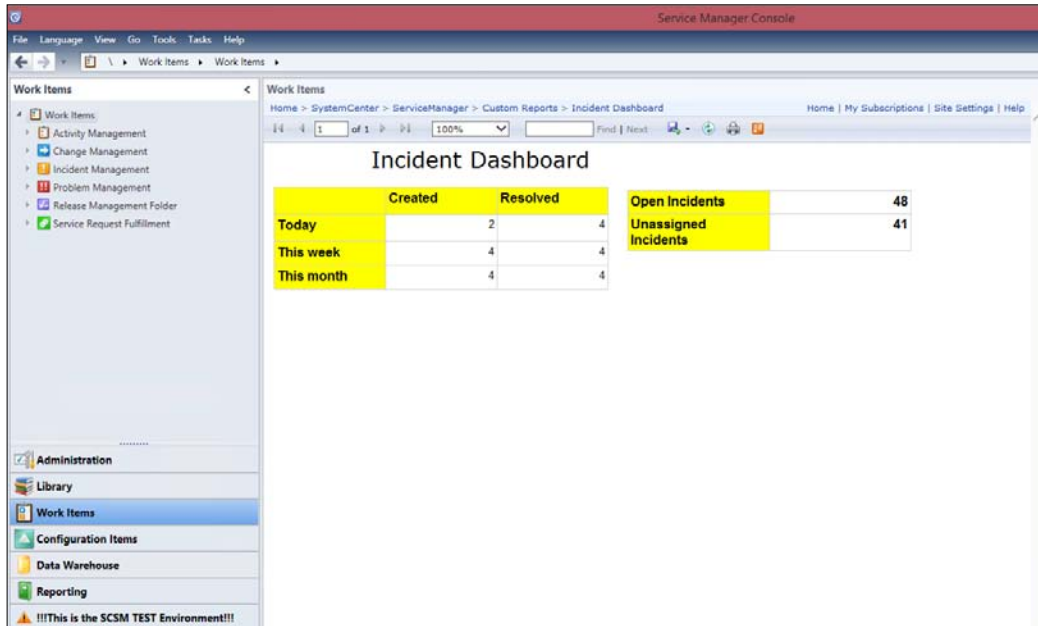
The report we just created is showing realtime data which you want to access frequently. You have the option to present this report in the Work Items section instead of the Reporting section of the Service Manager console.

Integrating the dashboard in the Service Manager console

Service Manager allows you to display any web page inside the Service Manager console. As reports hosted in SQL Server Reporting Services can be accessed via a web browser, we can display the report in the Service Manager console in the Work Items, *wunderbar!* To do so, make sure that the report was saved with the name `Incident Dashboard` to SQL Server Reporting Services in a custom folder. Create an XML file named `Custom.IncidentDashboard.View.xml` and type the code segment from the `Custom.IncidentDashboard.View.xml` listed in the code section of *Appendix, Useful Websites, Chapter Code, and Community Resources*, into the XML file using an XML text editor (for example, **NotePad++**).

Make sure that you change the `Source` attribute of the `<WebBrowser>` tag to point to the correct URL. Replace `[SSRSServerName]` with the fully qualified domain name of the server hosting SQL Server Reporting Services and also check the path to the report and adjust it if required.

Save the XML file and import the management pack into Service Manager. Restart the Service Manager console, navigate to **Work Items**, and click on the **Work Items** root element, as shown in the following screenshot:



Accessing data using Microsoft Excel

While Service Manager comes with OLAP cubes that allow you to gain a deep insight into the data stored in Service Manager, navigating the cubes and creating reports in Microsoft Excel can be a bit complicated, especially if the database schema and objects are not known to the report author.

An alternative approach is to use Microsoft Excel to access the `DWDDataMart` database, instead of the OLAP cubes. Using this method, you can import raw data from the Service Manager Data mart into Microsoft Excel and then use common features, such as PivotTables and PivotCharts, to create ad hoc reports.

Getting ready

We will use a view in the `DWDDataMart` database to provide an abstraction layer from the database model. The view will hold friendly names for the columns and will be limited to the data of interest to your report consumers.

In this example, we are providing a list of all the incidents with all the commonly used attributes. To create the view, open SQL Server Management Studio and connect to the SQL Server that hosts the `DWDataMart` database. Create a new view (select **New Query**, type the code, and click on **Execute**) using the query in the **Accessing data using Microsoft Excel Query Code** section of *Appendix, Useful Websites, Chapter Code, and Community Resources*.

Next, we need to grant permissions for the `reportuser` database role to the newly created view. This can be done by executing the following query:

```
USE DWDataMart
GO
GRANT SELECT ON v_Custom_r_AllIncidents TO reportuser
GO
```



You need to make sure that the users who access `DWDataMart` from Microsoft Excel are members of the `reportuser` database role.

It is recommended that you create an Active Directory group and add this group as a login to SQL Server. Add all the users that you want to grant access to the data to this Active Directory group. Then, add the login as a user to the `DWDataMart` database and assign it to the `reportuser` database role.

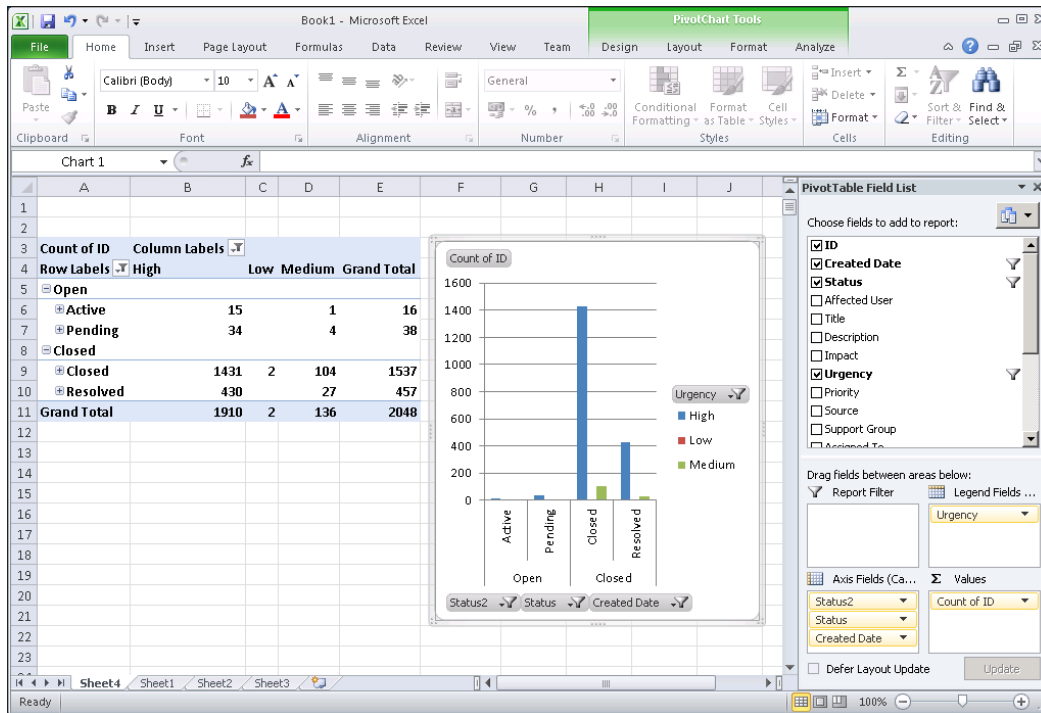
How to do it...

Perform the following steps:

1. Start Microsoft Excel.
2. On the **Data** ribbon, in the **Get External Data** area, click on **From Other Sources** and then choose **From SQL Server**.
3. In the **Server name** field, enter the name of the SQL Server that hosts the `DWDataMart` database, select **Use Windows Authentication**, and then click on **Next**.
4. Select the `DWDataMart` database, click on the `v_Custom_r_AllIncidents` view, and then click on **Finish**.
5. In the **Import Data** dialog, click on **OK**.

Microsoft Excel connects to the data source and retrieves the data from the view created earlier in this recipe. You can now use common Excel features, such as PivotTables or PivotCharts, to analyze your data.

When you want Excel to load the latest data from the database, right-click anywhere inside the data table and click on **Refresh**, as shown in the following screenshot:



How it works...

Microsoft Excel allows you to access external data from various data sources. Following the instructions in this chapter, you have learned how you can leverage Microsoft Excel to analyze the data stored in Microsoft SQL Server.

The process of using Excel to analyze the data stored in SQL Server involves creating database objects, such as views, to present the data in a reusable format as well as configuring database user roles and permissions.

Extending the Service Manager Data Warehouse

In the *Creating Work Item and Configuration Item reports* recipe earlier in this chapter you learned how you can create custom reports based on the data stored in the Service Manager data mart.

One of the key components of Service Manager is its extensible class model. Using authoring tools such as the **System Center Service Manager Authoring Tool**, you can create management packs that add new classes to the CMDB of Service Manager, extend the existing classes, or you can create new relationships between classes.

When you create custom classes and relationships for Service Manager, these are not automatically transferred to the data warehouse. This recipe walks you through the steps needed to transfer custom classes, relationships, and enumerations (aka lists) to the Service Manager Data Warehouse.



Class extensions

Class extensions are automatically transferred to the data warehouse, provided that they are defined in a sealed management pack.

Getting ready

Before we can start extending the data warehouse, we need to have some custom classes, relationships, and enumerations that we want to transfer to the data warehouse. You can define new entity types using authoring tools such as the System Center Service Manager Authoring Tool. Service Manager authoring is out of the scope of this book. For more information on how to author for Service Manager, we recommend *System Center 2012 Service Manager Cookbook*, which can be found at <http://www.packtpub.com/microsoft-system-center-service-manager-2012-cookbook/book>.

In this example, we have defined a custom configuration item class for mobile phones. This class has a `Manufacturer` property, which is an enumeration. Furthermore, the mobile phone class relates to the user class through the `MobilePhoneOwnedByUser` relationship, as shown in the following code:

```
<TypeDefinitions>
  <EntityTypes>
    <ClassTypes>
```

```

    <ClassType ID="Custom.MobilePhone" Accessibility="Public"
Abstract="false" Base="System!System.ConfigItem" Hosted="false"
Singleton="false" Extension="false">
        <Property ID="MobilePhoneID" Type="string"
AutoIncrement="true" Key="true" CaseSensitive="false"
MaxLength="256" MinLength="0" Required="false" Scale="0"
DefaultValue="MP{0}" />
        <Property ID="SerialNumber" Type="string"
AutoIncrement="false" Key="false" CaseSensitive="false"
MaxLength="256" MinLength="0" Required="false" Scale="0" />
        <Property ID="Manufacturer" Type="enum"
AutoIncrement="false" Key="false" CaseSensitive="false"
MaxLength="256" MinLength="0" Required="false" Scale="0"
EnumType="MobilePhoneManufacturerEnum" />
    </ClassType>
</ClassTypes>
<RelationshipTypes>
    <RelationshipType ID="MobilePhoneOwnedByUser"
Accessibility="Public" Abstract="false"
Base="System!System.Reference">
        <Source ID="MobilePhoneOwnedByUserSource" MinCardinality="0"
MaxCardinality="2147483647"
Type="Custom.MobilePhone" />
        <Target ID="MobilePhoneOwnedByUserTarget"
MinCardinality="0" MaxCardinality="1" Type="System!System.User" />
    </RelationshipType>
</RelationshipTypes>
<EnumerationTypes>
    <EnumerationValue ID="MobilePhoneManufacturerEnum"
Accessibility="Public" />
</EnumerationTypes>
</EntityTypes>
</TypeDefinitions>

```

How to do it...

We remember from the *Understanding the Service Manager Data Warehouse data mart* recipe that classes, relationships, and enumerations have their respective representations in the Service Manager Data Warehouse:

- ▶ Classes represent dimensions
- ▶ Relationships represent facts
- ▶ Enumerations represent outriggers

Using XML directives in the <Warehouse> area of a management pack, you can define which entities should be transferred to the data warehouse.

We will start with the `MobilePhone` class by defining the dimension as follows:

```
<Dimension ID="CustomMobilePhoneDim" Accessibility="Public"
  Target="Custom.MobilePhone" InferredDimension="true"
  HierarchySupport="IncludeExtendedClassProperties" Reconcile="true"
/>
```

The following table lists the relevant attributes that can be defined when creating dimensions for the data warehouse:

Attribute	Description
ID	This is a unique identifier for the dimension element. This will also be the table name of the dimension in the data warehouse and data mart.
Accessibility	This element should always be set to <code>Public</code> .
Target	This is the class name that the dimension is targeting.
InferredDimension	This is always <code>true</code> .
HierarchySupport	This has one of these three options: <ul style="list-style-type: none"> ▶ Exact: You must manually define each attribute that should be included in the dimension using the <code>InclusionAttribute</code> tag. ▶ IncludeExtendedClassProperties: All the attributes of the target class and all of its base classes will be included in the dimension. ▶ IncludeDerivedClassProperties: All the attributes of the target class, its base classes, and its derived classes will be included in the dimension.
Extends	This is an optional Boolean flag to indicate whether the dimension is a base dimension or is extending another dimension.
Reconcile	This is an optional Boolean flag to indicate whether two instances, that are otherwise identical and only differ through which source the data originated from, should be coalesced into one single row of data. Configuration Item-related dimensions should have this flag set to <code>true</code> , and Work Item-related dimensions will have this flag set to <code>false</code> .

Next, we need to define an outrigger for the mobile phone manufacturer enumeration using the following XML fragment:

```
<Outrigger ID="CustomMobilePhoneManufacturer"
Accessibility="Public">
  <Attribute ID="MobilePhoneManufacturer"
PropertyPath="$Context/Property [Type='Custom.MobilePhone']/Manufacturer$" />
</Outrigger>
```

The ID attribute of the <Outrigger> tag must contain a unique identifier for the outrigger element. This will also be the table name of the outrigger in the Data Warehouse data mart. The outrigger needs to point to all the class attributes that the enumeration is used by. This is done using <Attribute> tags inside the <Outrigger> tag. Each attribute must have a unique ID, and PropertyPath must uniquely identify the class and attribute that the outrigger attribute is targeting.

The last step involves the definition of the relationship fact for the mobile phone owner. See the following XML fragment for details:

```
<RelationshipFact ID="CustomMobilePhoneOwnedByUserFact"
Accessibility="Public"
Domain="DWBase!Domain.ConfigurationManagement" TimeGrain="Daily"
SourceType="Custom.MobilePhone"
SourceDimension="CustomMobilePhoneDim">
  <Relationships RelationshipType="MobilePhoneOwnedByUser"
TargetDimension="DWBase!UserDim" />
</RelationshipFact>
```

We need to define a <RelationshipFact> tag for each fact that we want to create and one or more <Relationships> tags that point to the actual class relationships. The following table lists the attributes used to define the relationship fact:

Attribute	Description
ID	This is a unique identifier for the relationship fact element. This will also be the table name of the relationship fact in the data warehouse and data mart.
Accessibility	This element should always be set to Public.
Domain	The value for this attribute must be an enumeration that is a child of the parent domain enumeration defined in the Microsoft.SystemCenter.Datawarehouse.Base management pack.

Attribute	Description
TimeGrain	This is the granularity of the relationship fact. The possible values are Hourly, Daily, Weekly, and Monthly.
SourceType	This is the class for the source of the relationship.
SourceDimension	This is the dimension that targets the source class. This is an optional field. If no source dimension is specified, Service Manager will automatically find the dimension that directly targets the source class itself, or the closest parent class of the source class in the class hierarchy.

The `<Relationships>` tag points to the actual relationship using the `RelationshipType` attribute. The `TargetDimension` attribute points to the dimension that targets the target class of the relationship.

The full management pack, including the definitions for class, relationship, and enumeration, as well as the data warehouse extension, is listed under *Extending the Service Manager Data Warehouse Code in Appendix, Useful Websites, Chapter Code, and Community Resources*.

Seal the management pack and import it into Service Manager.



Only sealed management packs will be transferred to the data warehouse.

Once `MPSyncJob` is finished and the ETL jobs have transferred all data to the data warehouse, you will see the new tables and views in the `DWDataMart` database, as shown in the following screenshot. This can take several hours to complete.



How it works...

You can use management packs to extend the Service Manager Data Warehouse with custom classes, relationships, and enumerations. When you import a sealed management pack to Service Manager, the `MPSyncJob` data warehouse job transfers its definition to the data warehouse. Certain things such as class extensions will be automatically applied to the `DWDataMart` database by Service Manager.

If a management pack contains a <Warehouse> directive, Service Manager will parse the contents and create objects such as dimensions, outriggers, and facts in the `DWDataMart` database. Also, the ETL jobs will be altered so that data for these new objects will be transported over from the Service Manager operational database.

See also

- ▶ Refer to *Understanding the Service Manager Data Warehouse data mart* recipe for more information on the Service Manager Data Warehouse

Creating Orchestrator runbook reports

Unlike most of the products in the System Center family, System Center Orchestrator does not include reporting out of the box. In this recipe, we will show you how you can access the `Orchestrator` database to create reports about your runbooks and the runbook instances.

Getting ready

Before we can access the `Orchestrator` database, we need to grant read permissions to a service account used for reporting. Perform the following steps:

1. Identify or create a service account you will use for reporting.
2. Start SQL Server Management Studio and connect to the SQL server that hosts the `Orchestrator` database.
3. Expand **Security** and **Logins**, and add the reporting service account.
4. Under **User Mapping**, select the **Orchestrator** database and assign the `db_datareader` role (you can leave the `public` role enabled).
5. Click on **OK**.

Next, we have to create a data source for the `Orchestrator` database in SQL Server Reporting Services. Perform the following steps:

1. Identify an instance of SQL Server Reporting Services that you will use for Orchestrator reporting.
2. Open your web browser and navigate to `http://[SSRS]/Reports`. Replace `[SSRS]` with the fully qualified domain name of the SQL Server Reporting Services server. If SQL Server Reporting Service is running as a named instance, the syntax of the URL will be `http://[SSRS]/Reports_[InstanceName]`.
3. Navigate to a folder of your choice or create a custom folder structure.

4. Click on **New Data Source** and complete the form as follows:
 - Name: Orchestrator
 - Data source type: **Microsoft SQL Server**
 - Connection string: data source=[SCOSQL];initial catalog=Orchestrator (replace [SCOSQL] with the fully qualified domain name of the SQL Server which hosts the Orchestrator database)
5. Select **Credentials stored securely in the report server**.
6. Enter the username and password of the service account used for reporting.
7. Select **Use as Windows credentials when connecting to the data source**.
8. Click on **Test Connection** to validate the connection.
9. Click on **OK** to save the new data source.

The following table lists the views that you can use in your reports to display information about the runbooks and the jobs in Orchestrator:

View	Description
[Orchestrator] . [Microsoft.SystemCenter.Orchestrator] . [Folders]	This contains all the folders in Orchestrator that are used to organize your runbooks.
[Orchestrator] . [Microsoft.SystemCenter.Orchestrator] . [Runbooks]	This contains all the runbooks available in Orchestrator.
[Orchestrator] . [Microsoft.SystemCenter.Orchestrator] . [RunbookParameters]	This contains the runbook parameters of all the runbooks available in Orchestrator.
[Orchestrator] . [Microsoft.SystemCenter.Orchestrator] . [RunbookDiagrams]	This contains the graphical representation of all the runbooks available in Orchestrator.

View	Description
[Orchestrator] . [Microsoft.SystemCenter.Orchestrator] . [Activities]	This contains the activities of all the runbooks available in Orchestrator.
[Orchestrator] . [Microsoft.SystemCenter.Orchestrator] . [Resources]	This contains all the resources available in Orchestrator.
[Orchestrator] . [Microsoft.SystemCenter.Orchestrator.Runtime] . [RunbookServers]	This contains all the runbook servers.
[Orchestrator] . [Microsoft.SystemCenter.Orchestrator.Runtime] . [Jobs]	This contains all jobs.
[Orchestrator] . [Microsoft.SystemCenter.Orchestrator.Runtime] . [RunbookInstances]	This contains all runbook instances that were triggered as part of jobs.
[Orchestrator] . [Microsoft.SystemCenter.Orchestrator.Runtime] . [RunbookInstanceParameters]	This contains all parameter data that was passed to the runbook instances.
[Orchestrator] . [Microsoft.SystemCenter.Orchestrator.Runtime] . [ActivityInstances]	This contains all activities that ran as part of the runbook instances.
[Orchestrator] . [Microsoft.SystemCenter.Orchestrator.Runtime] . [ActivityInstanceData]	This contains the activity-related data of all the activity instances.
[Orchestrator] . [Microsoft.SystemCenter.Orchestrator.Runtime] . [Events]	This contains Orchestrator events.
[Orchestrator] . [Microsoft.SystemCenter.Orchestrator.Statistics] . [Statistics]	This contains Orchestrator statistics.

How to do it...

We will now create a report that displays details about an Orchestrator runbook. We will use several datasets in our report to get data as listed in the following table:

Dataset	Description
PSRunbooks	This dataset is used as the source of a parameter in our report. This parameter allows us to pick the runbook for the report.
DSRunbook	This dataset returns the metadata of the runbook, such as its name and folder path.
DSDiagram	This dataset returns the graphical representation of the runbook workflow.
DSParameters	This dataset returns all the parameters of the runbook.
DSActivities	This dataset returns all the activities of the runbook.
DSInstances	This dataset returns the 20 most recent runbook instances.

Perform the following steps:

1. Start Report Builder.
2. Under **New Report**, select **Blank Report**.
3. Under **Data Sources**, add the newly created `Orchestrator` data source.
4. Under **Datasets**, add a dataset named `PSRunbooks`. Select the **Use a dataset embedded in my report** option, select the `Orchestrator` data source, select the **Text** query type, and type the following query. Then, click on **OK**:


```
SELECT Id, Name FROM
[Orchestrator].[Microsoft.SystemCenter.Orchestrator].Runbooks ORDER BY Name
```
5. Under **Parameters**, add a parameter named `Runbook`. Select the **Text** data type and disable all options regarding blank, null, and multiple values. Also, set the visibility to **Visible**. Under **Available values**, choose **Get values from a query**, pick the **PSRunbooks** dataset, and pick **Id** as the **Value field** and **Name** as the **Label field**. Click on **OK**.

6. Under **Datasets**, add a dataset named DSRunbook. Select the **Use a dataset embedded in my report** option, select the Orchestrator data source, select the **Text** query type, and type the following query. Then, click on **OK**:

```
SELECT * FROM  
[Orchestrator].[Microsoft.SystemCenter.Orchestrator].Runbook  
ks WHERE Id = @Runbook
```

7. Under **Datasets**, add a dataset named **DSDiagram**. Select the **Use a dataset embedded in my report** option, select the Orchestrator data source, select the **Text** query type, and type the following query. Then, click on **OK**:

```
SELECT Diagram FROM  
[Orchestrator].[Microsoft.SystemCenter.Orchestrator].Runbook  
kDiagrams WHERE RunbookId = @Runbook
```

8. Under **Datasets**, add a dataset named DSParameters. Select the **Use a dataset embedded in my report** option, select the Orchestrator data source, select the **Text** query type, and type the following query. Then, click on **OK**:

```
SELECT * FROM  
[Orchestrator].[Microsoft.SystemCenter.Orchestrator].Runbook  
kParameters WHERE RunbookId = @Runbook
```

9. Under **Datasets**, add a dataset named DSActivities. Select the **Use a dataset embedded in my report** option, select the Orchestrator data source, select the **Text** query type, and type the following query. Then, click on **OK**:

```
SELECT * FROM [Orchestrator].[Microsoft.SystemCenter.  
Orchestrator].Activities WHERE RunbookId = @Runbook
```

10. Under **Datasets**, add a dataset named DSInstances. Select the **Use a dataset embedded in my report** option, select the Orchestrator data source, select the **Text** query type, and type the following query. Then, click on **OK**:

```
SELECT TOP 20 * FROM [Orchestrator].[Microsoft.SystemCenter.  
Orchestrator.Runtime].RunbookInstances WHERE RunbookId = @Runbook  
ORDER BY CreationTime DESC
```

Now, it's time to design our report. Place an image on the report body, Click the **Insert Tab | Click Image | Draw** an image box in the report body. Choose **Database** under **Select the image source**, and under **Use this field**, click on the **fx** icon and type `=First(Fields!Diagram.Value, "DSDiagram")`. Choose **image/bmp** under **Use this MIME type**.

Use regular tables in your report to display data for the DSRunbook, DSParameters, DSActivities, and DSInstances datasets. Refer to the following screenshot for layout and design recommendations:

Runbook Details

Name	Monitor	Created	Last Modified	Checked Out
1. Applications	True	5/22/2013 10:50 AM	1/29/2014 9:24 PM	

Description:
Path: \\1. System Center\1.1.1 SCCSM\1.1.3 SCCM Interface\1. Applications

```

graph LR
    Start[Start at 4:10am] --> Truncate[Truncate Stage]
    Truncate --> ExportSCCM[Export SCCM]
    Truncate --> ExportAltiris[Export Altiris]
    ExportSCCM --> InsertSCCM[Insert SCCM]
    ExportAltiris --> InsertAltiris[Insert Altiris]
    InsertSCCM --> Junction[Junction]
    InsertAltiris --> Junction
    Junction --> CreateCSV[Create CSV]
    CreateCSV --> CopyFile[Copy File]
    CopyFile --> ImportCSV[Import CSV]
    
```

Parameters

Name	Type	Direction
Start at 4:10am	Monitor Date/Time	True
Streamed Applications	Trigger Policy	True
Truncate Stage	Query Database	True

Activities

Name	Type	Enabled
Copy File	Copy File	True
Create CSV	Execute PS Script	True
Export Altiris	Run Program	True
Export SCCM	Run Program	True
Import CSV	Execute PS Script	True
Insert Altiris	Query Database	True
Insert SCCM	Query Database	True
Junction	Junction	True
Start at 4:10am	Monitor Date/Time	True
Streamed Applications	Trigger Policy	True
Truncate Stage	Query Database	True

Runbook Instances (20 most recent)

Created	Completed	Duration	Status
2/8/2014 3:10 AM	2/9/2014 3:10 AM	1440.82 Minutes	success
2/7/2014 3:10 AM	2/8/2014 3:15 AM	1445.23 Minutes	success
2/6/2014 12:10 PM	2/7/2014 3:13 AM	902.3 Minutes	success
2/6/2014 12:06 PM	2/6/2014 12:09 PM	2.7 Minutes	failed
2/6/2014 11:59 AM	2/6/2014 12:00 PM	0.38 Minutes	failed

Current report server: http://localhost/reportserver

How it works...

You can use Report Server to access any accessible data source using the supported data source providers. Before you can use the System Center Orchestrator database, you need to configure permissions in SQL Server and then create a data source in SQL Server Reporting Services that points to the `Orchestrator` database.

8

Creating System Center Advanced Reports

In this chapter, we will cover the following recipes:

- ▶ Creating dashboards from basic reports
- ▶ Working with drillthrough reports
- ▶ Working with shared datasets and parameters
- ▶ Using Chargeback reports with System Center 2012 R2

Introduction

This chapter will show you recipes that build on and enhance the reports from the previous recipes in this book. The recipes in this chapter delve into advanced reporting techniques and combined data source reporting. The advanced topics in this chapter include using multiple data sources and creating active dashboards that present multiple reports in a single pane. The authors recommend that you first review and/or perform the steps in the earlier chapters as a primer. You will build on the techniques and examples from those chapters. The chapter recipes use the System Center product databases.

Creating dashboards from basic reports

In this recipe, we will create a Service Management Dashboard that displays the current open incidents from System Center Service Manager and current active alerts from System Center Operations Manager.

Getting ready

You must have a fully deployed Service Manager and Operations Manager environment. The Operations Manager environment must be tuned and optimized to alert and monitor to your organization's standards.

It is recommended that you plan the organization of your reporting environment in SQL Server Reporting Services. Organizing the environment is recommended as this simplifies reporting life cycle management as well as delegation options when you share reports with your stake holders. We recommend that you create separate folders for elements such as shared data sources and also use folders to logically group your reports. This is covered in the *Organizing the reporting environment and delegating access to reports* recipe in *Chapter 3, Unpacking System Center Report Building Tools*.

How to do it...

In order to complete the recipe's final dashboard report, we will create two basic reports, the first of which will display data from Service Manager and the second from Operations Manager.

Creating the Open Incidents report

Chapter 7, Creating Reports for System Center Service Manager and Orchestrator, discussed the ETL process that System Center Service Manager uses to transfer data from the operational database to the Data Warehouse database. As a result of this ETL process, new objects such as incidents or changes to the existing objects can take up to 1 hour before they are available in the Service Manager Data Warehouse database. As we want the data in the dashboard report to be real time, we need to access the Service Manager operational database in our report.



Accessing the Service Manager operational database

Note that accessing the operational database is not supported by Microsoft. This is because there is a possibility that the database schema might change between release cycles. Also, in order to prevent locks from occurring when accessing the database (read only), it is recommended that you always use the `NOLOCK` query option. Avoid targeting this database as it can lead to performance issues if the SQL server does not have adequate resources.

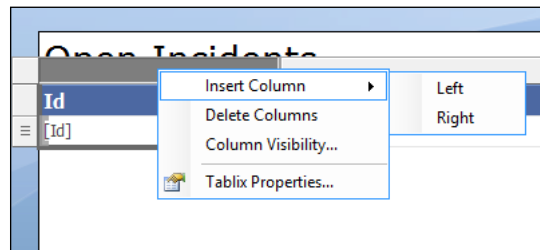
We first need to create a shared data source for the Service Manager Operational database. Refer to the *Creating reports using the operational database* recipe in *Chapter 7, Creating Reports for System Center Service Manager and Orchestrator*, for instructions. You must perform these steps using the **SQL Server Reporting Service (SSRS)** instance for Service Manager. SSRS is deployed as part of the Data Warehouse setup for Service Manager.

Once you have created the shared data source, follow these instructions to create the Open Incidents report:

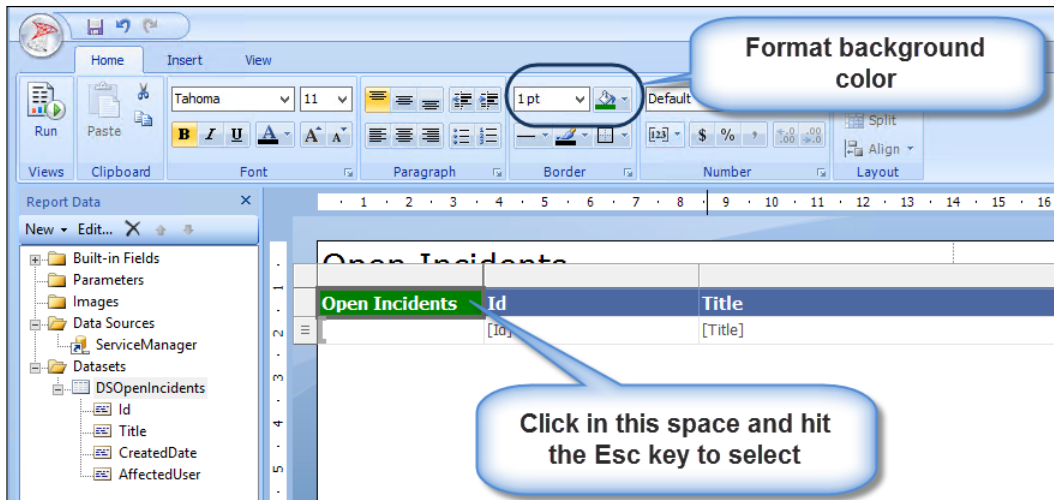
1. Start Report Builder (using the Report Manager website or from a standalone installation).
2. Under **New Report**, select **Blank Report**.
3. Right-click on **Data Sources** and click on **Add Data Source...**
4. Type `ServiceManager` in the Name: field.
5. Select **Use a shared connection or report model**, click on **Browse...**, and browse to and select the newly created data source (this would be called **ServiceManager** if you followed the steps to create this data source).
6. Click on **Test Connection**, and on success, click on **OK**.
7. Right-click on **Datasets** and click on **Add Dataset...**
8. Type `DSOpenIncidents` as the name and select the **Use a dataset embedded in my report** option.
9. Select the **ServiceManager** data source, select **Text** query type, and type the following query. Then, click on **OK**:

```
SELECT
    Id_9A505725_E2F2_447F_271B_9B9F4F0D190C AS [Id],
    Title_9691DD10_7211_C835_E3E7_6B38AF8B8104 AS [Title],
    CreatedDate_6258638D_B885_AB3C_E316_D00782B8F688 AS
[CreatedDate],
    U.DisplayName AS [AffectedUser]
FROM
    MT_System$WorkItem$Incident I WITH (NOLOCK)
    LEFT OUTER JOIN Relationship R WITH (NOLOCK) ON
        I.BaseManagedEntityId = R.SourceEntityId
        AND R.RelationshipTypeId = 'DF99BE66-38B0-B6D6-6144-
A412A3EBD4CE'
        AND R.IsDeleted = 0
    LEFT OUTER JOIN MT_System$Domain$User U WITH (NOLOCK) ON
        R.TargetEntityId = U.BaseManagedEntityId
WHERE
    Status_785407A9_729D_3A74_A383_575DB0CD50ED NOT IN
('2B8830B6-59F0-F574-9C2A-F4B4682F1681', 'BD0AE7C4-3315-
2EB3-7933-82DFC482DBAF')
ORDER BY
    3 DESC
```

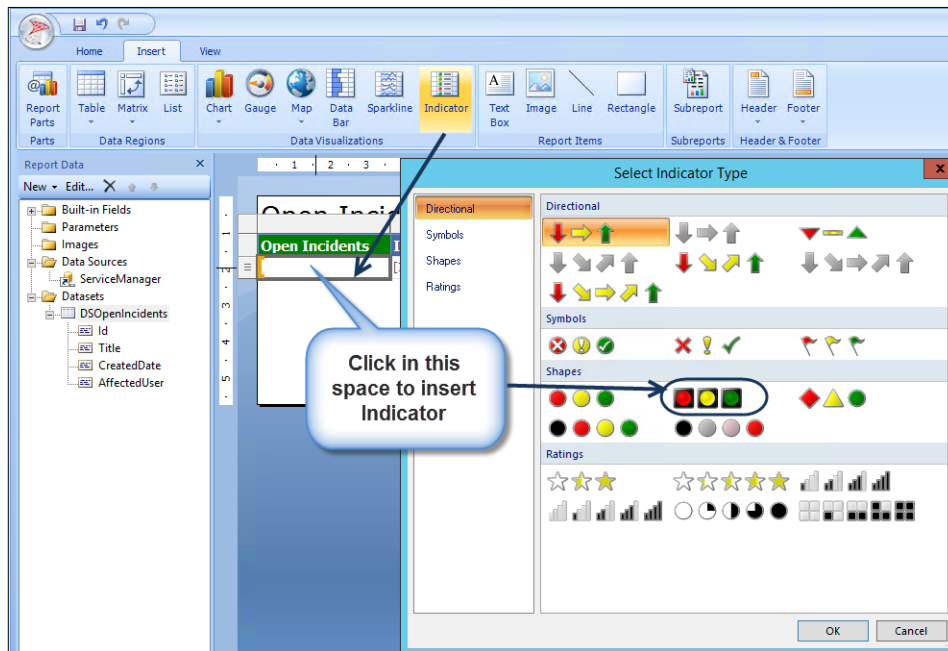

10. When prompted to **Enter Data Source Credentials**, select **Use the current Windows user** or type the password for the service account and then click on **OK**.
11. In Report Builder, in the **Design** area, replace the text `Click to add title` with `Open Incidents`.
12. In the toolbar, under **Insert**, click on **Table**, and then click on **Table Wizard...**
13. In the **New Table or Matrix wizard**, select **Choose an existing dataset in this report or a shared dataset**, select the **DSOpenIncidents** dataset, and then click on **Next**.
14. Drag and drop all the fields from the **Available fields** list to the **Values** area and then click on **Next**.
15. On the **Choose the layout** page, click on **Next**, select the desired style (for example, **Ocean**), and then click on **Finish**.
16. Change the columns' width and font sizes so that the rendered report is displayed properly. You can click on the **Run** button in the **Home** toolbar at any time to render data. Click on **Design** in the **Run** toolbar to return to the **Design** mode.
17. Right-click on the footer area in the report and click on **Remove Page Footer**.
18. Right-click on the **Id** column, click on **Insert Column**, and then click on **Left** to insert a new column, as shown in the following screenshot:



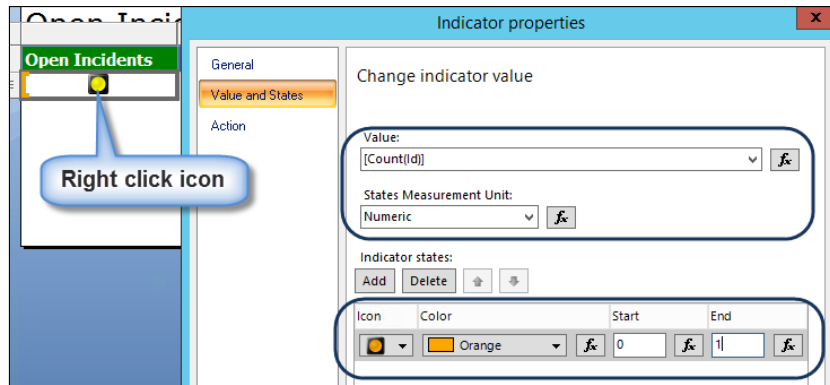
19. Type `Open Incidents` as the title of the new column. Expand the column so that the text fits.
20. Select the new column title by clicking on the space next to the title text and press the `Esc` key. Use the formatting tools to change the fill color to green, as shown in the following screenshot:



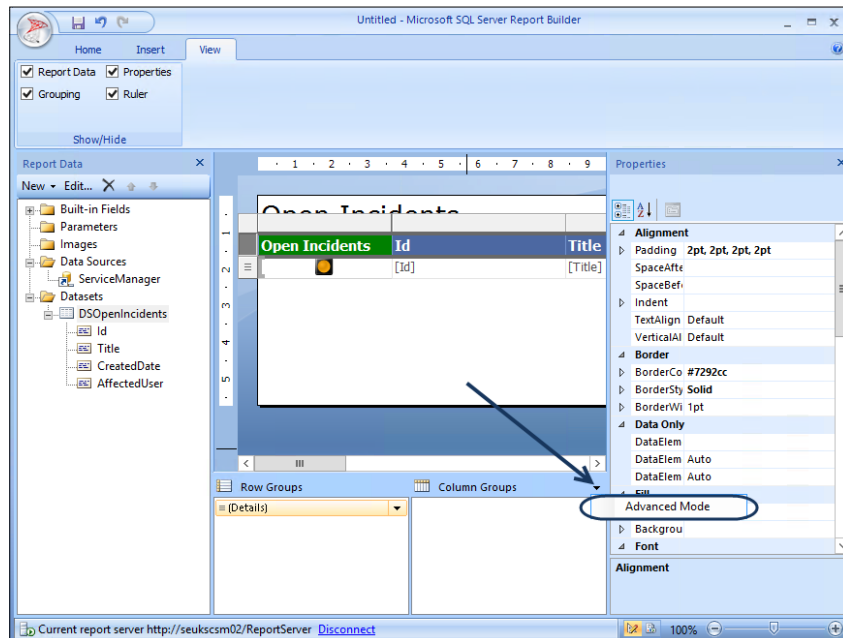
21. Click on the **Insert** tab, then on **Indicator**, and finally on the cell below the new column title. Select the rimmed traffic lights under **Shapes** and click on **OK**, as shown in the following screenshot:



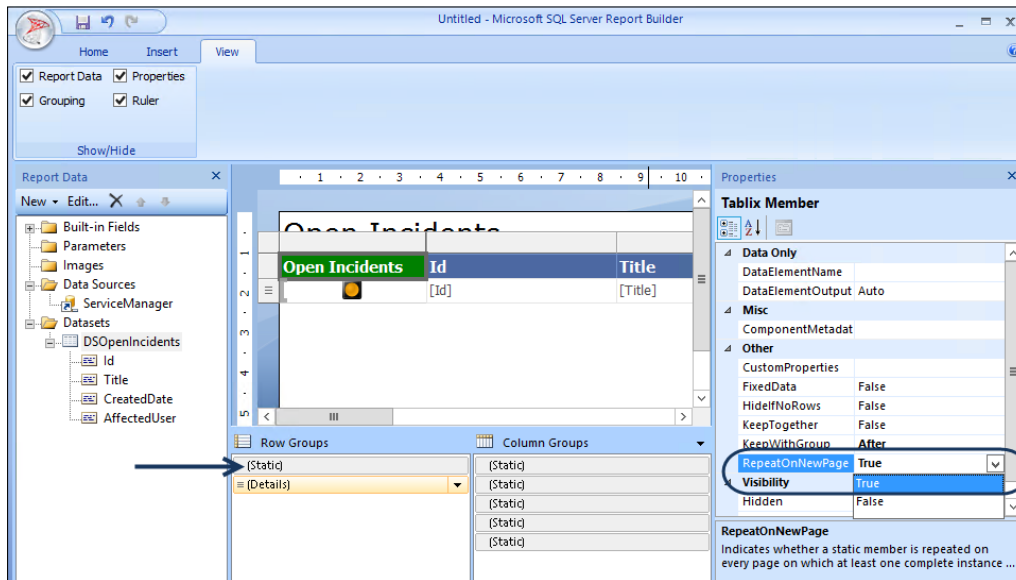
22. Right-click on the **Indicator** icon. Select **Indicator properties**. Make the following changes to the properties:
- ❑ For **Value:**, select **[Count(Id)]**.
 - ❑ For **States Measurement Unit:**, select **Numeric**.
 - ❑ For **Indicator states:**, delete the green and yellow indicators. Change the red indicator color to orange. Type 0 in the **Start** field and 1 in the **End** field. Click on **OK**, as shown in the following screenshot:



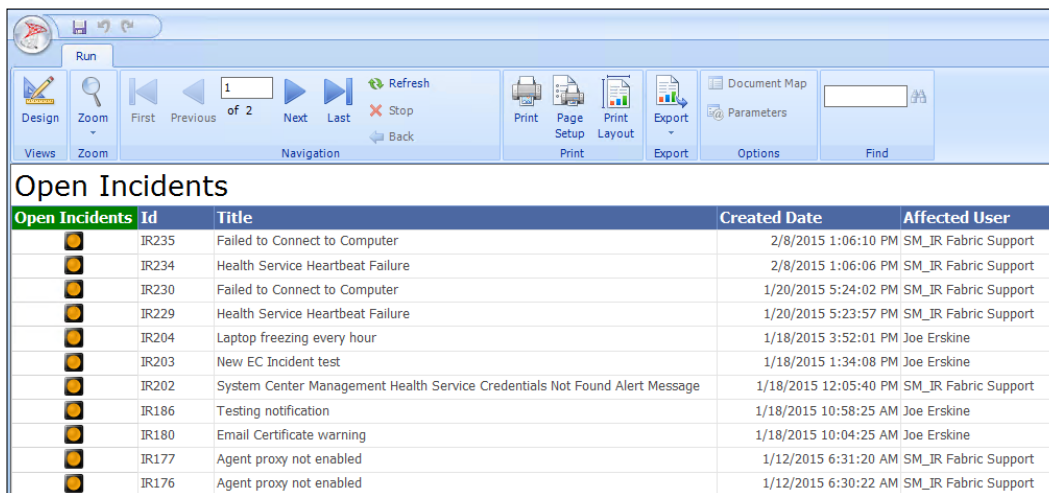
23. Click on the little arrow next to **Column Groups** and select **Advanced Mode**, as shown in the following screenshot:



24. Select the row header of the table. Click on **(Static)** under **Row Groups**. Change the **RepeatOnNewPage** property to **True**, as shown in the following screenshot:



25. Test the report with the **Run** button and toggle back to the design mode with the **Design** button, as shown in the following screenshot:



26. Click on **Save** from the **File** menu (or press **Ctrl + S**), browse to a folder in SSRS of your choice, and save the report with the name **Open Incidents**.

Creating the Active Alerts report

Before you can start authoring custom reports for System Center Operations Manager, it is recommended that you create a shared data source that you will use for all your reports. As we want the data in our report to be in real time, we will create a data source that points to our operational Operations Manager database.



Accessing the Operations Manager operational database

Note that accessing the operational database is not supported by Microsoft. This is because there is a possibility that the database schema might change between release cycles. Also, in order to prevent locks from occurring when accessing the database (read only), it is recommended that you always use the `NOLOCK` query option.

Before we can access the operational database, we need to grant read permissions to a user account. We will use the SCOM Data Reader account for this purpose. Perform the following steps:

1. Start SQL Server Management Studio and connect to the SQL server that hosts the `OperationsManager` database.
2. Expand **Security** and **Logins**, and check whether the SCOM Data Reader account is present in the list of logins. If not, add the login.
3. Double-click on the **SCOM Data Reader** login. Then, under **User Mapping**, check the **OperationsManager** database and assign the `db_datareader` role (you can leave the `public` role enabled).
4. Click on **OK**.
5. Next, we have to create a data source for the `OperationsManager` database in SQL Server Reporting Services. The instance of SQL Server Reporting Services discussed in this section is the same instance you used to create the Open Incidents report. Open your web browser and navigate to `http://[SSRS]/Reports`. Replace `[SSRS]` with the Fully Qualified Domain Name of the SQL server that is used for reporting. If SQL Server Reporting Service is running as a named instance, the syntax of the URL will be `http://[SSRS]/Reports_[InstanceName]`.
6. Navigate to the folder where you would like to create your shared data source. We recommend that you use a separate folder for data sources. Use the same folder you used when you created the data source for the `ServiceManager` database.
7. Click on **New Data Source** and complete the form as follows:
 - Name: `OperationsManager`
 - Data source type: **Microsoft SQL Server**

- Connection string: `data source=[SCOMSQL];initial catalog=OperationsManager` (replace [SCOMSQL] with the Fully Qualified Domain Name of the SQL Server that hosts the OperationsManager database)

Now perform the following steps:

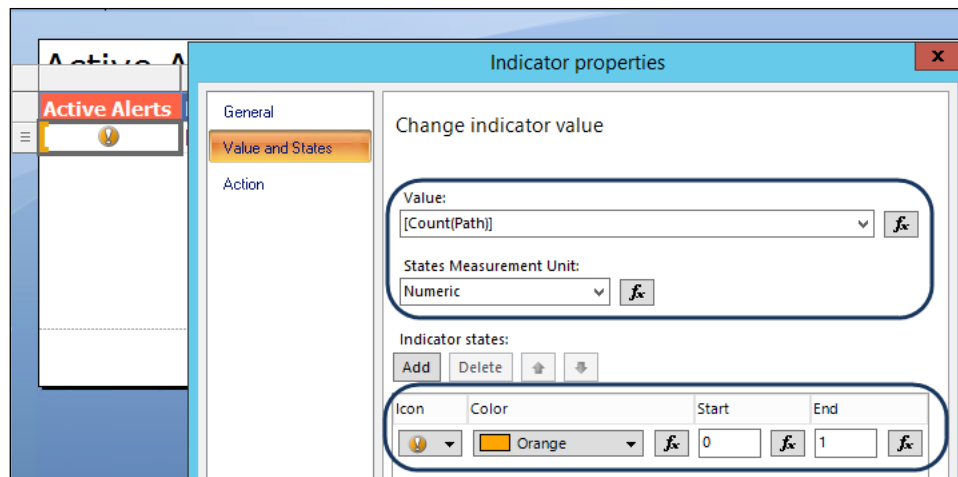
1. Select **Credentials stored securely in the report server**.
2. Enter the username and password of the SCOM Data Reader account.
3. Select **Use as Windows credentials when connecting to the data source**.
8. Click on **Test Connection** to validate the connection.
9. Click on **OK** to save the new data source.

Once you have created the shared data source, follow these instructions to create the Active Alerts report:

1. Start Report Builder (using the Report Manager website or from a standalone installation).
2. Under **New Report**, select **Blank Report**.
3. Right-click on **Data Sources** and click on **Add Data Source....**
4. Enter `OperationsManager` as the name.
5. Select **Use a shared connection or report model**, click on **Browse...**, and browse to the newly created **OperationsManager** data source.
6. Click on **Test Connection**, and on success, click on **OK**.
7. Right-click on **Datasets** and click on **Add Dataset....**
8. Enter `DSActiveAlerts` as the name and select the **Use a dataset embedded in my report** option.
9. Select the **OperationsManager** data source, select **Text** query type, and type the following query. Then, click on **OK**:

```
SELECT
    MonitoringObjectPath AS [Path],
    MonitoringObjectDisplayName AS [Source],
    ISNULL(AlertStringName, Name) AS [Name],
    TimeRaised AS [Created]
FROM
    AlertView
WHERE
    ResolutionState <> 255
ORDER BY
    4 DESC
```

10. When prompted to **Enter Data Source Credentials**, select **Use the current Windows user** or type the password for the service account, and click on **OK**.
11. In Report Builder, in the **Design** area, replace `Click to add title` with `Active Alerts`.
12. In the toolbar, under **Insert**, click on **Table** and then click on **Table Wizard...**
13. In **New Table or Matrix wizard**, select **Choose an existing dataset in this report or a shared dataset**, select the **DSActiveAlerts** dataset, and then click on **Next**.
14. Drag and drop all the fields from the **Available fields** list to the **Values** area, and then click on **Next**.
15. On the **Choose the layout** page, click on **Next**, select the desired style, and then click on **Finish**.
16. Change the columns' width and font sizes so that the rendered report is displayed properly. You can click on the **Run** button in the **Home** toolbar at any time to render data. Click on **Design** in the **Run** toolbar to return to the **Design** mode.
17. Right-click on the footer area in the report and click on **Remove Page Footer**.
18. Right-click on the **Path** column. Go to **Insert Column | Left** to insert a new column.
19. Type **Active Alerts** as the title of the new column. Expand the column so that the text fits.
20. Select the new column title by clicking on a space next to the column title text and pressing the `Esc` key. Use the formatting tools to change the fill color to **Orange**.
21. Click on the **Insert** tab. Click on **Indicator** and click on the cell below the new column title. Select the three symbols (circled) under **Symbols** and click on **OK**.
22. Right-click on the **Indicator** icon. Select **Indicator properties...** Make the following changes to the properties in the **Value and States** tab:
 - For **Value:**, select **[Count(Path)]**.
 - For **States Measurement Unit:**, select **Numeric**.
 - For **Indicator states:**, delete the red and green indicators. Change the yellow indicator color to orange. Type 0 in the **Start** field and 1 in the **End** field. Click on **OK**, as shown in the following screenshot:



23. Click on the little arrow next to **Column Groups** and select **Advanced Mode**.
24. Select the row header of the table. Click on **(Static)** under **Row Groups**. Change the **RepeatOnNewPage** property to **True**.
25. Test the report with the **Run** button and toggle back to the design mode with the **Design** button, as shown in the following screenshot:

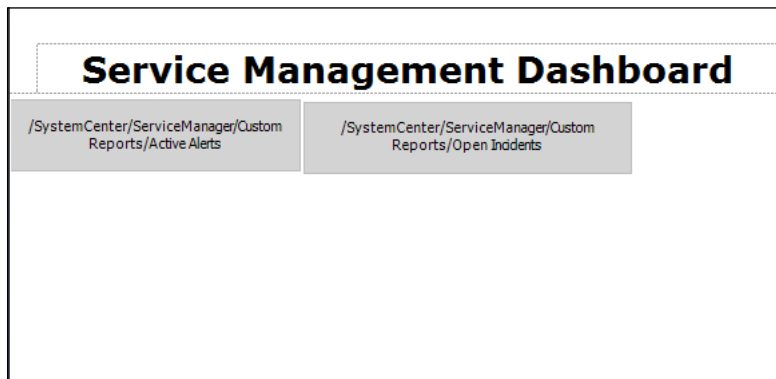
Active	Path	Source	Name	Created
🚩	SEUKSCOM01.FW2RW.COM	SEUKSCOM01.FW2RW.COM	Advisor Http Module Traffic Monitor	2/15/2015 8:37:57 PM
🚩	SEUKEX01.FW2RW.COM;SEUKEX01.FW2RW.COM	SEUKEX01 - Autodiscover	{2}	2/13/2015 5:19:05 AM
🚩	SEUKEX01.FW2RW.COM;SEUKEX01.FW2RW.COM	SEUKEX01 - FEP	{2}	2/13/2015 4:51:42 AM
🚩	SEUKSCOM01.FW2RW.COM	SEUKSCOM01.FW2RW.COM	Object Health State data collection process unable to store data in the Data Warehouse in a timely manner	2/13/2015 4:39:33 AM
🚩	SEUKSCOM01.FW2RW.COM	SEUKSCOM01.FW2RW.COM	Performance data collection process unable to store data in the Data Warehouse in a timely manner	2/10/2015 8:57:39 PM
🚩	SEUKSCOM01.FW2RW.COM	SEUKSCOM01.FW2RW.COM	Event data collection process unable to store data in the Data Warehouse in a timely manner	2/10/2015 8:54:20 PM
🚩		Operations Manager APM Data Transfer Service	APM Data Transfer Health	2/10/2015 8:50:47 PM
🚩	SEUKEX01.FW2RW.COM;SEUKEX01.FW2RW.COM	SEUKEX01 - RemoteMonitoring	{0}	2/9/2015 5:53:37 AM
🚩	Microsoft.SystemCenter.AgentWatchersGroup	SEUKDPM01.FW2RW.COM	Failed to Connect to Computer	2/8/2015 1:06:10 PM
🚩	Microsoft.SystemCenter.AgentWatchersGroup	SEUKDPM01.FW2RW.COM	Health Service Heartbeat Failure	2/8/2015 1:06:06 PM

26. Click on **Save** from the **File** menu (or press **Ctrl + S**), browse to a folder in SSRS of your choice, and save the report with the name `Active Alerts`.

Creating the combined dashboard

Next, we will create a new report that includes data from the other two reports using Report Builder. Perform the following steps:

1. Start Report Builder.
2. Under **New Report**, select **Blank Report**.
3. Click on the **Insert** tab and then on **Subreport**.
4. Draw a rectangle on the report body using the **Subreport** icon to hold the **Active Alerts** subreport.
5. Right-click on **<Subreport>** and click on **Subreport Properties...**
6. Change the name of the subreport to `SRActiveAlerts`.
7. Click on **Browse...** and browse to the **Active Alerts** report. Then, click on **OK** to close the **Subreport Properties** window.
8. On the **Insert** toolbar, click on **Subreport**.
9. Draw a rectangle on the report body to the right-hand side of the first subreport to hold the **Open Incidents** subreport.
10. Right-click on **<Subreport>** and click on **Subreport Properties...**
11. Change the name of the subreport to `SROpenIncidents`.
12. Click on **Browse...** and browse to the **Open Incidents** report. Then, click on **OK** to close the **Subreport Properties** window.
13. In the **Design** area, replace the `Click to add title` with `Service Management Dashboard`. Change the font size of the text box to **22** and enable bold font.
14. Your **Report Design** area should now look similar to the following illustration:



15. Click on **Save** from the **File** menu (or press **Ctrl + S**), browse to a folder in SSRS of your choice, and save the report with the name *Service Management Dashboard*.
16. Click on **Run** from the **Home** toolbar to test the newly created report. The following screenshot shows an example of the combined dashboard. It is recommended to show this report on a large screen which could be placed in the operations team's office:

Active Alerts					Open Incidents			
Alert ID	Path	Source	Name	Created	Incident ID	Title	Created Date	Affected User
31843081	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	Advised Help Module Traffic Monitor	3/5/2015 8:57:57 PM	30235	Failed to Connect to Computer	2/27/2015 1:06:13 PM	SM_R Fabric Support
31843082	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	Autobrowser	3/5/2015 9:59:08 AM	30234	Health Service Heartbeat Failure	2/27/2015 1:06:08 PM	SM_R Fabric Support
31843083	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	FTP	3/5/2015 4:21:40 AM	30230	Failed to Connect to Controller	1/22/2015 3:24:42 PM	SM_R Fabric Support
31843084	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	Health	3/5/2015 4:38:33 AM	30229	Health Service Heartbeat Failure	1/22/2015 9:23:07 PM	SM_R Fabric Support
31843085	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	Health	3/5/2015 8:57:39 PM	30204	License: Missing every hour	1/18/2015 5:52:01 PM	Jim Drake
31843086	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	Performance Data Collection process unable to store data in the Data Warehouse in a timely manner	2/18/2015 9:57:39 PM	30203	New IIC Incident Alert	1/18/2015 5:54:08 PM	Jim Drake
31843087	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	Event data collection process unable to store data in the Data Warehouse in a timely manner	3/5/2015 9:54:20 PM	30186	Testing notification	1/18/2015 10:39:25 AM	Jim Drake
31843088	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	Operations Manager ARM Data Transfer Service ARM Data Transfer health	3/5/2015 8:50:47 PM	30180	Event Controller warning	1/18/2015 10:08:25 AM	Jim Drake
31843089	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	RemoteMonitoring	3/5/2015 5:53:37 AM	30177	Agent proxy not enabled	1/12/2015 6:51:31 AM	SM_R Fabric Support
31843090	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	Health Service Heartbeat Failure	3/5/2015 5:53:37 AM	30176	Agent proxy not enabled	1/12/2015 6:50:22 AM	SM_R Fabric Support
31843091	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	Health Service Heartbeat Failure	3/5/2015 5:53:37 AM	30175	Agent proxy not enabled	1/12/2015 6:50:22 AM	SM_R Fabric Support
31843092	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	Health Service Heartbeat Failure	3/5/2015 5:53:37 AM	30174	Agent proxy not enabled	1/12/2015 6:50:22 AM	SM_R Fabric Support
31843093	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	Health Service Heartbeat Failure	3/5/2015 5:53:37 AM	30173	Agent proxy not enabled	1/12/2015 6:50:22 AM	SM_R Fabric Support
31843094	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	Health Service Heartbeat Failure	3/5/2015 5:53:37 AM	30172	Agent proxy not enabled	1/12/2015 6:50:22 AM	SM_R Fabric Support
31843095	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	Health Service Heartbeat Failure	3/5/2015 5:53:37 AM	30171	Agent proxy not enabled	1/12/2015 6:50:22 AM	SM_R Fabric Support
31843096	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	Health Service Heartbeat Failure	3/5/2015 5:53:37 AM	30168	Unknown Faulting	1/12/2015 9:23:34 PM	Sam Eskine
31843097	SSRS\CONSL\FY09\ALCON	SSRS\CONSL\FY09\ALCON	Health Service Heartbeat Failure	3/5/2015 5:53:37 AM	30146	Sam not getting alerts	1/12/2015 9:24:02 PM	Sam Eskine

How it works...

The *Creating dashboards from basic reports* recipe is split into three parts. The combined result of these parts is the final dashboard.

The breakdown of the parts is as follows

- ▶ **Creating the Open Incidents report:** This first set of steps creates the data source object that targets the Service Manager operational database. The dataset (query) filters out the closed and resolved incidents. The filter is denoted by the GUID (internal name of the two status types).
- ▶ **Creating the Active Alerts report:** This second set of steps creates the data source object that targets the Operations Manager operational database. The dataset (query) filters out the resolved alerts. The filter is denoted by `ResolutionState <> 255` (the value of the Operations Manager resolved state of an alert).
- ▶ **Creating the combined dashboard:** The third and final part of the recipe uses the subreport option to create the dashboard. This is, in effect, a shortcut to the two reports, and when executed, it renders the data and displays it as a single report.

The `NOLOCK` option is very important as the recipe accesses the operational databases. This is a recommended practice to ensure that you do not impact the operational databases.



Develop in a test environment

We recommend that you use a test environment to develop and test your reports.

Working with drillthrough reports

Building on the Service Management Dashboard example we created in the previous recipe, we will create a drillthrough report to display the details about an incident when the report consumer clicks on the incident ID.

Getting ready

In this recipe, we will modify the `Open Incidents` report created in the previous recipe to allow the report consumer to click on an incident ID to retrieve further information about the incident.

In order to create the drillthrough report in this recipe, you need to have completed the instructions from the previous recipe, *Creating dashboards from basic reports*.

How to do it...

The steps involved in creating the drillthrough report can be grouped into the following two tasks:

- ▶ Creating the incident drillthrough report
- ▶ Linking the main report and the drillthrough report

Creating the Incident Details drillthrough report

The Incident Details report will include detailed information about an incident, such as the description, the assigned-to analyst, and the classification category. We will need to create a parameter for the Incident Details report to pass the incident ID to the report. Perform the following steps:

1. Start Report Builder.
2. Under **New Report**, select **Blank Report**.
3. Right-click on **Data Sources**, click on **Add Data Source...**, and enter `ServiceManager` as the name.
4. Select **Use a shared connection or report model** and select the **ServiceManager** data source from the list of available data sources. If the data source is not displayed, click on **Browse...** and browse to the location where the data source is stored in SQL Server Reporting Services.

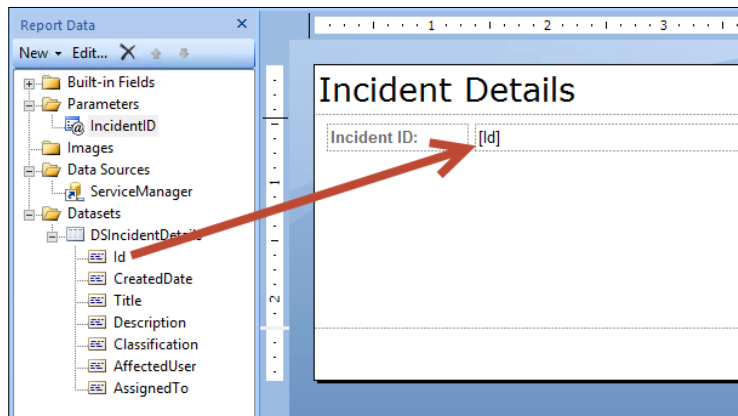
5. Click on **Test Connection** and then click on **OK** twice.
6. Right-click on **Datasets** and click on **Add Dataset...**
7. Enter `DSIncidentDetails` as the name and select the **Use a dataset embedded in my report** option.
8. Select the **ServiceManager** data source, select **Text** query type, and type the following query. Then, click on **OK**:

```

SELECT
    I.Id_9A505725_E2F2_447F_271B_9B9F4F0D190C AS [Id],
    I.CreatedDate_6258638D_B885_AB3C_E316_D00782B8F688 AS
[CreatedDate],
    I.Title_9691DD10_7211_C835_E3E7_6B38AF8B8104 AS [Title],
    I.Description_59B77FD5_FE0E_D2B5_D541_0EBBD1EC9A2B AS
[Description],
    TClassification.LTValue AS [Classification],
    AFU.DisplayName AS [AffectedUser],
    ASU.DisplayName AS [AssignedTo]
FROM
    MT_System$WorkItem$Incident I WITH (NOLOCK)
    LEFT OUTER JOIN Relationship AFR WITH (NOLOCK) ON
        I.BaseManagedEntityId = AFR.SourceEntityId
        AND AFR.RelationshipTypeId = 'DF9BE66-38B0-B6D6-6144-
A412A3EBD4CE'
        AND AFR.IsDeleted = 0
    LEFT OUTER JOIN MT_System$Domain$User AFU WITH (NOLOCK)
ON
    AFR.TargetEntityId = AFU.BaseManagedEntityId
    LEFT OUTER JOIN Relationship ASR WITH (NOLOCK) ON
        I.BaseManagedEntityId = ASR.SourceEntityId
        AND ASR.RelationshipTypeId = '15E577A3-6BF9-6713-4EAC-
BA5A5B7C4722'
        AND ASR.IsDeleted = 0
    LEFT OUTER JOIN MT_System$Domain$User ASU WITH (NOLOCK)
ON
    ASR.TargetEntityId = ASU.BaseManagedEntityId
    LEFT OUTER JOIN LocalizedText TClassification ON
        I.Classification_00B528BF_FB8F_2ED4_2434_5DF2966EA5FA =
TClassification.LTStringId
        AND TClassification.LanguageCode = 'ENU'
        AND TClassification.LTStringType = 1
WHERE
    Id_9A505725_E2F2_447F_271B_9B9F4F0D190C = @IncidentID

```

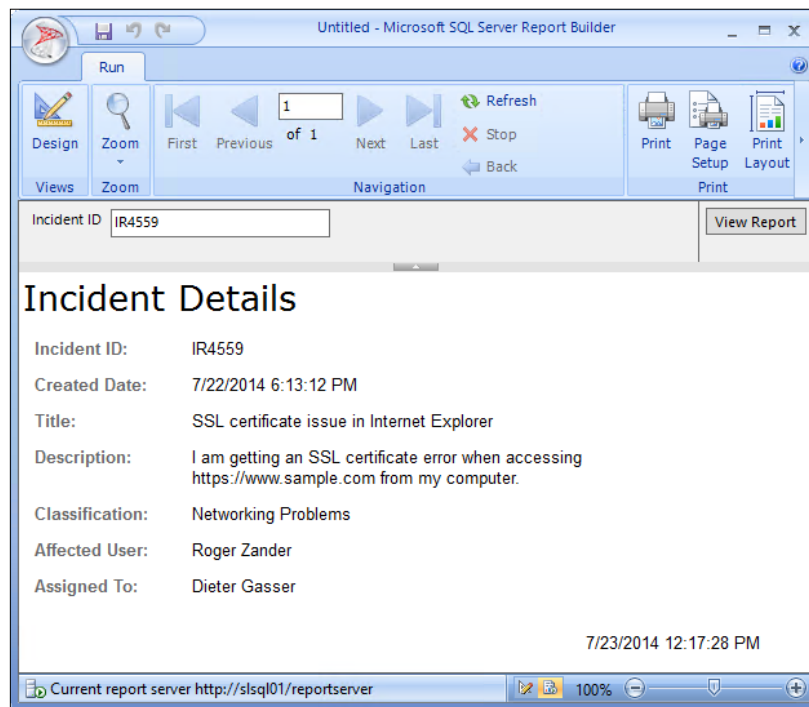
9. When prompted to **Enter Data Source Credentials**, select **Use the current Windows user** or type the password for the service account and click on **OK**.
10. Notice that the **@IncidentID** placeholder we used in the query we just added to our dataset appears under **Parameters** in the **Report Data** area. Report Builder automatically created a corresponding report parameter to allow **incident ID** to pass to our report. Expand the **Parameters** node in Report Builder and make sure the **IncidentID** parameter has been added.
11. In Report Builder, in the **Design** area, replace the `Click to add title` with `Incident Details`.
12. In the toolbar, under **Insert**, click on **Text Box**, and draw a rectangle in the reports body. Enter `Incident ID:` in the text box. Apply bold text and change the font color to gray.
13. In the toolbar, under **Insert**, click on **Text Box** and draw a rectangle in the reports body right next to the right-hand side of the **Incident ID** text box created in the previous step.
14. Drag and drop the **Id** field from the **DSIncidentDetails** dataset into the newly created text box, as shown in the following screenshot:



15. Repeat steps 12 to 14 to create text boxes for the following fields (this is shown in the next screenshot). You can also copy and paste the existing text boxes to speed up this process.
 - ❑ **CreatedDate**
 - ❑ **Title**
 - ❑ **Description**
 - ❑ **Classification**
 - ❑ **AffectedUser**
 - ❑ **AssignedTo**

Incident Details	
Incident ID:	[Id]
Created Date:	[CreatedDate]
Title:	[Title]
Description:	[Description]
Classification:	[Classification]
Affected User:	[AffectedUser]
Assigned To:	[AssignedTo]
[&ExecutionTime]	

16. Click on **Run** from the **Home** toolbar to test your report.
17. Enter a valid incident ID in the IRxxxx format in the **Incident ID** field and then click on **View Report**. Ensure that the report shows valid data for the selected incident, as shown in the following screenshot:



18. Click on **Save** from the **File** menu (or press *Ctrl + S*), browse to a folder in the SSRS of your choice, and save the report with the name `Incident Details`.

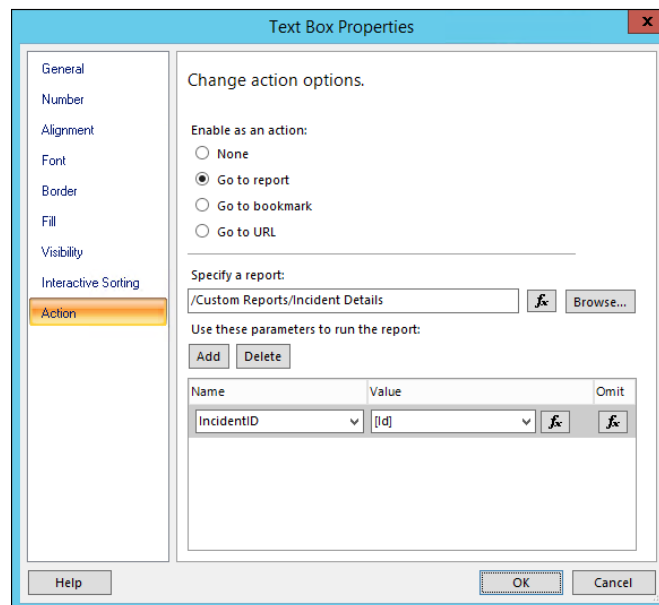
Linking the Open Incidents report to the Incident Details report

We will now change the Open Incidents report created in the previous recipe to allow the report consumer to click an incident ID to display detailed information about the incident. Perform the following steps:

1. Start Report Builder.
2. Under **Open**, browse to the **Open Incidents** report and click on **Open**.
3. Select the text box that holds the **[Id]** placeholder, as shown in the following screenshot:



4. Enable **Underline font style for this text box** (use the formatting tools area) and change the font color to **Blue**.
5. Right-click on the text box and click on **Text Box Properties....**
6. Under **Action**, click on **Go to report**. Then, under **Specify a report**, click on **Browse...** and browse to the **Incident Details** report and click on **Open**. Under **Use this parameters to run the report**, click on **Add**, select the **IncidentID** parameter in the **Name** column, and select **[Id]** in the **Value** column. Click on **OK**, as shown in the following screenshot:



7. Click on **Save** from the **File** menu or press *Ctrl* + *S* to save the changes to your report.
8. Click on **Open** from the **File** menu and open the **Service Management Dashboard** report.
9. Click on **Run** from the **Home** toolbar to test the report.
10. Click on any of the underlined blue incident IDs to open the drillthrough report that shows incident details.

How it works...

A drillthrough report is a report that is opened by clicking on a link within another report. Drillthrough reports commonly have additional details about an item that is in an original summary report. The data in the drillthrough report is not retrieved until the user clicks on the link in the main report that opens the drillthrough report. If the data for the main report and the drillthrough report must be retrieved at the same time, consider using a subreport instead by following the procedures in the previous recipe.

A drillthrough typically contains parameters that are passed to it by the summary report.

The recipe also makes use of formatting to show the report consumer that this is a drillthrough report. This is achieved by underlining and changing the font color to blue (the result is similar to how a web link looks).

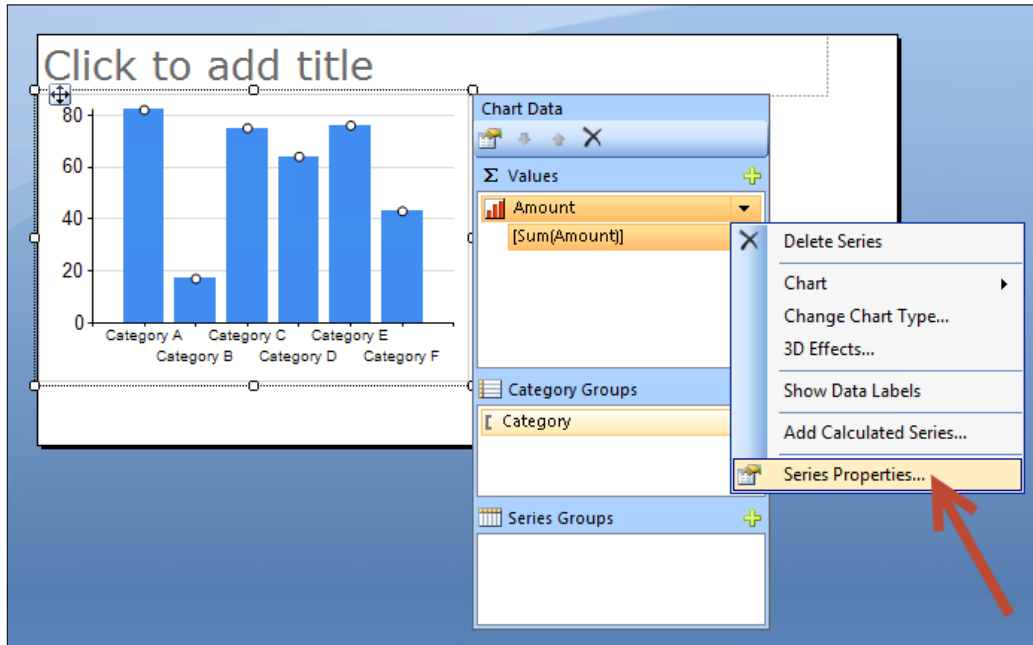
There's more...

The drillthrough feature can be used for other report types, an example is to use it with charts.

Using drillthrough reports in charts

You can also enable charts in reports to allow drilling through to other reports. For instance, you could create a report that shows the number of incidents per support group in a column chart and then enable each individual column to display the incidents of the corresponding support group.

For this to work, you will have to create an **Action** that follows the procedures described earlier in this recipe. Note, however, that you have to create **Action on Series** that you would like to enable for the drillthrough, as shown in the following screenshot:



Working with shared datasets and parameters

In this recipe, we will create a shared dataset for Operations Manager event levels. Shared datasets are datasets stored in SQL Server Reporting Services which can be reused in your reports. Shared datasets are typically used as a source for parameters in your reports. An Operations Manager event level identifies the level of an event, such as **Information**, **Warning**, or **Error**. We will create a sample report that uses the shared dataset to prepopulate the available values for an event-level parameter. Using this report, you will be able to browse events by event level.

Getting ready

We are accessing the Operations Manager event dataset in this recipe. Read through the *Creating event reports* recipe of *Chapter 5, Creating Reports for SCOM and SCVMM*, to get an understanding of the event dataset in the Operations Manager Data Warehouse. You will need access to a fully deployed and functional Operation Manager 2012 SP1/R2 environment with the data warehouse components enabled.

You also need to create a shared data source to work with the `OperationsManagerDW` database. Complete the steps in the *Preparing your environment for authoring reports* recipe in Chapter 5, *Creating Reports for SCOM and SCVMM*.

How to do it...

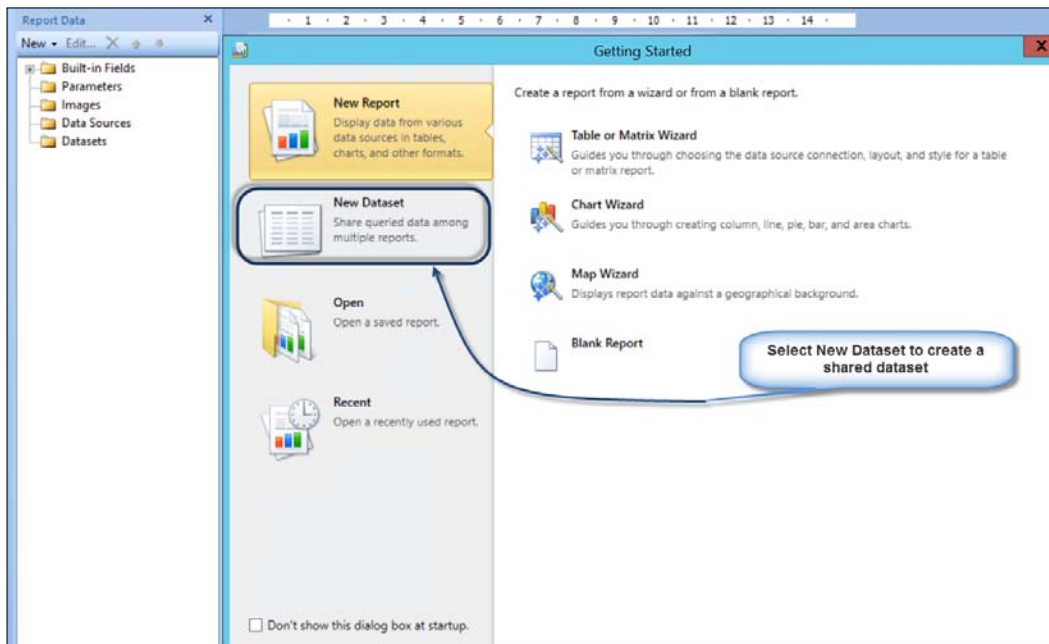
The steps involved in creating the report for this recipe can be grouped into the following two tasks:

- ▶ Creating the Events Levels shared dataset
- ▶ Creating a report that uses the shared dataset to prepopulate a parameter

Creating the Event Levels shared dataset

The Event Levels shared dataset lists all the available event levels from the Operations Manager Data Warehouse. The dataset returns the event level ID and the event level name. Perform the following steps:

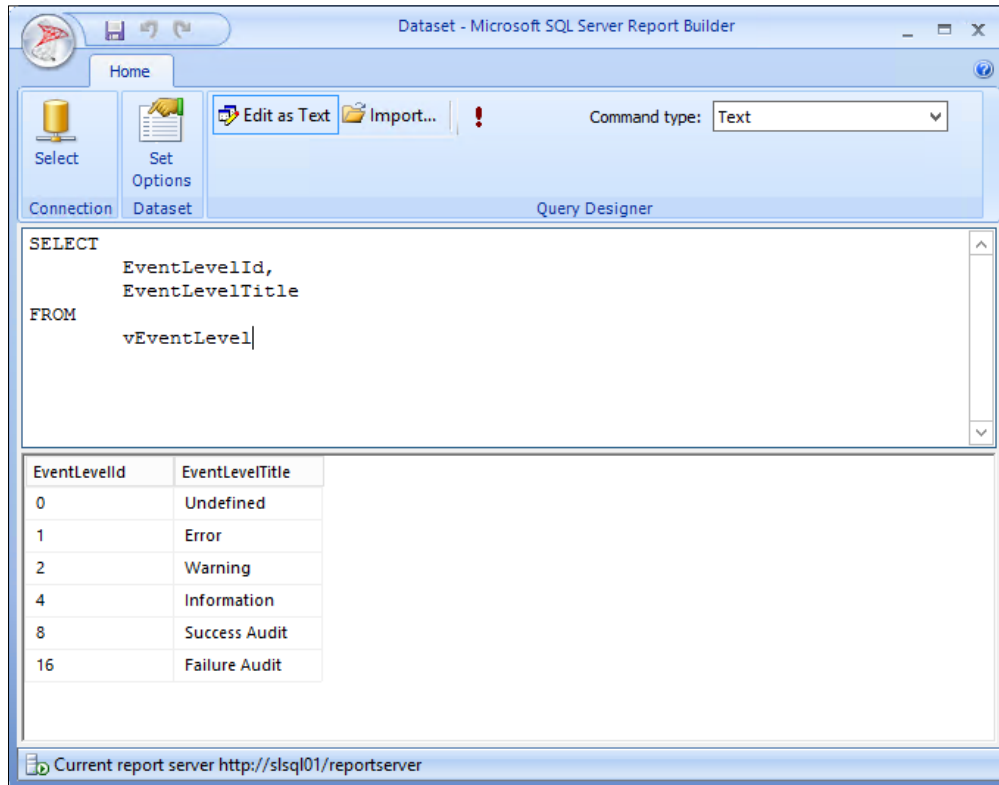
1. Start Report Builder.
2. On the **Getting Started** page, select **New Dataset**, as shown in the following screenshot:



- Under **New Dataset**, select the **OperationsManagerDW** data source. If it is not listed, click on **Browse other data sources...** and browse to the location of the **OperationsManagerDW** data source. Click on **Create**.
- When prompted to **Enter Data Source Credentials**, select **Use the current Windows user** or type the password for the service account and click on **OK**.
- In the **Home** toolbar, click on **Edit as Text** and type the following query:

```
SELECT
    EventLevelId,
    EventLevelTitle
FROM
    vEventLevel
```

- Click on the exclamation mark from the toolbar to run the query, as shown in the following screenshot:



- Click on the **Save** icon (or press **Ctrl + S**), browse to a folder in SSRS of your choice, and save the dataset with the name `Event Levels`.
- Close Report Builder.

Creating the Events Last 30 Days report

We will now create a report that lists all the events of the last 30 days using the data from the Operations Manager data warehouse. We will allow the report consumer to specify an event level using a report parameter. The available values for the event level will be retrieved from the shared dataset created earlier in this recipe. Perform the following steps:

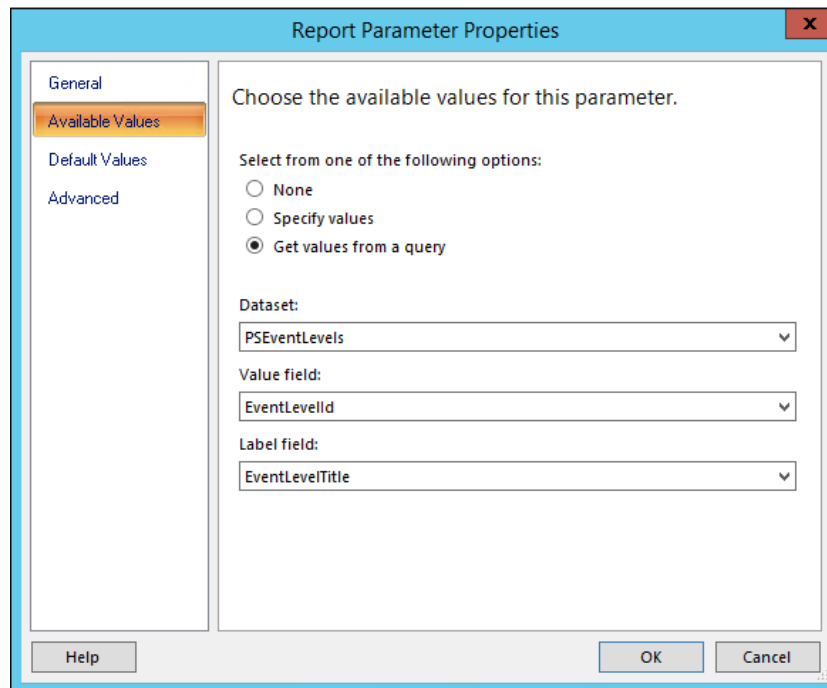
1. Start Report Builder.
2. Under **New Report**, select **Blank Report**.
3. Right-click on **Data Sources** and click on **Add Data Source...**
4. Enter `OperationsManagerDW` as the name.
5. Select **Use a shared connection or report model** and select the **OperationsManagerDW** data source from the list of available data sources. If the data source is not displayed, click on **Browse...** and browse to the location where the data source is stored in SQL Server Reporting Services.
6. Click on **Test Connection** and on success click on **OK**.
7. Right-click on **Datasets** and click on **Add Dataset...**
8. Enter `DSEvents` as the name and select the **Use a dataset embedded in my report** option.
9. Select the `OperationsManagerDW` data source, select the **Text** query type, and type the following query. Then, click on **OK**.

```

SELECT
    E.EventOriginId,
    E.EventNumber,
    E.DateTime,
    E.EventLevelId,
    ELc.ComputerName,
    EPU.EventPublisherName,
    ED.RenderedDescription
FROM
    Event.vEvent E
    LEFT OUTER JOIN vEventLoggingComputer ELc ON
        E.LoggingComputerRowId = ELc.EventLoggingComputerRowId
    LEFT OUTER JOIN vEventPublisher EPU ON
        E.EventPublisherRowId = EPU.EventPublisherRowId
    LEFT OUTER JOIN Event.vEventDetail ED ON
        E.EventOriginId = ED.EventOriginId
WHERE
    E.DateTime >= DATEADD(d, -30, GETUTCDATE())
    AND E.EventLevelId = @EventLevel
ORDER BY
    3 DESC, 6 ASC, 5 ASC

```

10. When prompted to **Enter Data Source Credentials**, select **Use the current Windows user** or type the password for the service account and click on **OK**.
11. Notice the @EventLevel placeholder we used in the query we just added to our dataset. Report Builder automatically created a corresponding report parameter to allow the event level to pass to our report. Expand the **Parameters** node in Report Builder and make sure that the **EventLevel** parameter has been added. As we do not want the report consumer to pass free text in the parameter, we will restrict the available values for this parameter to the event levels returned by the shared dataset created earlier in this recipe.
12. Right-click on **Datasets** and click on **Add Dataset....**
13. Enter PSEventLevels as the name and select the **Use a shared dataset** option. Click on **Browse...** and browse to the location where you saved the **Event Levels** shared dataset. Click on **OK**.
14. In the **Parameters** node, right-click on the **EventLevel** parameter and click on **Parameter Properties**.
15. Under **General**, change **Data type** to **Integer**.
16. Under **Available Values**, select **Get values from a query**. Under **Dataset:**, select the **PSEventLevels** dataset. Under **Value field**, select **EventLevelId**. Under **Label field**, select **EventLevelTitle**. Click on **OK**, as shown in the following screenshot:



17. In Report Builder, in the **Design** area, replace the `Click to add title` with `Events Last 30 Days`.
18. In the toolbar, under **Insert**, click on **Table** and then click on **Table Wizard...**
19. In **New Table or Matrix wizard**, select **Choose an existing dataset in this report or a shared dataset**, select the `DSEvents` dataset, and then click on **Next**.
20. Drag and drop the following fields from the **Available fields** list to the **Values** list, and then click on **Next**:
 - ❑ **DateTime**
 - ❑ **ComputerName**
 - ❑ **EventPublisherName**
 - ❑ **RenderedDescription**
21. Click on **Next** on the **Choose the layout** page, select the desired style, and then click on **Finish**.
22. Change the columns' width and font sizes so that the rendered report is displayed properly.
23. Click on **Run** in the **Home** toolbar to test the report. Under **Event Level**, select an event level and then click on **View Report** to display all the events of this event level, as shown in the following screenshot:

The screenshot shows the Microsoft SQL Server Report Builder interface. The report title is "Events Last 30 Days". The report is displayed in a table with the following data:

Date Time	Computer Name	Event Publisher Name	Rendered Description
1/18/2015	SEUKSCOM01.FW2RW.COM	Health Service Script	LogEndToEndEvent.js : This event is logged to the Windows Event Log periodically to test a event collection.
1/18/2015	SEUKSCOM01.FW2RW.COM	Health Service Script	LogEndToEndEvent.js : This event is logged to the Windows Event Log periodically to test a event collection.
1/18/2015	SEUKSCOM01.FW2RW.COM	Health Service Script	LogEndToEndEvent.js : This event is logged to the Windows Event Log periodically to test a event collection.
1/18/2015	SEUKSCOM01.FW2RW.COM	Health Service Script	LogEndToEndEvent.js : This event is logged to the Windows Event Log periodically to test a event collection.
1/18/2015	SEUKSCOM01.FW2RW.COM	Health Service Script	LogEndToEndEvent.js : This event is logged to the Windows Event Log periodically to test a event collection.
1/18/2015	SEUKSCOM01.FW2RW.COM	Health Service Script	LogEndToEndEvent.js : This event is logged to the Windows Event Log periodically to test a event collection.
1/18/2015	SEUKSCOM01.FW2RW.COM	Health Service Script	LogEndToEndEvent.js : This event is logged to the Windows Event Log periodically to test a event collection.
1/18/2015	SEUKSCOM01.FW2RW.COM	Health Service Script	LogEndToEndEvent.js : This event is logged to the Windows Event Log periodically to test a event collection.
1/18/2015	SEUKSCOM01.FW2RW.COM	Health Service Script	LogEndToEndEvent.js : This event is logged to the Windows Event Log periodically to test a event collection.
1/18/2015	SEUKSCOM01.FW2RW.COM	Health Service Script	LogEndToEndEvent.js : This event is logged to the Windows Event Log periodically to test a event collection.

24. Change **Event Level** to a different value and click on **View Report** to refresh the report and display events of another event level.
25. Click on **Save** from the **File** menu (or press *Ctrl + S*), browse to a folder in SSRS of your choice, and save the report with the name `Events Last 30 Days`.

How it works...

The recipe is split into two parts. How each part works is as follows:

- ▶ **Creating the Event Levels shared dataset:** This part of the recipe provides the steps to create a dataset as a standalone entity that can be reused in multiple reports. The common element when using a shared dataset is the data source target. When you select **New Dataset** from the wizard, Report Builder switches to a shared dataset mode. Once you have created the dataset, selecting the **Save** option allows you to create a shared dataset.
- ▶ **Creating the Events Last 30 Days report:** The second part of the recipe provides steps to create a report that leverages the shared dataset you created in the first part. This report uses a variable that is fed from a SQL query (dataset). When you define a parameter using the `@<parameter>` name, you get the option to create a manual list of valid values or, in this case, a dynamic list that is generated by a SQL query to return the current data at the time of execution.

The dataset definition is only stored once in SQL Server Reporting Services rather than inside each individual report. This allows you to make a modification in one place and propagate the change to all the dependent reports.

Shared datasets are commonly used for report parameters. This allows you to have the same parameters using one shared dataset in several reports.

There's more...

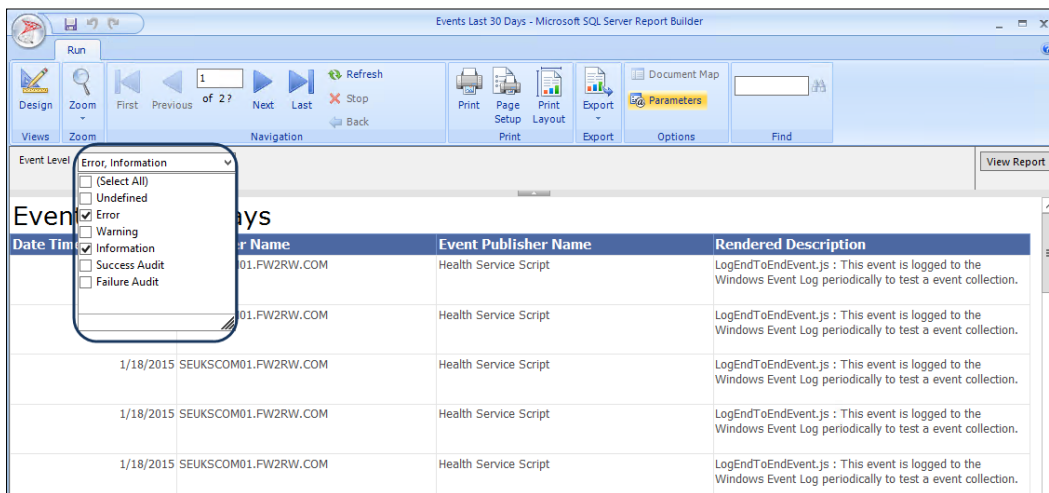
Instead of having to select a single value from a list of report parameter, parameters that allow the selection of multiple values can also be defined.

Allow multiple values to be selected for a parameter

You can configure a report parameter to allow multiple values to be selected by the report consumer by performing the following steps:

1. In Report Builder, open the **Events Last 30 Days** report and make sure that you are in the **Design** mode.
2. In the **Parameters** node, right-click on the **EventLevel** parameter and click on **Parameter Properties**.

- Under **General**, enable the **Allow multiple values** option. Click on **OK**. SQL Server Reporting Services will pass all the selected values as a string of comma-separated values to the query of your dataset. This means that we need to change the query of our `DSEvents` dataset to allow for this string to be interpreted correctly.
- In the **Datasets** node, right-click on the **DSEvents** dataset and click on **Dataset Properties**.
- In the query, replace the text `E.EventLevelId = @EventLevel` with `E.EventLevelId IN (@EventLevel)`.
- When prompted to **Enter Data Source Credentials**, select **Use the current Windows user** or type the password for the service account and click on **OK**.
- Click on **OK**. Click on **Run** from the **Home** toolbar to test the report, as shown in the following screenshot:



Analyzing Chargeback reports using System Center 2012 R2 data

This recipe provides the steps to report on (or analyze) Chargeback-related reports using the data from System Center Virtual Machine Manager, Operations Manager, and Service Manager.

Getting ready

Chargeback reporting leverages various components of System Center 2012 R2. These components are as follows:

- ▶ **System Center 2012 R2 Virtual Machine Manager:** This is used to create an abstraction layer between your hardware resources (fabric) and your end users. This abstraction layer is called **Clouds** in VMM.
- ▶ **System Center 2012 R2 Operations Manager:** This is used to monitor the usage and capacity of resources managed by VMM. These resources include your fabric, virtual machines, and private clouds.
- ▶ **System Center 2012 R2 Service Manager:** This is the central component that holds all the data about your VMM clouds. You can define price sheets in Service Manager and associate them with your clouds.
- ▶ **System Center 2012 R2 Service Manager Data Warehouse:** This is used to create and view Chargeback reports.

For Chargeback to work, you need to configure all of the preceding components in accordance with the following instructions.

Virtual Machine Manager

Before you can start configuring Virtual Machine Manager to manage clouds, you must have a Virtual Machine Manager Server installed, and this server must be managing at least one Hyper-V host server. There should be a couple of virtual machines running in the environment managed by VMM.

For the integration between VMM and Operations Manager to work, you must install the Operations Manager Console on the Virtual Machine Manager Server. You must also install an Operations Manager Agent on the Virtual Machine Manager Server so that monitoring data can be obtained from Operations Manager.

Once these requirements are met, you can start configuring Virtual Machine Manager to manage clouds and configure the integration between Virtual Machine Manager and Operations Manager.

Creating clouds in VMM

The steps to create clouds involve administrative configuration in Virtual Machine Manager. If you are running VMM in a production environment, you are likely to already have clouds configured in VMM. If this is the case, you can skip this section and continue with the next section, *Provisioning Virtual Machines to clouds*.

If you are not familiar with administering VMM or if you are configuring Chargeback reporting for a production environment, it is highly recommended that you work on the following steps with your virtualization administrator:

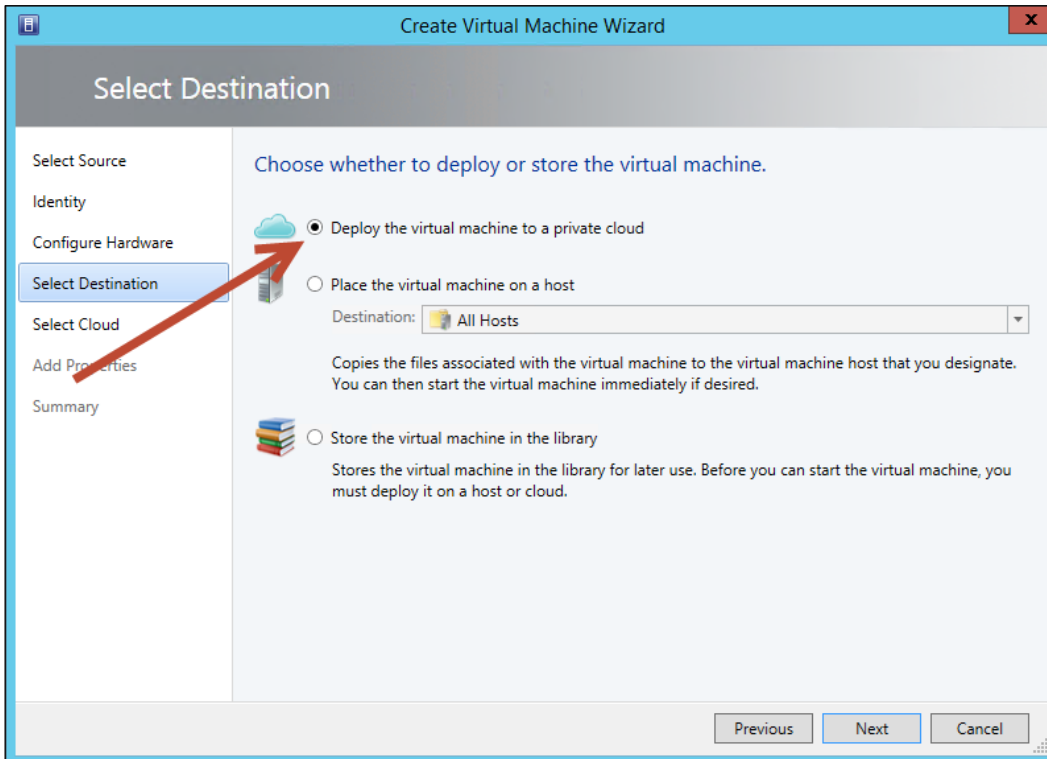
1. Open **VMM Console** and connect to your Virtual Machine Manager server using an account that has administrative access to VMM.
2. In the **VMs and Services** area, in the **Home** ribbon, click on **Create Cloud**.
3. Give the cloud a name and click on **Next**.
4. Select **Host Group** that should host the VMs that you provision in this cloud, and then click on **Next**.
5. Select the logical networks that should be accessible from your cloud and click on **Next**.
6. If required, configure options for **Load Balancers, VIP Templates, Port Classifications, Storage, and Library**.
7. Under **Capacity**, define the capacity limits for your cloud or select **Use Maximum**. Click on **Next**.
8. Select **Hyper-V** and click on **Next**.
9. Click on **Finish** to create your cloud.

You can create more clouds as required for your environment. The price sheets that we will create later in this recipe are per cloud, so you can define different pricing models for different clouds in your environment.

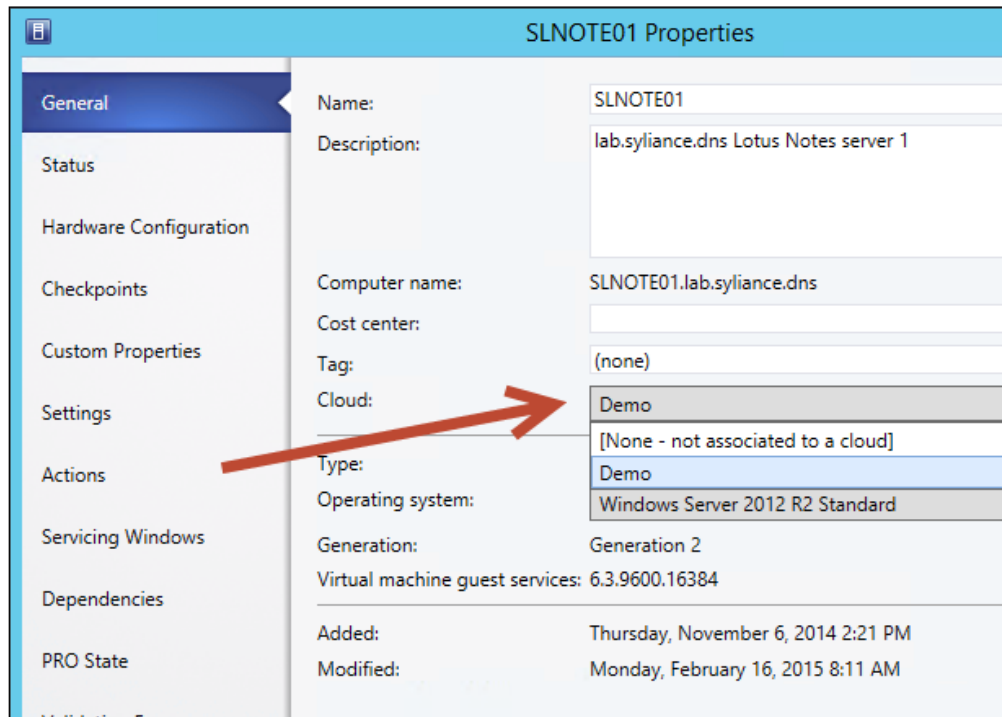
Provisioning virtual machines to clouds

Next, we need to have virtual machines assigned to our cloud(s) to calculate actual usage. A virtual machine can be assigned to a cloud when it is provisioned or after provisioning.

To assign a virtual machine to a cloud during the provisioning process, in **Create Virtual Machine Wizard**, under **Select Destination**, select the **Deploy the virtual machine to a private cloud** option. In the next step, **Select Cloud**, select the desired cloud that will host the VM, as shown in the following screenshot:




To assign an already existing virtual machine to a cloud, right-click on the virtual machine, click on **Properties**, and choose the desired cloud under the **Cloud** node, as shown in the following screenshot:



Operations Manager

Chargeback requires a set of management packs to be imported into Operations Manager. There is a script available to facilitate the import of these required management packs:

1. In the Service Manager Management Server, locate the **Chargeback** folder in the Service Manager installation directory.
2. Copy the **Dependencies** folder to a temporary folder on your Operations Manager Management Server.
3. In the Operations Manager Management Server, run the `ImportToOM.ps1` script and wait for it to complete.

 To enable PowerShell to execute scripts, run the `Set-ExecutionPolicy -force RemoteSigned` command.

Next, you need to make sure that the VMM and Operations Manager service accounts have administrative rights on both systems. First, on your VMM server, verify that the **System Center Virtual Machine Manager** service is running under a domain account. Then, add this domain account to Operations Manager Administrators. Next, on your Operations Manager server, verify that the **System Center Data Access** service is running under a domain account. Then, add this domain account to the VMM Administrators.

Furthermore, we must configure the integration between Operations Manager and Virtual Machine Manager. This process is described in the *Using Virtual Machine Manager reports* recipe in *Chapter 5, Creating Reports for SCOM and SCVMM*.


Service Manager

Now that VMM and Operations Manager are configured and exchanging data, we will configure Service Manager to retrieve data from Operations Manager.

Make sure that your Service Manager management server and your Service Manager Data Warehouse management server are installed and that the Data Warehouse management server has been registered with Service Manager.

Importing the Chargeback management pack

The Chargeback management packs include classes and data warehouse objects required to leverage Chargeback in System Center Service Manager.

 If you have the Provance IT Asset Management Pack installed in your Service Manager environment, you will not be able to import the Chargeback management packs into Service Manager, because Provance IT Asset Management Pack introduced a dimension in the data warehouse named `CostCenterDim`, which is also used in Chargeback. This issue is described at <http://blogs.technet.com/b/servicemanager/archive/2013/03/21/incompatibility-between-provance-and-chargeback-management-pack-costcenterdim.aspx>. Although Microsoft has promised a fix that will resolve this issue, this fix is not yet available at the time of writing this book.

Perform the following steps:

1. On the Service Manager Management server, start **Service Manager Shell** as an administrator.
2. Change to the **Chargeback** folder in the Service Manager installation directory.
3. Run the `ImportToSM.ps1` script.



To enable PowerShell to execute scripts, run the `Set-ExecutionPolicy - force RemoteSigned` command.

4. When prompted for **omServer**, specify the FQDN of the Operations Manager server.
5. When prompted for **omUsername**, specify a user account with administrative access to Operations Manager.
6. When prompted for **omPassword**, specify the password of the user.
7. When prompted for **smUsername**, specify a user account with administrative access to Service Manager.
8. When prompted for **smPassword**, specify the password of the user.
9. Wait for the script to complete.

Configuring the SCOM Connector

We are now creating a SCOM Configuration Item Connector to import data from Operations Manager to Service Manager.



If you have already configured an Operations Manager CI Connector, then open the properties of the connector, go to **Management Packs**, click on **Refresh**, and make sure that the **Microsoft.SystemCenter.VirtualMachineManager.2012.Discovery** and the **Microsoft.SystemCenter.VirtualMachineManager.Pro.2008.Library** management packs are selected. You can then skip the following instructions after refreshing and selecting the preceding management packs in an environment where the SCOM CI connector is already configured.

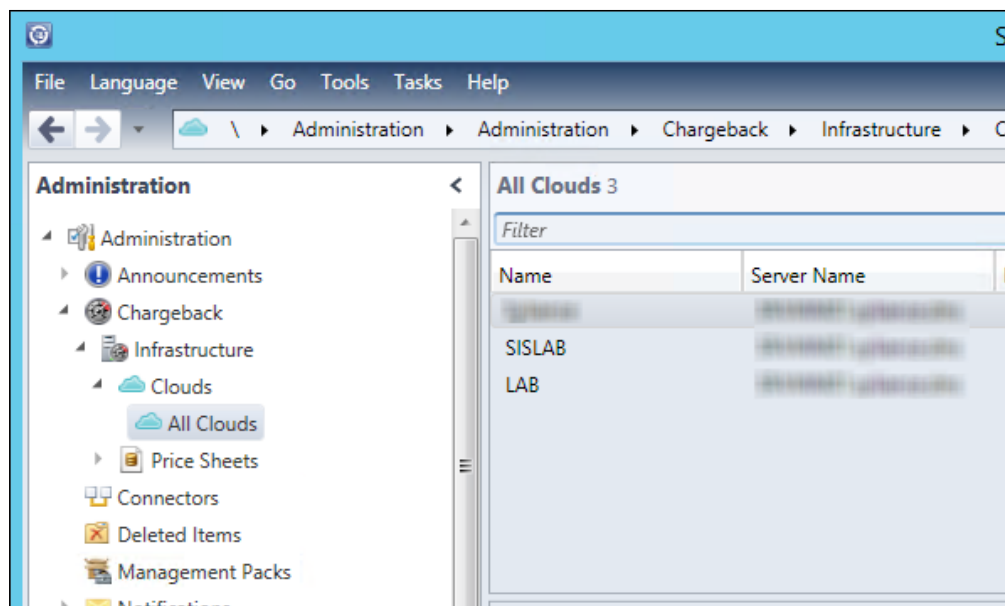
Perform the following steps:

1. Open the **Service Manager** console as a Service Manager administrator.
2. Under **Administration, Connectors**, on the **Tasks** pane, click on **Create connector** and then click on **Operations Manager CI connector**.
3. Click on **Next**, enter a name for the connector, and click on **Next** again.
4. Under **Server name**, enter the FQDN of the Operations Manager server.
5. Under **Credentials**, click on **Run As account** to run an existing account or create a new **Run As account** that has administrative access to Operations Manager. Click on **Next**.
6. In the list of management packs, select all management packs and verify that none are gray. Ensure that the following two management packs are listed and selected:
 - ❑ **Microsoft.SystemCenter.VirtualMachineManager.2012.Discovery**
 - ❑ **Microsoft.SystemCenter.VirtualMachineManager.Pro.2008.Library**



If any of the management packs are gray and cannot be selected, make sure that you check whether the management pack in Operations Manager and Service Manager are the same version. In some cases, the Operations Manager management pack might be newer than the management pack in Service Manager. If this is the case, download the latest management pack from the Operations Manager catalog and select **Download to disk**. Next, copy this file to the Service Manager server and import the management pack using the Service Manager console.

7. Click on **Next**, select a **Schedule**, click on **Next** again, verify the information in the summary, and then click on **Create**.
8. Wait for 5 minutes and select **Refresh** in the **Tasks** menu. Repeat this step until **Finished Success** is displayed in the **Status** column for the newly created connector.
9. Under **Administration**, expand **Chargeback, Infrastructure**, and **Clouds**, and then click on **All Clouds**.
10. Verify that the clouds configured in VMM show up in the list, as shown in the following screenshot:



11. Under **Data Warehouse**, click on **Data Warehouse Jobs**, click on the **MPSyncJob** job, and click on **Resume** from the **Tasks** pane.
12. Wait for the job to finish, and then verify that all the management packs are associated in the details pane of the **MPSyncJob**. In particular, make sure that the **Microsoft.SystemCenter.VirtualMachineManager.2012.Discovery** management pack is associated.

How to do it...

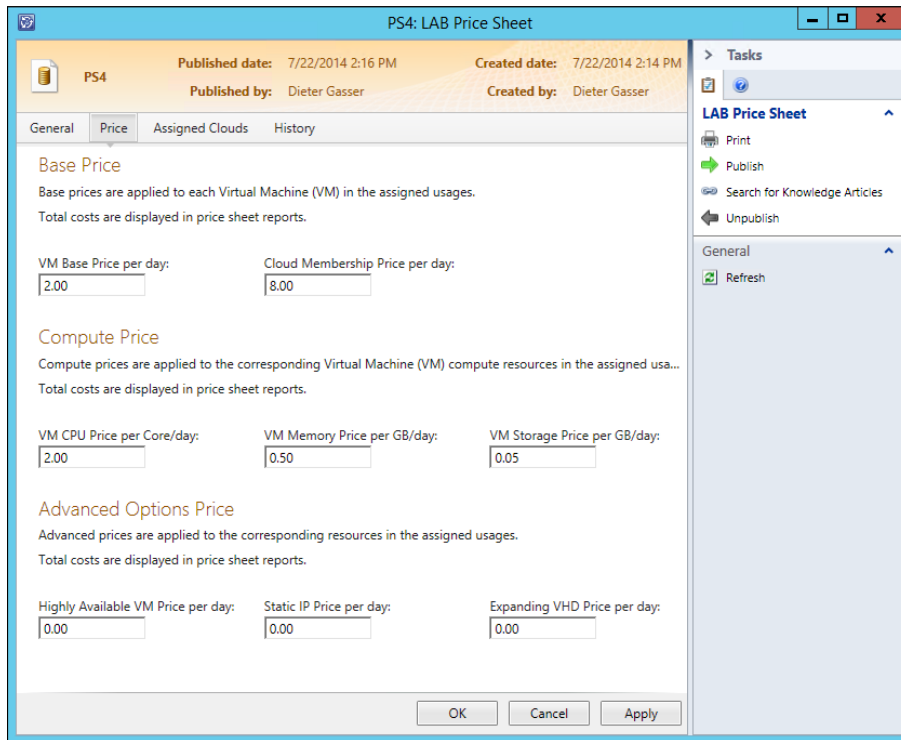
Follow these steps to prepare your cloud environment to analyze Chargeback data.

Creating price sheets

Follow these steps to create the Chargeback price sheets:

1. Start the Service Manager console as a user with administrative access to Service Manager.
2. Under **Administration**, expand **Chargeback**, **Infrastructure**, and **Price Sheets**, and then click on **All Price Sheets**.
3. Click on **Create Price Sheet** from the **Tasks** pane.
4. Enter a name for the price sheet and switch to the **Price** tab.

5. Enter pricing information for your cloud(s). An example is given in the following screenshot:



6. Click on **Publish** in the **Tasks** pane.
7. Switch to the **Assigned Clouds** tab.
8. Add at least one cloud using the **Add...** button.
9. Click on **OK** to save the price sheet.

Analyzing Chargeback data

There are several tools available for analyzing Chargeback data; some of them are as follows:

- ▶ **Microsoft Excel**
- ▶ **Microsoft Excel PowerPivot**
- ▶ **Microsoft SharePoint PerformancePoint**
- ▶ **Microsoft SQL Server Reporting Services**

In this example, we will use Microsoft Excel to access the cube and create an ad hoc PivotTable report to display Chargeback information.



Make sure that Microsoft Excel is installed on the computer that runs the Service Manager console.

Perform the following steps:

1. Start the Service Manager console as a user with administrative access to Service Manager.
2. Under **Data Warehouse**, click on **Cubes** and select **Service Manager Chargeback Cube**. Then, click on **Analyze Cube in Excel** in the **Tasks** pane.
3. Excel will now open, and a connection to the Chargeback cube is automatically established. You will see an empty PivotTable.
4. In the **PivotTable Fields** area, in the **ServiceManagerInfraChrgbackDateDim** group, select the **Fiscal Year Hierarchy** dimension.
5. In the **VirtualMachineDim** group, select **Cloud** and **Computer Name**.
6. In **Service Manager Infra Daily Chargeback**, select **Cloud Cost**, **VM Cost**, and **VM Total Cost**.
7. Expand and collapse dimensions in the PivotTable as desired. Your PivotTable should now look similar to the following screenshot:

Row Labels	Cloud Cost	VM Cost	VM Total Cost
2014	0.00	0.00	0.00
2015	64.00	90.50	154.50
Q1	64.00	90.50	154.50
LAB	64.00	90.50	154.50
SLAPP01	16.00	20.00	36.00
slscsm01	16.00	33.50	49.50
SLSCSM0	16.00	20.00	36.00
SLTEST01	16.00	17.00	33.00
Grand Total	64.00	90.50	154.50

How it works...

The recipe provides steps in the *Getting ready* section on how to prepare the environment for Chargeback data analysis (reporting). Once you complete the *Getting ready* steps, you are provided with steps to create price sheets, and when you create the price sheets, you perform the final steps on analyzing the data. The breakdown of the two parts follows:

- ▶ **Creating price sheets:** The steps in this section walk you through importing the required management packs and creating sample pricing data
- ▶ **Analyzing Chargeback data:** The steps in this final part provide information on using Microsoft Excel to access the cubes created by the Chargeback management pack import. You must wait for the ETL process to complete before you can perform these steps.

We need to create at least one price sheet, publish it, and assign it to one or more clouds before the actual Chargeback data is available to be analyzed.

Service Manager transfers the Chargeback data to the Service Manager Data Warehouse and processes an Analysis Service cube that you can access to analyze data.



Depending on your environment, it can take several hours after the price sheet is created until all data has been transferred to the data warehouse and the cube is processed.

9

Using Power BI to Analyze and Visualize System Center Data

In this chapter, we will cover the following recipes:

- ▶ Configuring Microsoft Excel for System Center data analysis
- ▶ Connecting to System Center data sources with Power Query
- ▶ Creating a data model with PowerPivot
- ▶ Visualizing the analysis with Power View and Power Maps
- ▶ Using Microsoft Cloud services to share and visualize System Center data

Introduction

This chapter focuses on the creation and visualization of System Center datasets using Microsoft Excel and Microsoft Power BI services. You create the model and then visualize and analyze the data using the add-ins enabled or installed in Microsoft Excel. The four add-ins that extend Microsoft Excel for a self-service BI are as follows:

- ▶ **Power Query:** Microsoft Power Query for Excel provides an intuitive and consistent experience to discover, combine, and refine data across a wide variety of data sources. Data gathering and modeling that is used to match your analysis requirements is done using Power Query. You can use Power Query to create queries in preparation for further analysis and modeling by tools such as PowerPivot, Power View, and Power Maps.

- ▶ **PowerPivot:** With PowerPivot, you can create relationships between heterogeneous data, create calculated columns and measures, and build PivotTables and PivotCharts for further analysis. You can transform large datasets into meaningful information to get the answers you need in seconds.
- ▶ **Power View:** Power View makes it possible to create interactive data explorations and visualizations using Excel sheets for intuitive ad-hoc reporting. This add-in consumes the data model created in PowerPivot.
- ▶ **Power Maps:** Power Map makes it possible to explore and navigate geospatial data through a three-dimensional (3D) map experience in Microsoft Excel.

Creating reports (or dashboards) using these Excel add-ins can be compared with the classic **Microsoft SQL Server Reporting Services (SSRS)** report creation. Data gathering and modeling is done with Power Query and PowerPivot. Visualization of the analysis is made available through Power View and Power Maps. The difference between using Power BI tools and SSRS report creation is the type of knowledge and skills that each requires. SSRS can be used to create interactive dashboards and ad-hoc reports but requires a higher level of skill than BI add-ins for Microsoft Excel.

The recipes in this chapter provide the foundational steps required to visualize the results of System Center data analysis using these add-ins.

Configuring Microsoft Excel for System Center data analysis

This recipe is focused on preparing your environment for self-service **Business Intelligence (BI)**.

Getting ready

Self-service BI in Microsoft Excel is made available by enabling or installing add-ins. You must download the add-ins from their respective official sites:

- ▶ **Power Query:** Download Microsoft Power Query for Excel from <http://www.microsoft.com/en-gb/download/details.aspx?id=39379>.
- ▶ **PowerPivot:** PowerPivot is available in the Office Professional Plus and Office 365 Professional Plus editions, and in the standalone edition of Excel 2013.
- ▶ **Power View:** Power View is also available in the Office Professional Plus and Office 365 Professional Plus editions, and in the standalone edition of Excel 2013.
- ▶ **Power Maps:** At the time of writing this book, this add-in can be downloaded from the Microsoft website. Power Map Preview for Excel 2013 can be downloaded from <http://www.microsoft.com/en-us/download/details.aspx?id=38395>.

How to do it...

The tasks discussed in this recipe are as follows:

- ▶ Installing the Power Query add-in
- ▶ Installing the Power Maps add-in
- ▶ Enabling PowerPivot and Power View in Microsoft Excel

Installing the Power Query add-in

The Power Query add-in must be installed using an MSI installer package that is available at **Microsoft Download Center**. The installer deploys the bits and enables the add-in in your Excel installation.

The functionality in this add-in is regularly improved by Microsoft. Search for Microsoft Power Query for Excel in Download Center for the latest version. The add-in can be downloaded for 32-bit and 64-bit Microsoft Excel versions.

Follow these steps to install the Power Query add-in:

1. Review the system requirements on the download page and update your system if required. Note that when you initiate the setup, you may be prompted to install additional components if you do not have all the requirements installed.
2. Right-click on the MSI installer and click on **Install**.
3. Click on **Next** on the **Welcome** page.
4. Accept the **License Agreement** and click on **Next**.
5. Accept the default or click on **Change** to select the destination installation folder.
6. Click on **Next**. On the **Ready to Install Microsoft Power Query for Excel** page, click on **Install**.
7. The installation progress is displayed. Click on **Finish** on the **Installation Completed** page.
8. The **Power Query** tab is available on the Excel ribbon after this installation.

Installing the Power Map add-in

The Power Map add-in must be installed using an executable (.exe) installer package that is available at Microsoft Download Center.

The functionality in this add-in also is regularly improved by Microsoft. Search for Microsoft Power Map for Excel in the Download Center for the latest version.

Follow these steps to install the Power Map add-in:

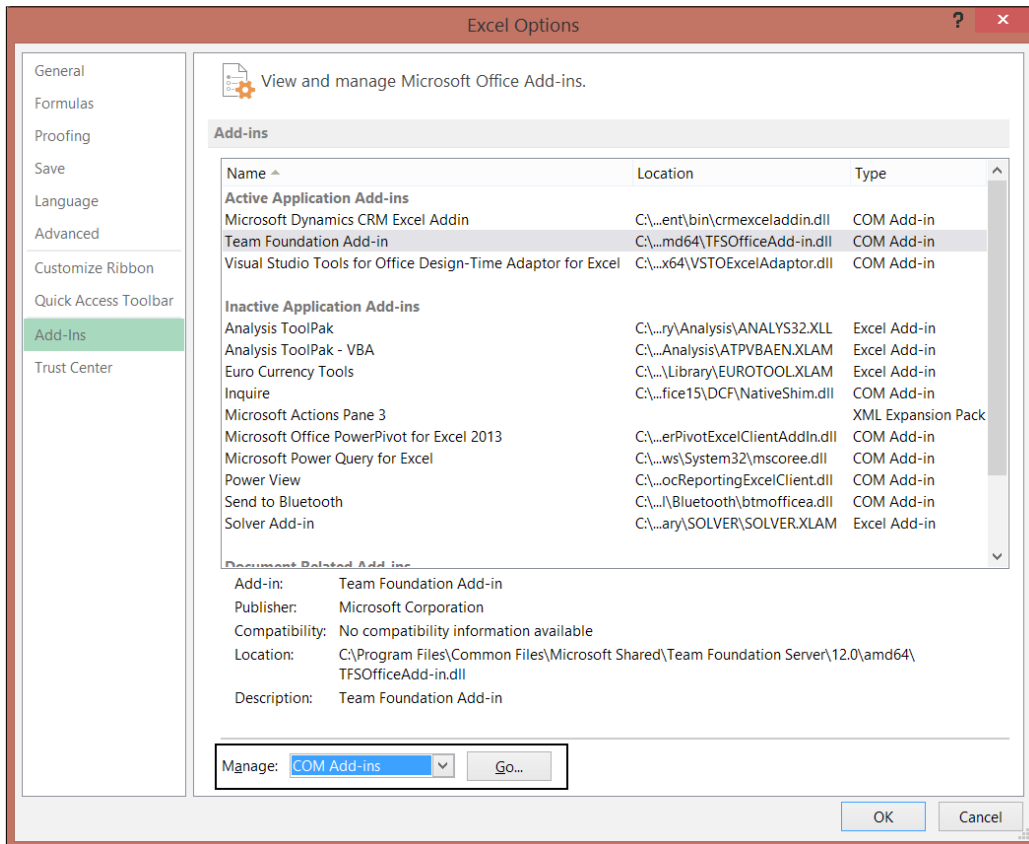
1. Review the system requirements on the download page and update your system if required.
2. Double-click on the EXE installer (Microsoft Power Map Preview for Excel) and click on **Yes** if you get the **User Access Control** dialog prompt.
3. When prompted to install **Visual C++ 2013 Runtime Libraries (x86)**, click on **Close** under **Install**. Check to agree to the terms and click on **Install**.
4. Click on **Next** on the **Welcome** page.
5. Click on the **I Agree** radio button on the **License Agreement** page, and then click on **Next**.
6. Accept the default folder or click on **Browse** to select a different destination installation folder. Make your selection on who the installation should be made available to: **Everyone** or **Just me**. Click on **Next**.
7. Click on **Next**. On the **Confirm Installation** page, click on **Next**.
8. The installation progress is displayed. Click on **Close** on the **Installation Completed** page.
9. The **Power Map** task will be made available in the **Insert** tab on the Excel ribbon after this installation.

Enabling PowerPivot and Power View in Microsoft Excel

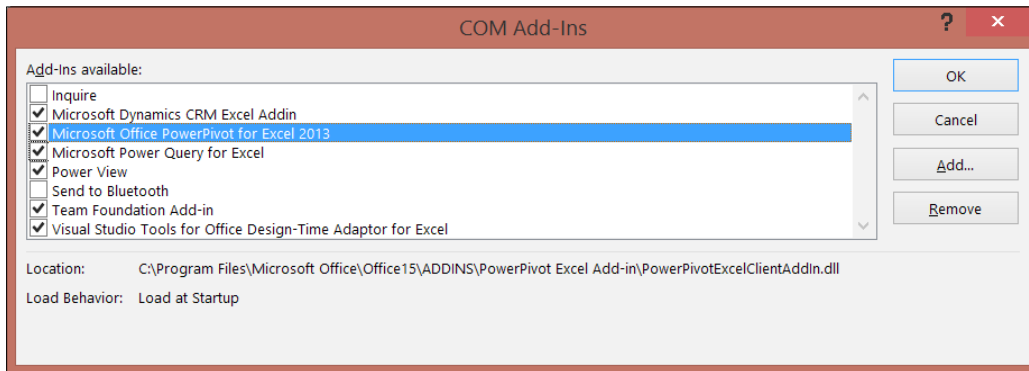
Perform the following steps in Microsoft Excel to enable PowerPivot and Power View:

1. In the **File** menu, select **Options**.

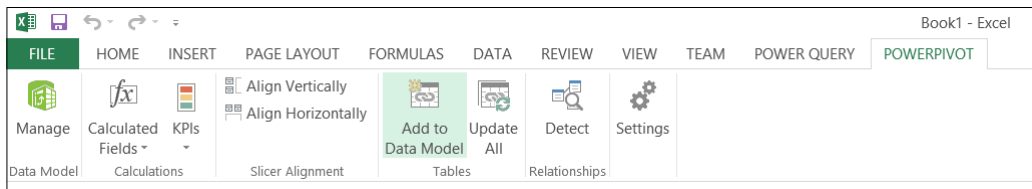
- In the **Add-Ins** tab, select **COM Add-Ins** from the **Manage:** dropdown at the bottom and click on the **Go...** button, as shown in this screenshot:



3. Select the Power add-ins from the list of **Add-Ins available**, as shown in the following screenshot:



4. Click on **OK** to complete the procedure of enabling add-ins in Microsoft Excel. After you've enabled the required add-ins, the different types of add-in tasks and tabs should be available on the Excel ribbon, as shown in this screenshot:



5. This procedure can be used to enable or disable all the available Excel add-ins.

You are now ready to explore System Center data, create queries, and perform analysis on the data.

How it works...

The add-ins for Microsoft Excel provide additional functionality to gather and analyze System Center data. Wizards can be added, interfaces can be made available to combine different sources, and a common language, **Data Analysis SyntaX (DAX)**, can be made available for calculations and performing different forms of visualizations.

The steps discussed in this recipe are required for the use of the Power BI features and functionality using Microsoft Excel. You followed the steps to install Power Query and Power Map, and you enabled PowerPivot and Power Views. These add-ins provide the foundation for self-service Business Intelligence using Microsoft Excel.

See also

- ▶ Different types (enhanced) of functionality and integrations are available for you when you use Microsoft SQL Server or SharePoint, which are not discussed in this chapter. Refer to <http://office.microsoft.com> for additional information on them.

Connecting to System Center data sources with Power Query

Using the Power Query add-in, you can connect to data from different sources. SQL Server databases, Azure SQL databases, OData feed, different file formats, and many others are available as data sources in the **POWER QUERY** tab on the Excel ribbon. Once connected, you can merge functions, integrate them, and perform other manipulations on data from these sources to meet your analytical requirements.

Getting ready

Chapters 4 to 8 of this book provide information about data sources for reporting. You can use the same sources as the input for your Power BI queries.

You must have a fully deployed and functioning System Center 2012/R2 Service Manager Data Warehouse to complete this recipe. The following functionality in Power Query for Microsoft Excel 2013 is described in these recipe sections:

- ▶ Connecting to System Center data sources
- ▶ Exploring the data and overviewing the interface
- ▶ Extending the query data
- ▶ Adding date and time functions to Power Query

The following table lists what is required to complete this recipe:

Recipe requirement or information	Description
System Center data source connections in Microsoft Excel	Service Manager's DWDataMart database is the System Center data source for this recipe
In the <i>There's more</i> section, refer to the additional data source example	Orchestrator database
Microsoft Excel	Version 2013

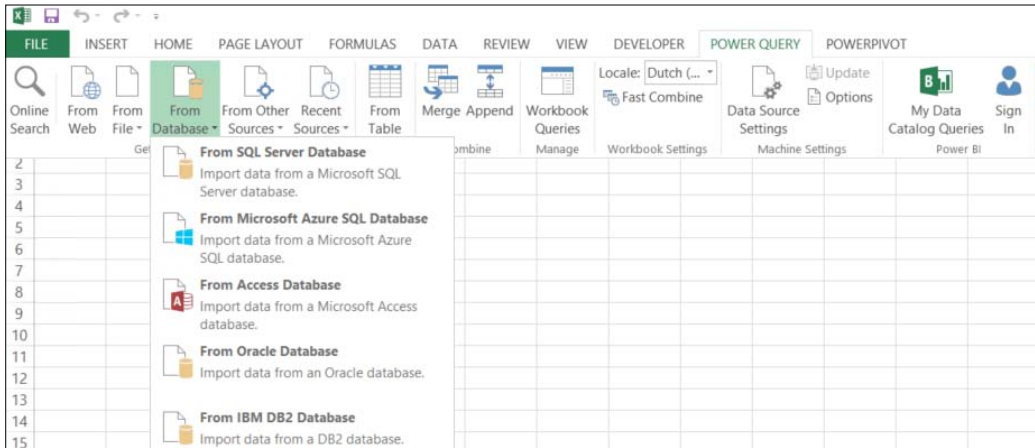
How to do it...

Now let's look at how we can connect to System Center data sources with Power Query.

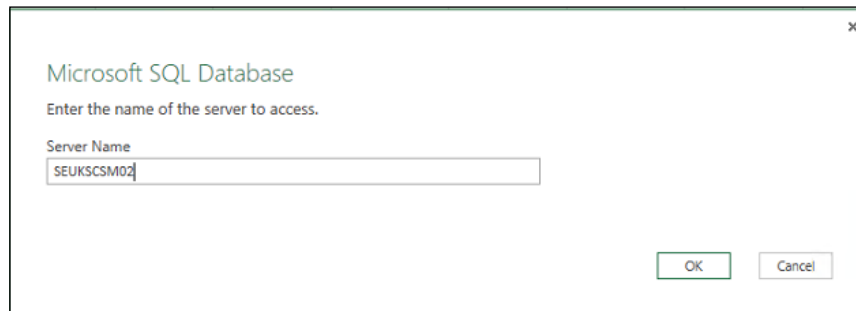
Connecting to System Center data sources

Perform these steps to connect to a System Center data source:

1. Launch Microsoft Excel, navigate to the **POWER QUERY** tab and to data source settings, and click on **From Database**.
2. From the dropdown, select **From SQL Server Database** as shown in the following screenshot:



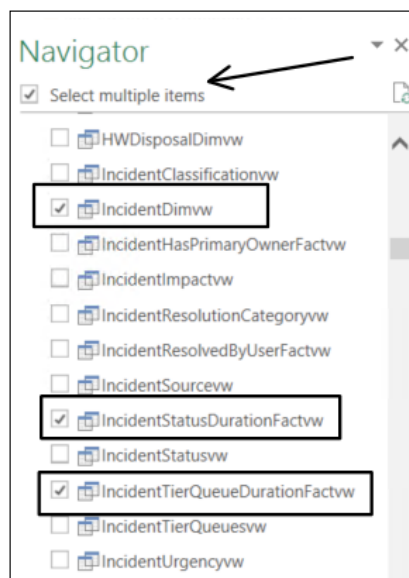
3. In the **Microsoft SQL Database** pop-up window, specify the SQL Server name and database. For this recipe, use the Service Manager data warehouse database as shown in the following screenshot:



4. Click on **OK**.
5. From the **Access a Microsoft SQL Database** page, select the authentication preference and click on **Save**. If prompted for **Encryption Support**, click on **OK**.

The **Data Source** navigation pane is displayed on the right-hand side of your Excel worksheet after the successful connection to the data source.

6. Under the **Navigator**, check **Select multiple items** and select the following views from **DWDDataMart: DisplayStringDimVW**, **IncidentDimvw**, **IncidentStatusDurationFactvw**, and **IncidentTierQueueDurationFactvw**, as shown in the following screenshot:



7. Click on the little arrow next to the **Load** button at the bottom-left corner of the **Navigator** pane, and select **Load To...**
8. In the **Load To** page, select **Only Create Connection** and ensure that **Add this data to the Data Model** is selected. Then, click on **Load**.

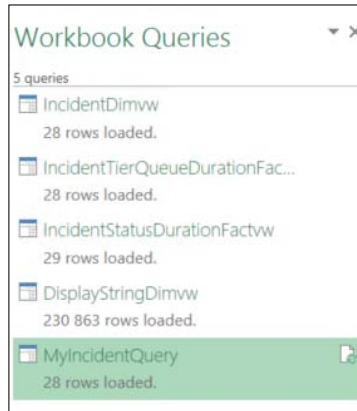


There is no need to load the complete dataset in your Excel sheet when creating the query.

The selected SQL database views are evaluated and loaded into the data model.

9. In the **Workbook Queries** area, right-click on **IncidentDimvw** and select **Duplicate** from the menu.

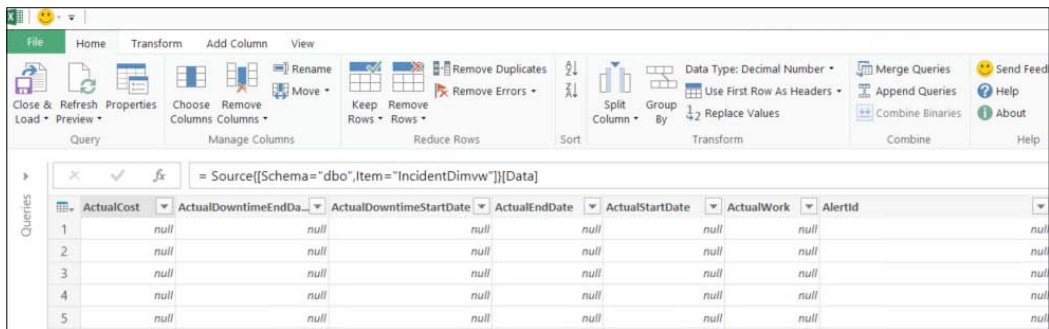
- Right-click on **IncidentDimvw (2)**, select **Properties...**, and rename the query (for example, `MyIncidentQuery`). Click on **OK** to complete the renaming of the query.



Explore the data and overview of the interface

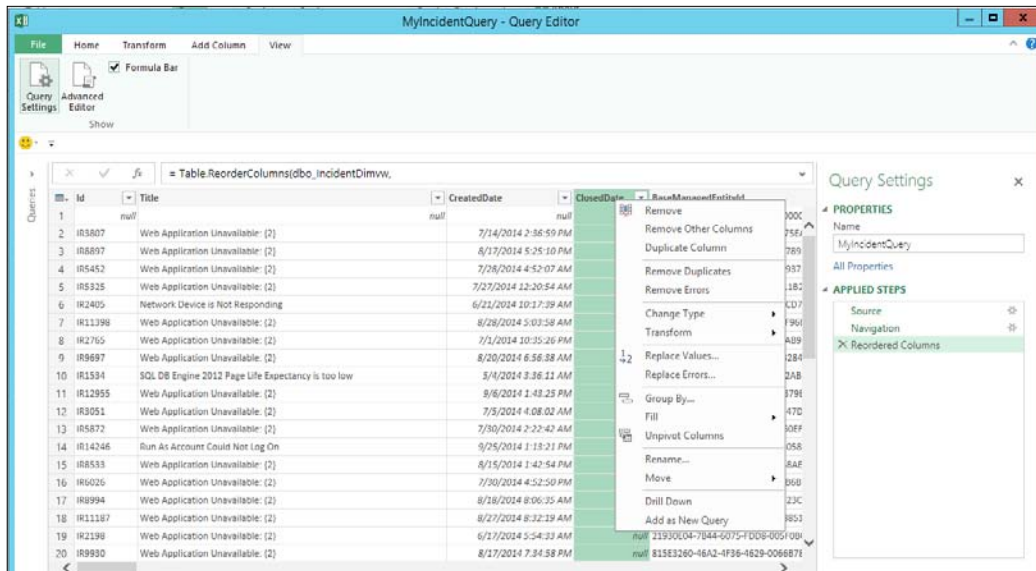
Follow these steps to explore and get an overview of the Power BI Excel interface:

- Right-click on the duplicated query (**MyIncidentQuery**) and select **Edit**.
- Query Editor** is opened. It provides an interface to model your dataset, as shown in the following screenshot:



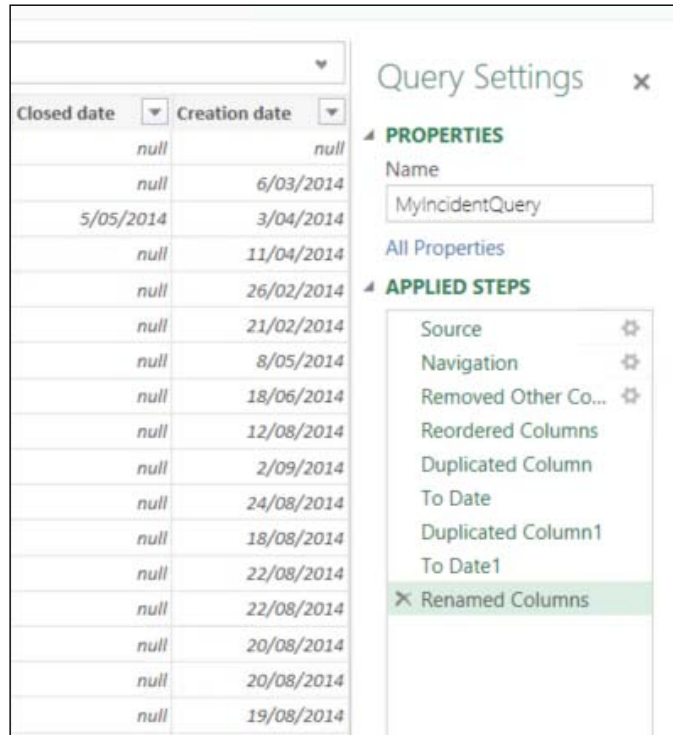
- Each tab in the **Query Editor** provides access to functions that help you shape the data, as shown in the screenshot above:
 - The applicable tasks on the ribbon are also available if you right-click on a column of your query.

- On the right-hand side of the editor, you have the step recorder. This pane of the query editor records all manipulations, and these are listed in the **Applied Steps** area. Each step can be updated, removed, or reordered from this pane.
 - The queried data is visible in the query editor sheet. Changes made in the data definition are also visible in this sheet.
4. Remove columns you do not require from the dataset:
- In the query editor, select the columns that you want to include in your query (use the *Ctrl* key for multiple column selection). Ensure that you select at least one **Date Time** column (for example, select **CreatedDate**, **Id**, **Title**, **Description**, **Classification**, **DisplayName**, **CloseDate**, and **Impact**).
 - Right-click on all the selected columns you need using the *Ctrl* key and select **Remove Other Columns**, as shown in the following screenshot:



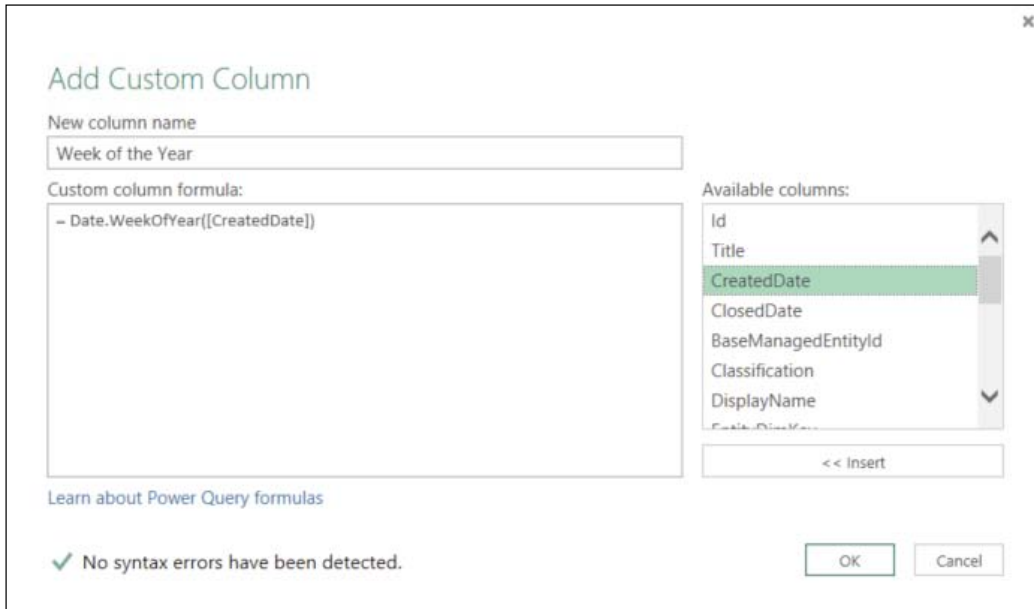
5. You can transform or change the data type of column data in the dataset using these steps:
- Right-click on a **Date/Time** (for example, **CreatedDate**) column in your query and click on **Duplicate Column**.
 - Navigate to the last column (**Copy of CreatedDate**). Right-click on the column and click on **Rename...** Remove **Copy of** and insert a space between **Created** and **Date**.

- Right-click on the **Created Date** column, click on **Change Type**, and go to **Data/Time | Time Zone**.
- Right-click on the **Created Date** column and click on **Transform** and **Date Only**. All right-click tasks provide functionality to manipulate data.
- All actions that are taken to update the query in the editor are recorded in the **Query Settings** area.



6. Data in the query can be the input for Power Query formulas. Date, time, number, text, and many other functions are available for you out of the box. You may have a requirement to know the week of the year when incidents are created. By performing the following steps you can update the query by creating a column that calculates this number based on the date of creation:
 - In the **Add Column** tab of the ribbon, click on **Add Custom Column**.

- On the **Add Custom Column** page, specify the column name. For this example, the column name is **Week of the Year**, as shown in the following screenshot:



- In the **Custom column formula** textbox, type the following formula: `Date.WeekOfYear ([CreatedDate])`. Then, click on **OK**.
 - The new column is added as the last column of the query.
 - Using the formula function and the functions that are available in your query editor (the `Date.WeekOfYear` example in this recipe) gives you the ability to perform transformations in the query editor.
7. Click on the **Close & Load** button on the **Home** tab and select **Close & Load** to save your query.

Extending the query with data from other queries

Follow these steps to extend your query using data from other queries you have created:

1. We will use the example at the beginning of this recipe, where we listed display names instead of internal names and set the language to **English (ENU)**:
 - On the **Workbook Queries** page, right-click on the **DisplayStringDimvw** query. Then, select **Edit**.
 - Click on the drop-down icon (filter) of the **LanguageCode** column, uncheck **(Select All)**, and only select the language code **(ENU)**.
 - Click on the **Close & Load** button on the **Home** tab and select **Close & Load** to save your query.
 - The filter is saved in the query. Merges with this query will provide only English display names.
 - The same principle of filtering data before defining data merges can be used for any System Center data analysis. Think about what you want to include in your dataset. This can be a specific language to monitor information or data from WAP.
2. Merging queries is similar to SQL joins, and it adds additional data to your query. For example, to get the friendly name of the **Impact** column data, right-click on the workbook query (the **duplicate of the IncidentDimvw** query; you may have renamed this to **MyIncidentsQuery**), and select **Edit**.
3. In the query editor on the **Home** tab, click on **Merge Queries**, as shown in the following screenshot:



4. On the **Merge** page, select the column that you want to initiate the merge on. In this example, the impact of the incident is selected.

5. Select the query you want to merge with in the lower pane. The **DisplayStringDimvw** query is selected in this example, and we will use **ElementName** as the common column to provide the merge-matching feature:
 - ❑ Merging queries requires that you have matching rows. The selection that is made on the **Merge** page is validated, and the validation result is displayed on the left-hand side at the bottom of the page.
 - ❑ The details required to get specific data can be found in the data model documentation of the System Center product you are querying, as shown in the following screenshot:

Merge

Select a table and matching columns to create a merged table.

Classification	DisplayName	EntityDimKey	Impact
000000	<i>null</i>	<i>null</i>	<i>null</i>
6F5610	IncidentClassificationEnum.Email	IR24 - test	812 System.WorkItem.Trouble
1042AA	IncidentClassificationEnum.Email	IR50 - test demo	1018 System.WorkItem.Trouble
F0B5F3	IncidentClassificationEnum.Hardware	IR190 - Screen issue	1949 System.WorkItem.Trouble
F96E46	IncidentClassificationEnum.Email	IR6 - Test IT Incident	2214 System.WorkItem.Trouble

< >

DisplayStringDimvw

DisplayStringDimKey	ElementName	InsertedB...
	7 System.WorkItem.TroubleTicket.UrgencyEnum.Low	
formation	30 System.WorkItemHasSLAInstanceInformation	
	53 System.ApplicationLog.IISLog.CorrelatedEventSingleEvent2StateMonitorT	
	76 System.ApplicationLog.GenericLog.MissingCorrelatedEventSingleEvent25t	
	99 System.ApplicationLog.IISLog.MissingEventTimer2StateMonitorTypePage5	

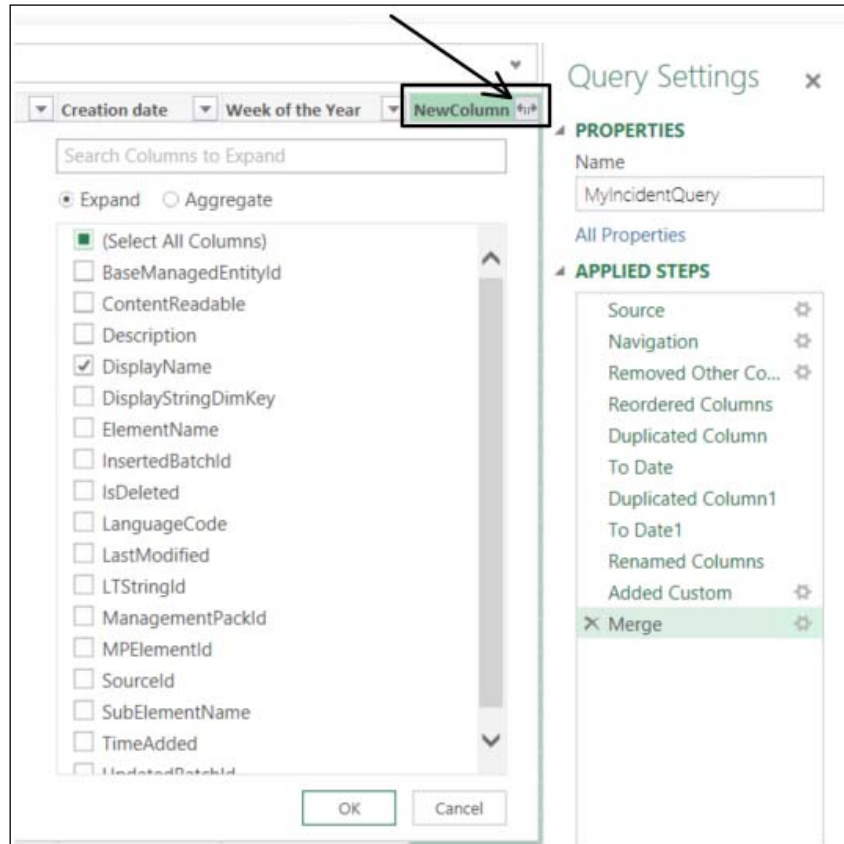
< >

Only include matching rows

i The selection has matched 27 out of the first 28 rows.

OK
Cancel

6. Click on **OK**. The merge action is added to the query, and a column is added to the end of the table view. Click on the filter icon (to the right-hand side of the new column), uncheck **(Select All)**, and select the **DisplayName** column. Click on **OK**.

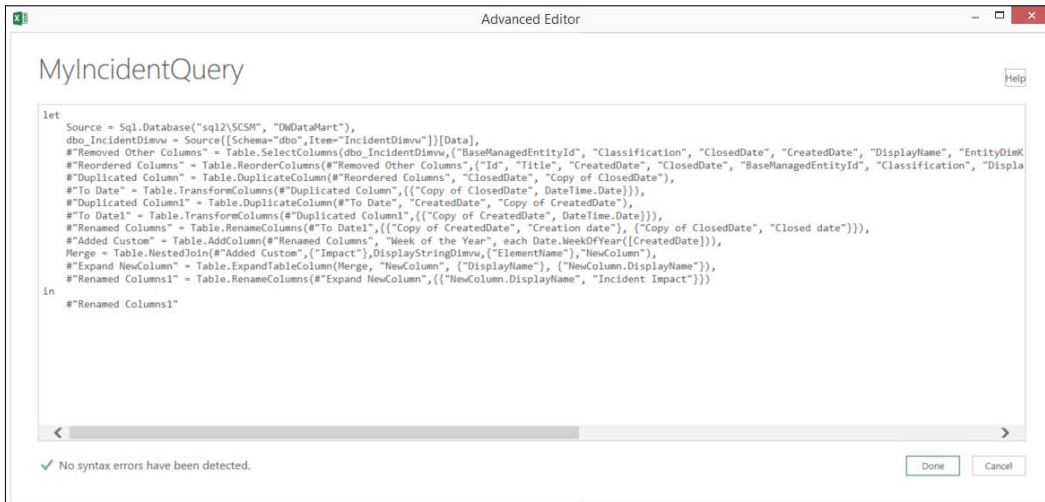


- You can rename the new column and even remove the original column that represented the same data before you performed the merge action, as shown in the following screenshot:

The screenshot displays a data table with the following columns: Closed date, Creation date, Week of the Year, and Incident Impact. The data rows show various dates and impact levels. To the right, the 'Query Settings' panel is open, showing the 'APPLIED STEPS' section. The steps listed are: Source, Navigation, Removed Other Co..., Reordered Columns, Duplicated Column, To Date, Duplicated Column1, To Date1, Renamed Columns, Added Custom, Merge, Expand NewColumn, and Renamed Columns1. The 'Renamed Columns1' step is currently selected and highlighted.

Closed date	Creation date	Week of the Year	Incident Impact
null	null	null	null
null	6/03/2014	10	Medium
5/05/2014	3/04/2014	14	Medium
null	11/04/2014	15	Medium
null	18/06/2014	25	Medium
null	12/08/2014	33	Medium
null	2/09/2014	36	Medium
null	26/02/2014	9	Low
null	21/02/2014	8	Low
null	8/05/2014	19	Low
null	24/08/2014	35	Low
null	18/08/2014	34	Low
null	22/08/2014	34	Low
null	22/08/2014	34	Low
null	20/08/2014	34	Low
null	20/08/2014	34	Low
null	19/08/2014	34	Low
null	24/08/2014	35	Low

8. The **Query Settings** pane can be made visible or hidden from the **View** tab. This is done by clicking on the **Query Settings** button. **Advanced Editor** gives you the complete syntax of your query. The syntax of the query you created in this recipe is illustrated in this screenshot:



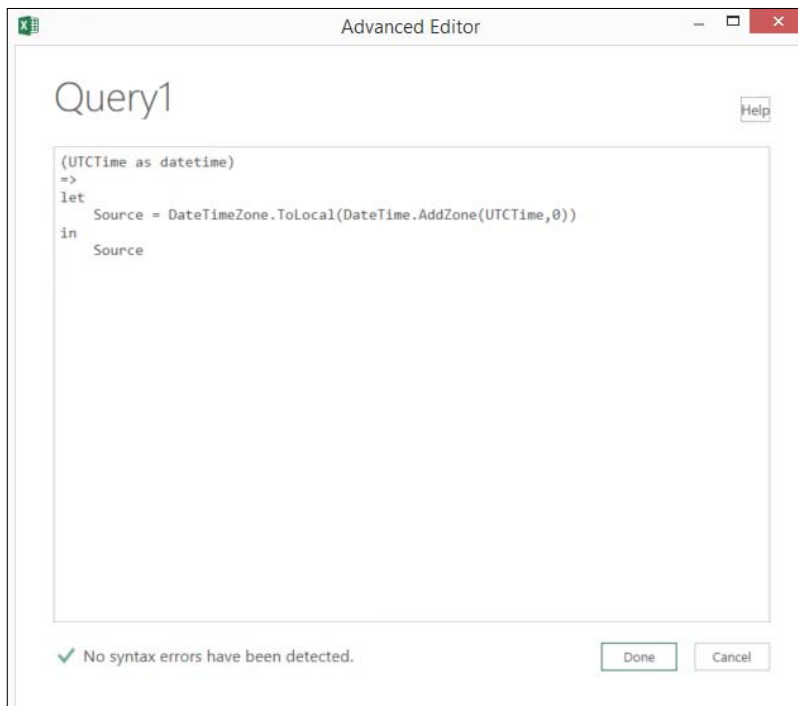
9. Click on the **Close & Load** button on the **Home** tab and select **Close & Load** to save your query.

Adding functions in your query

Follow these steps to add functions to your query:

1. In the **Power Query** tab, select **From Other Sources** and click on **Blank Query**.
2. The query editor opens. Select the **View** tab and click on the **Advanced Editor** button.
3. Type the following query in the **Advanced Editor**:

```
(UTCtime as datetime)
=>
let
    Source = DateTimeZone.ToLocal(DateTime.AddZone(UTCtime, 0))
in
    Source
```



4. The syntax of your query is validated. Click on **Done** to close the **Advanced Editor**.
5. On the right-hand side of the query editor, rename your query to `ToLocalTime` for the example used in this recipe.
6. Click on the **Home** tab. Click on the **Close & Load** button and select **Close & Load** to save your query.
7. Loading of functions is disabled by default. To use your functions, you must right-click on the query that you want to use the function in and click on **Edit**.
8. The **Query Editor** dialog opens. Click on the **Add Column** tab and then on the **Add Custom Column** button. The **Add Custom Column** option is displayed:
 - Provide a column name, such as `Locale Created Date` for this example
 - The formula will be `ToLocalTime([CreatedDate])`
9. Click on **OK**.
10. The new column is added to the query. The function can be reused for any date stamp in your dataset.

How it works...

This recipe is broken down as follows:

- ▶ **Connecting to System Center data sources:** Data gathering is the initial step in the creation of your query. The data in your dataset is retrieved from the selected data source. From this point, you start manipulating the data with the Power BI tools in Excel.
- ▶ **Exploring the data and overview of the interface:** This part of the recipe describes the functionality used to edit the original data load query. The data that you need to keep, manipulate, or add to the query depends on the required end result. In the example of the incident view from Service Manager, all incidents with their properties defined in the default view of SCSM were included in your dataset. Incident data that is not required for your analysis can be removed from the query. After shaping and manipulating the data, you can remove the data that was required only for calculations or manipulation.
- ▶ **Extending the query with data from other queries:** You can merge additional data from different data sources to create one super dataset. This may be required in order to translate column data into friendly names, and extend the query with information from other queries or data sources. In the example used in this recipe, Service Manager data was included in the query. Service Manager uses internal names for all list items that can be viewed and selected in the incident form. The equivalent friendly names are included in the `DisplayStringDimvw` query, which we included in the workbook queries at the beginning of this recipe. This is just an example of the capabilities of merging data from different queries. Different scenarios will exist for each System Center product. Additionally, the merge can also be done with a different System Center component (for example, **Orchestrator**).
- ▶ **Adding functions to your Query:** Calculations and manipulations are the key to perform an accurate analysis of data. Repetitive calculations or transformations on data can be included in a function and added to the workbook query list. Database date/time data is, in most cases, is time-zone-specific. Date and time entries that you see in the dataset can be localized. You can create a function to perform this localization. Calculations and manipulations are important in dataset creation. The examples used in the recipes describe functions for the date/time data type. Numbers, text, and many other data-type-specific functions are available for you to include in your query.

The steps you followed in this recipe are a simplified approach to creating a dataset with the Power Query add-in for Microsoft Excel. Most dataset manipulations can be done using the standard functionality included in the add-in. The steps taken are recorded in your query, and you can always refer back to a previous step.

There's more...

The example used in this recipe explained how you can create a dataset for Service Manager. The display name is retrieved from another query. Adding other data sources such as Operations Manager or Orchestrator database information is a simple task. Perform the following steps to create a dataset for Orchestrator:

1. In the **Power Query** tab, select **From Database** and click on **From SQL Server Database**.
2. The Microsoft SQL database page opens:
 - **Server:** Enter the name and instance of the other data source (for example, Orchestrator)
 - **Database:** Specify the other database (for example, the Orchestrator database) that you want to integrate in the query
3. In the **Navigator** pane, select the views from the Orchestrator database and click on **Load**. Load retrieves the data and brings it into the dataset. Alternatively, use **Load to..** to load the query only in the workbook queries.

Once the query is available in the workbook query list, you can start using it to extend your data query with the procedures that are described in this recipe.

See also

- ▶ More information about the Power Query add-in for Microsoft Excel can be found at <https://support.office.com/en-in/article/Introduction-to-Microsoft-Power-Query-for-Excel-6e92e2f4-2079-4e1f-bad5-89f6269cd605>.
- ▶ Detailed information on the functions that you can use in data modeling can be found in the official online documentation. You can refer to Microsoft Office Support at <https://support.office.com/en-us/article/Power-Query-formula-categories-125024ec-873c-47b9-bdfd-b437f8716819>. For an overview of Power Query formula categories, refer to <https://support.office.com/en-us/article/Power-Query-formula-categories-125024ec-873c-47b9-bdfd-b437f8716819>.

Creating the data model with PowerPivot

This recipe will show you how to use PowerPivot to define your data model and create visualization in your Excel sheet. Using the PowerPivot add-in, you can create relationships between queries, add formulas with **Data Analysis Expressions (DAX)**, or create **Key Performance Indicators (KPIs)** for your data model.

PowerPivot provides a set table and column properties to improve the visualization experience with tools such as Power View and Power Map. The big difference between Power Query and PowerPivot is the following:

- ▶ **Power Query:** Here, you can create, transform, and manipulate a dataset
- ▶ **PowerPivot:** Here, you can create a dataset model that can be used in your Pivot Table, Power View, or Power Map visualization

Getting ready

To demonstrate the capabilities of the PowerPivot add-in, System Center Operations Manager data is gathered in the dataset and further updated in this recipe. The data is performance data and the data of interest is the logical free disk space.

You must have access to a fully deployed Operations Manager environment in order to perform the steps in this recipe.

To complete this query, follow the instructions from the previous recipe with the information that follows:

1. Connect to the `OperationsManager` database and select (after checking the **Select multiple items** option) **Views and Tables** (you will be presented with all Views and Tables in the database when you connect). These items are sorted first by **Views** and then by **Tables** in descending order:
 - ❑ The `PerformanceDataAllView` view
 - ❑ The `Rulesview` view and the `BaseManagedEntity` table
 - ❑ `PerformanceSource` table
2. Once you have made your selection in the **Navigator**, click on the arrow next to **Load**. Select **Load to...** and click on **Load**.
3. Right-click on **PerformanceDataAllView** and select **Edit** to extend the dataset.
4. Click on **Merge Queries** from the ribbon. Use **PerformanceDataAllView** as the source (top part of the **Merge** dialog box).

5. Click on the little arrow in the lower pane of the **Merge** window. Select **PerformanceSource** as the target.
6. Click on the **PerformanceSourceInternalId** column in the source (**PerformanceDataAllView**) and the target (**PerformanceSource**) to be used as the merge column. Then click on **OK**.
7. Click on the double arrow icon to the right-hand side of the **NewColumn**; uncheck **Select All Columns**; select **BaseManagedEntityId**, **RuleId**, and **PerfmonInstanceName**; and then click on **OK**.
8. Click on **Merge Queries** from the ribbon. Note that you do not need to select a new source. The result of the previous query (the extended **PerformanceDataAllView** view) is the source (top part of the **Merge** dialog box).
9. Click on the little arrow in the lower pane of the **Merge** window and select **RuleView** as the target.
10. Click on the **NewColumn.RuleId** column in the extended source (**PerformanceDataAllView**).
11. Click on the **Id** column in the target (**RuleView**) as the target merge column. Then click on **OK**.
12. Click on the double arrow icon to the right-hand side of **NewColumn**, uncheck **Select All Columns**, select **Name**, and then click on **OK**.
13. Click on **Merge Queries** from the ribbon. Note that you do not need to select a new source. The result of the previous query (the extended **PerformanceDataAllView** view) is the source (top part of the **Merge** dialogue box).
14. Click on the little arrow in the lower pane of the **Merge** window and select **BaseManagedEntity** as the target.
15. Click on the **NewColumn.BaseManagedEntityId** column in the extended source (**PerformanceDataAllView**).
16. Click on the **BaseManagedEntityId** column in the target (**BaseManagedEntity**) as the target merge column, and click on **OK**.
17. Click on the double arrow icon to the right-hand side of **NewColumn**, uncheck **Select All Columns**, select **Path**, and then click on **OK**.
18. Click on the arrow icon to the right of **NewColumn.Name**. Uncheck **Select All**. Type `Microsoft.Windows.Server.6.2.LogicalDisk.FreeMB.Collection` in the search box, and then click on **OK**.
19. In the ribbon, click on **Close & Load**.
20. Save the Excel.
21. This is the base query for the recipe in this section of the chapter.

The following table lists what is required to complete this recipe:

Requirement or information	Description
Dataset as described in the <i>Getting ready</i> section	Use a pre-created dataset for logical free disk space data. This is the saved Excel workbook.

This table will serve as an input for the tasks required to complete this recipe.

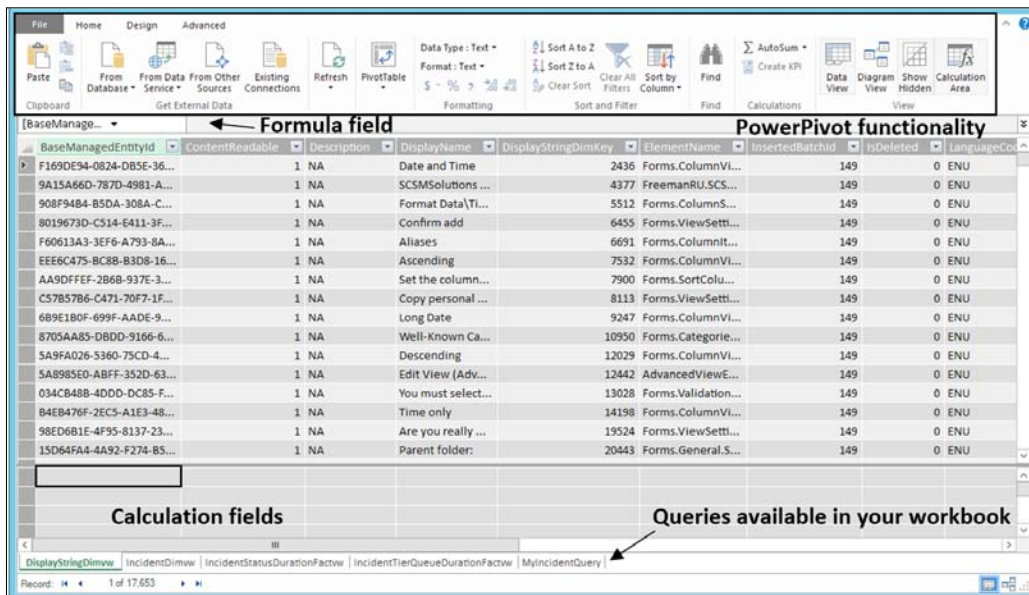
How to do it...

The following steps will show you how to create the report for this recipe.

Data model design

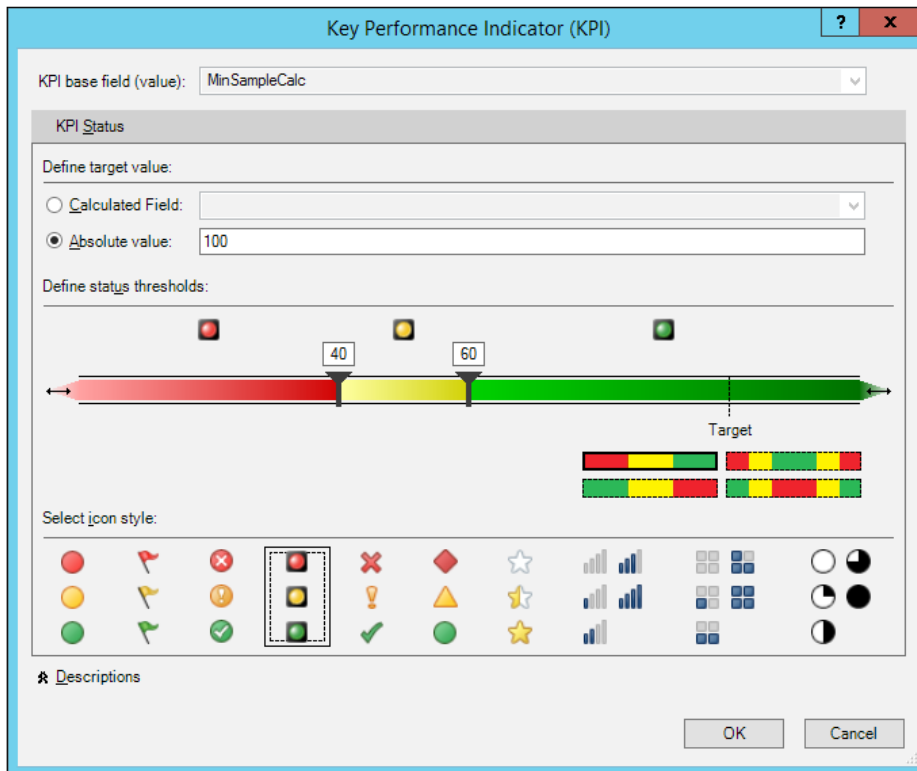
This recipe's steps begin with your data model design:

1. Open the Microsoft Excel workbook with this recipe's example query. Click on the **POWERPIVOT** tab and then on the **Manage** button. PowerPivot for Excel is launched, and you can start editing the data model, as shown in the following screenshot:



2. In the **Home** tab, select **Diagram View** to see the relationships between the different queries in your workbook:
 - Based on matching data in each table, you can create a relationship between two tables of data so that Excel can cross-reference rows in one table with those of another table
 - In a data model, table relationships can be one-to-one or one-to-many, but not many-to-many
 - The data types in the two columns must be compatible
 - Power Query merges can bring data from different sources into one data model
3. Click on the **Data View** icon and then on a location in the calculation fields:
 - Click on the formula field to create your calculation
 - The calculation definition format is `<Calculation name>:=<DAX calculation>`
 - Type `MinSampleCalc:=MIN([SampleValue])` in the formula field
4. Click on the **Validate** (√) icon to the left-hand side of the formula field.
5. Click on the **Design** tab and then on the **Add** icon.
6. Type `=DAY([TimeSampled])` in the formula field.
7. Click on the **Validate** (√) icon to the left-hand side of the formula field. A new column, called `CalculatedColumn1`, is created. Right-click on the new column, select **Rename Column**, and change the name to `Day`.
8. In the **Design** tab, click on the **Add** icon.
9. Type `=Month([TimeSampled])` in the formula field.
10. Click on the **Validate** (√) icon to the left-hand side of the formula field. A new column, called `CalculatedColumn1`, is created. Right-click on the new column, select **Rename Column**, and change the name to `Month`.
11. Click on the **Home** tab and then on the **MinSampleCalc**: cell in the calculated fields' area.
12. Click on the **Create KPI** button and click on **Absolute value**:. Ensure that the default value is **100**:
 - To create KPIs, you need to have calculation fields in the data model. This is because KPIs are based on calculations that are defined in the data model, not on raw data columns.
 - We will use the `MinSampleCalc` calculation field, which was created in the previous recipe. The KPI base field will be `MinSampleCalc`.

- You can move the sliders in the middle of the page to the desired values. For this example, 40 percent of logical free space is in red, and everything above 60 percent is in green. The percentage between 40 and 60 percent is marked yellow.
- Different icon styles are available for showing the KPI status.



13. Click on **OK** to save the KPI definition.

14. Click on the **Pivot Table** icon and select **Pivot Table**. Then click on **OK**.

Visualization with PowerPivot

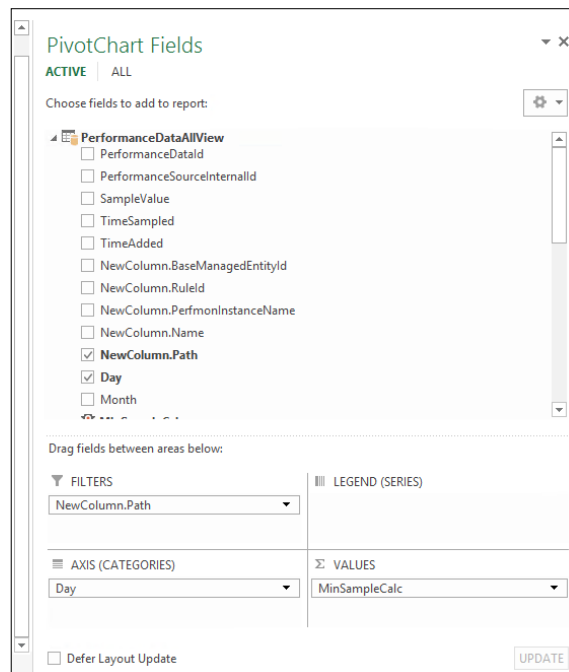
Next, this recipe describes the steps to create KPIs and visualize the data in **Pivot Table**:

1. The previous recipe ended with the creation of a Pivot Table; this opens the interface to configure your chart.
2. Under **PivotTable Fields**, expand **PerformanceDataAllView** and drag **NewColumn.Path** onto **FILTERS**.
3. Drag **Day** onto **ROWS**.

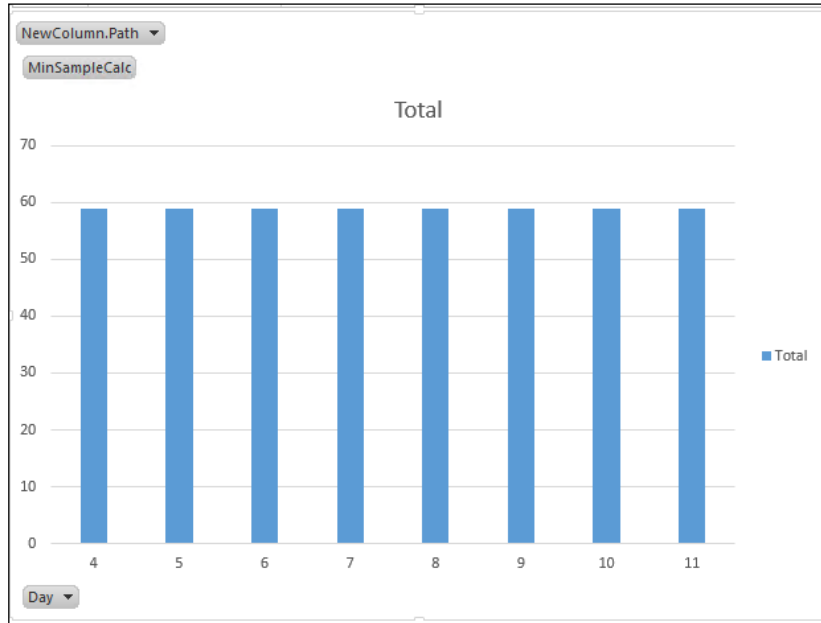
- Expand **MinSampleCalc** and drag **Value (MinSampleCalc)**, **Goal**, and **Status** onto **VALUES**, as shown in the following screenshot:

Row Labels	Column Labels													
	9.00	10.00	11.00	12.00	13.00	14.00	15.00							
	Min Free	KPI	Min Free	KPI	Min Free	KPI	Min Free	KPI	Min Free	KPI	Min Free	KPI	Min Free	KPI
CM01.contoso.com	65.27	65.21	65.19	65.17	65.01	64.92	64.87							
dc01.contoso.com	75.60	75.38	75.03	74.80	74.88	75.28	75.09							
dc02.contoso.com	90.22	90.28	90.27	89.79	89.67	90.10	90.32							
hyperv01.contoso.com	24.61	24.58	24.47	24.42	24.39	24.38	24.37							
OM01.contoso.com	70.28	70.31	70.28	70.31	70.29	70.26	70.44							
OR01.contoso.com	68.39	68.32	68.17	67.22	67.16	67.16	67.10							
SM01.contoso.com	72.09	72.09	72.08	72.08	72.09	72.10	72.10							
SM02.contoso.com	75.82	75.81	75.80	75.80	75.78	75.76	75.75							
sql01.contoso.com	2.47	2.39	2.37	2.35	2.32	2.30	2.31							
VMM01.contoso.com	50.17	50.17	50.17	50.17	50.17	50.17	50.17							
WAPAdmin01.contoso.com	72.73	72.84	72.61	72.61	72.62	72.53	72.53							
WAPTenant01.contoso.com	75.14	75.10	75.04	75.06	75.04	74.97	74.96							

- Click on the **File** tab and save your Excel workbook.
- Click on the **Insert** tab, then click on **PivotChart**, and select **PivotChart**.
- Ensure that **Column** is selected and then click on **OK**.
- Ensure that **NewColumn.Path** is selected as the filter (you may have renamed this column to **Path**).
- Ensure that **Day** is selected as **AXIS (CATEGORIES)**, as shown in the following screenshot:



10. The chart is updated with data after the configuration of the area.



11. Click on the **File** tab and save your Excel workbook.

How it works...

The recipe shows you how to update the data model and visualize the data using the PowerPivot functionality. The dataset you created with the Power Query add-in is further enhanced using the PowerPivot functionality.

The first exercise in this recipe is to ensure that all of the required data is available in the dataset. There are different ways to get data in the data model. Using Power Query gives you the option to merge data. Using calculation columns, you can also retrieve data from other queries or data sources. This requires some knowledge of how to build your formula (DAX).

You can use DAX to define custom calculations for calculated columns and calculated fields (measures). DAX is *not* a programming language but a formula language. DAX functions make it possible to create reports quickly and define complex data manipulations in the data model. Many DAX functions have the same functionality and names as familiar Excel functions. However, these functions have been modified to use DAX data types and work with tables and columns.

Creating visualization is just a matter of dragging and dropping the data column names into the area of the Pivot Table or PivotChart. The example in this recipe shows how you can visualize logical free disk space in a Pivot Table and PivotChart.

There's more...

Data model definition and visualization are totally dependent on what you want to achieve. Many sources can be used as input for data modeling, just as in Power Query. This makes it possible to connect directly to the System Center databases and start from an empty data model.

Review links in the *See also* section for additional information on the PowerPivot add-in and the DAX library.

Data models and queries can be shared through the Power BI for Office 365 Microsoft online service.

See also

- ▶ More information on the PowerPivot add-in for Microsoft Excel can be found at <http://office.microsoft.com/en-gb/excel-help/power-pivot-powerful-data-analysis-and-data-modeling-in-excel-HA102837110.aspx?CTT=5&origin=HA104125038>
- ▶ For information on how Power Query and PowerPivot work together, refer to <http://office.microsoft.com/en-gb/excel-help/how-power-query-and-power-pivot-work-together-HA104125038.aspx>
- ▶ Detailed information on the DAX functions that you can use in data modeling can be found in the official online documentation at <http://office.microsoft.com/en-gb/excel-help/types-of-dax-functions-HA102836089.aspx?CTT=5&origin=HA102836919#top>

Visualizing the analysis with Power View and Power Maps

Power View and Power Maps provide you with the means to get interactive data exploration, visualization, and a presentation experience that enables intuitive ad-hoc reporting. Compared to PowerPivot and Power View, Power Maps are purely an alternative to approaching the visualization layer of PowerPivot. The next section describes the creation of a Power View sheet in your Excel workbook.

Getting ready

You must complete the *Creating the data model with PowerPivot* recipe as a prerequisite to this recipe.

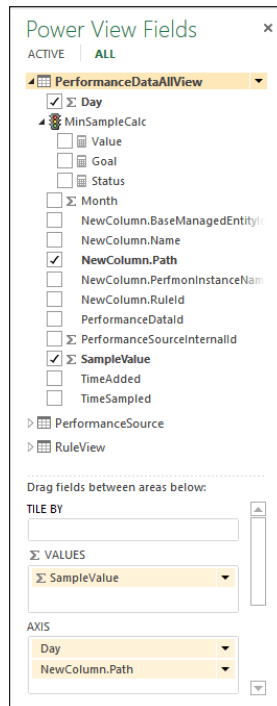
How to do it...

The steps to visualize the analysis with Power View and Power Maps follow.

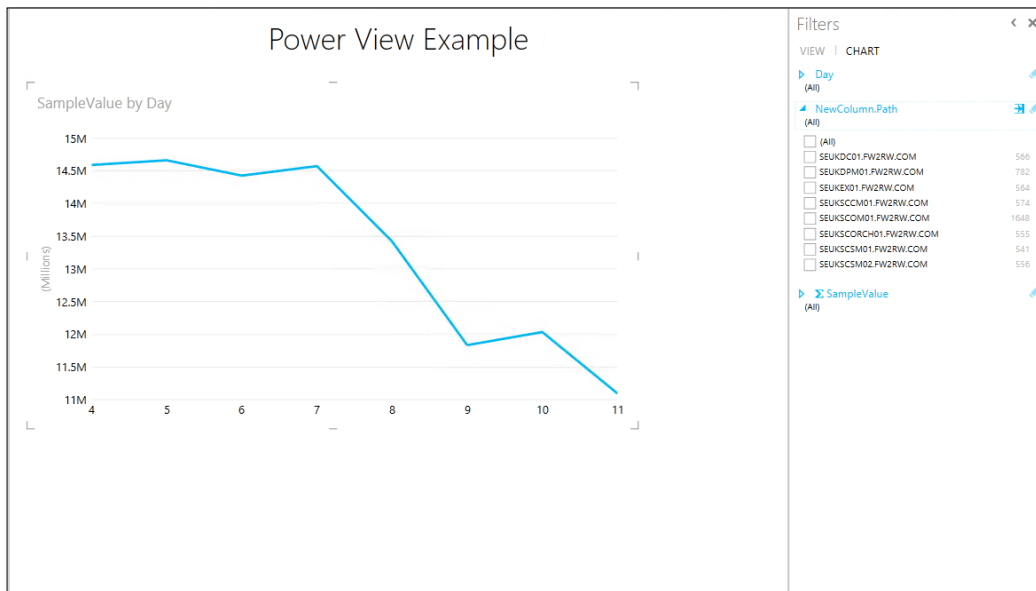
Creating Power View visualization

In this recipe, the same example as that in the *Creating the data model with PowerPivot* recipe is used. The following steps will show you how to create a Power View sheet from the data model created in the PowerPivot recipe:

1. Open the Excel sheet created in the *Creating the data model with PowerPivot* recipe.
2. On the **Insert** tab, click on **Power View**.
A new sheet is created in the Excel workbook to configure Power View.
3. Expand **PerformanceDataAllView** under **Power View Fields** (on the right-hand side) and check Σ **Day**.
4. In the **File** tab in the ribbon area, click on **Other Chart** and select **Line**.
5. Click on **Day**, drag it from the **VALUES** area, and drop it into the **AXIS** area.
6. Click on **NewColumn.Path** under **PerformanceDataAllView** and drag it into **AXIS**.
7. Click on Σ **SampleValue** and drag it into **VALUES**, as shown in the following screenshot:



8. Click on the title space and type `Power View Example`.
9. Drag the right-hand corner of the chart box to enlarge the chart for better visibility.
10. Power View is immediately updated with data after the configuration of the area, as shown in this screenshot:



11. Save the Excel sheet.

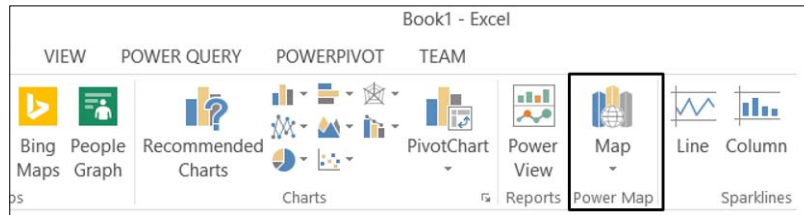
Creating Power Map visualization

With the Microsoft Power Map add-in for Excel, we can take a big step in data visualization. Data can be visualized three-dimensionally (3D) and also based on the geographic (map) location. Visualization can be created as 3D Bing maps or a custom map for your data layout. Data is delivered from the query. Then it is filtered and visualized on the map. With Power Map, you can create the following:

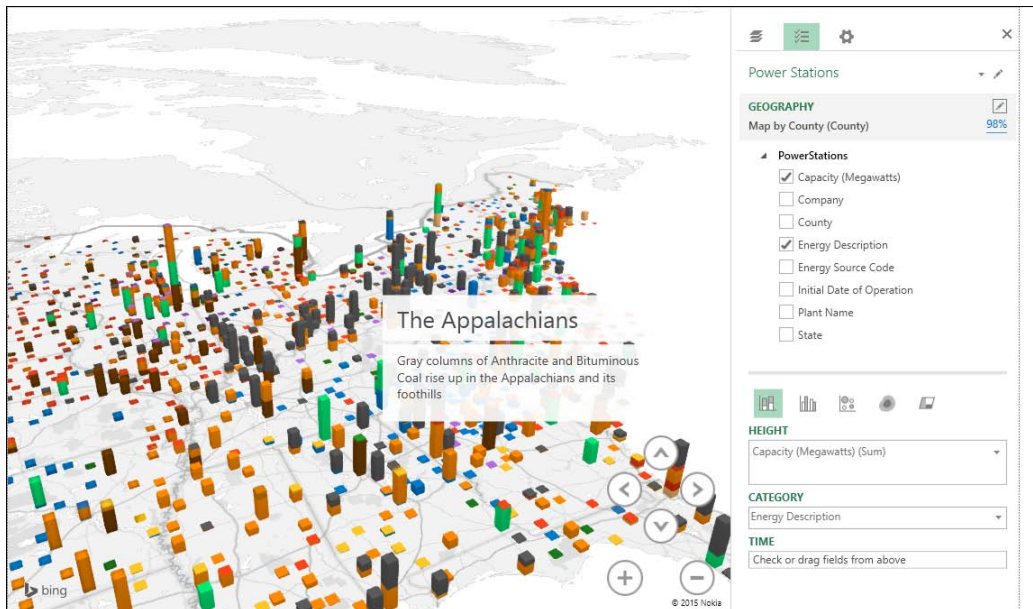
- ▶ **Geographical data mapping:** Visualize your Excel tables or data model, which can include more than a million rows on the maps. Maps could be based on **Bing Maps** or custom maps that represent an environment of interest.
- ▶ **New insights:** Data visualization of your data in a geographic representation that visualizes data changes (with timestamps) over specific periods creates a new insight into analysis.

- ▶ **Share complete data analysis stories:** Power Maps completes the Power BI add-ons for Microsoft Excel. You can extend this functionality by creating guided tours of the analysis that you share with colleagues as a link, or export the analysis for offline use.

You'll find the **Map** button in the **Tours** group on the **Insert** tab of the Excel ribbon, as shown in this screenshot:



The configuration page is shown in a separate edit page on the right-hand side of the map. Data can be mapped onto geographical locations from this page. The following illustration shows an example map that is used for starting a Power Map (https://support.office.com/client/Power-Map-88a28df6-8258-40aa-b5cc-577873fb0f4a#_exploring_sample_datasets). Different public examples are available for downloading and experimentation with data map visualization.



How it works...

The data source for Power View is a data model in the same Excel workbook as the PowerPivot sheet. The data model created by PowerPivot is what Power View interacts with.

It's all about visualization of data with Power View; you can quickly create different kinds of visualizations, from tables and matrices to pie, bar, and bubble charts. To find the best visualization to illustrate your data, start with a table. When you've finished configuring the table, convert it to the visualization that fits your requirements. Power View only enables visualization that works best for data in the source table.

Besides the visualization of data, the Power View sheet can be provisioned with additional functionality to explore the data. Filter configuration can be made visible or hidden. Slicers in Power View enable you to compare and evaluate your data from different perspectives. When you have multiple slicers on a view and you select an entry in one slicer, that selection filters the other slicers in the view.

Power Map uses Bing Maps to create a geographical representation of your data. You have the ability to create your own map for specific locations; for example, you can use your data center layout to visualize its defined KPIs.

Using Microsoft Cloud services to share and visualize System Center data

There are different solutions available for extending the standalone approach using Microsoft Excel, to a more centralized shared environment. This can be done with SQL Server or the integration in a SharePoint portal. This requires the initial cost of hardware and the software license and an extra load on the virtualized or physical environment before the service can be consumed. The alternative approach is to use a cloud service to create a centralized and shared environment.

Getting ready

Power BI for Office 365 is a service in Office 365 that you can subscribe to in order to gain access to its cloud capabilities.

The following table lists what is required to complete this recipe:

Requirement or information	Description
Power BI for Office 365 plan	This is the appropriate Microsoft Services subscription
Excel	Microsoft Excel for creating and editing the cloud options of Power BI

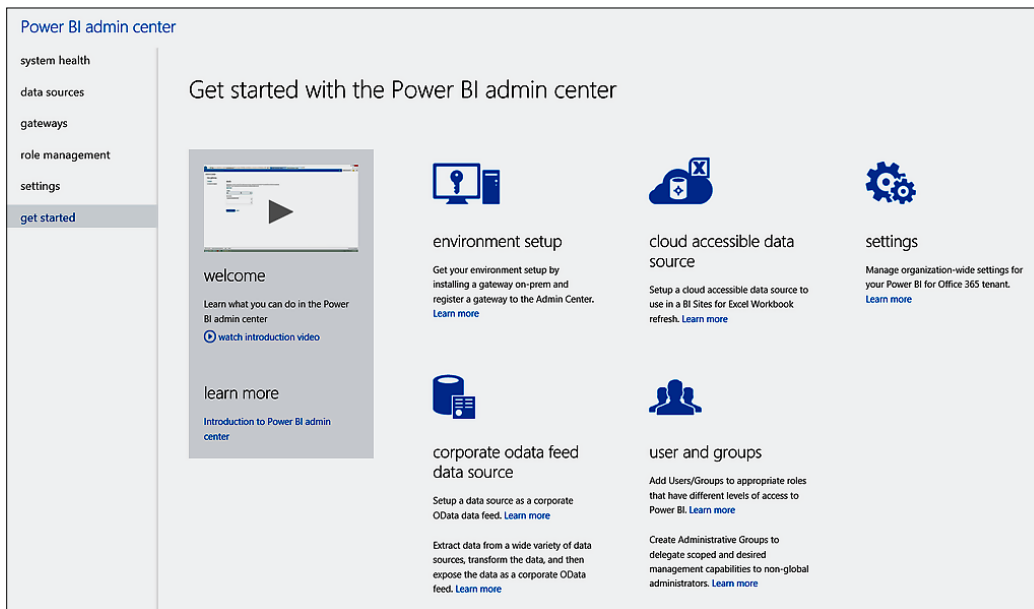
This table will serve as an input for the tasks required to complete this recipe.

How to do it...

Power BI for Office 365 must be enabled with a subscription in the Office 365 admin portal. Once enabled, you can configure your Power BI site in Office 365 to share and visualize data queries or analyses.

Connect to your Power BI for the Office 365 admin center (<http://admin.powerbi.com>).

Navigate to the **get started** tab on the left-hand-side menu and follow the instructions in the procedures available for downloading. In the right-hand-side pane of **Admin Center**, you have up-to-date guidance for configuring **Data Management Gateway**, creating shared data sources, and managing access to these shared data sources, as shown in the following screenshot:



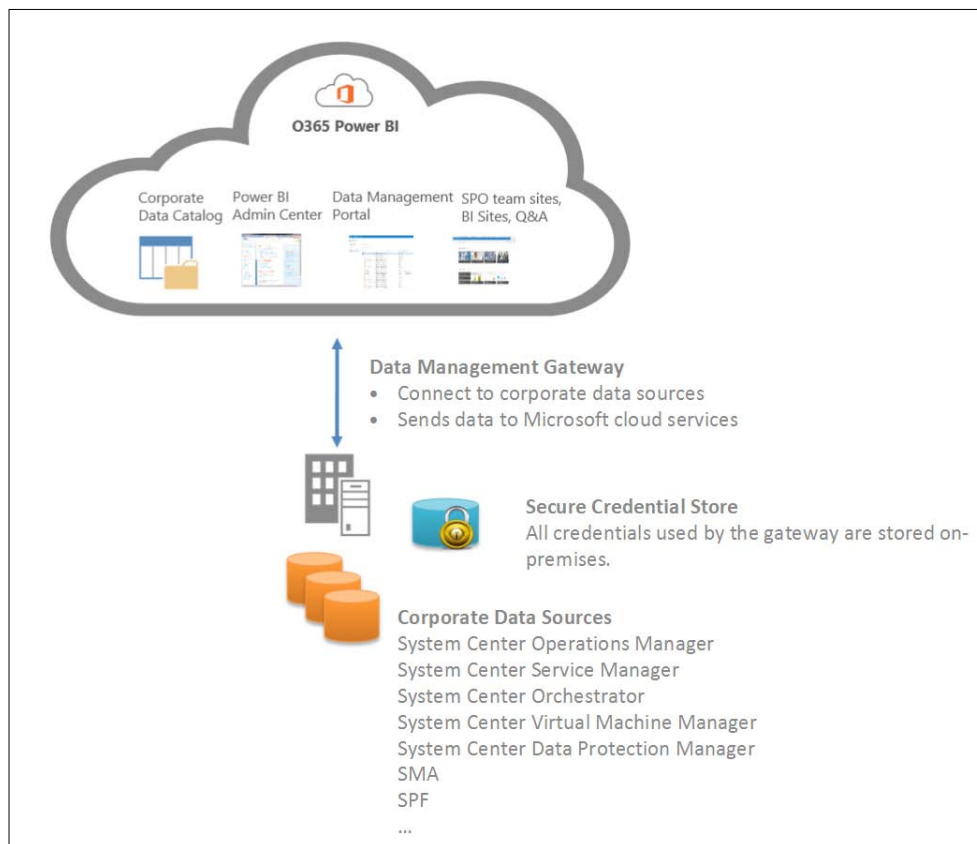
How it works...

Sharing, visualization, **Question and Answer (Q&A)**, and accessibility flexibility for Power BI are enabled with your subscription. No on-site (high-specification) infrastructure investment is required to start consuming this Microsoft service. Uploading and providing access to the reports is configured using SharePoint functions on the site. All of the processing is handled by the Microsoft service. This will work if you have direct access to the data source.

With Data Management Gateway, you can register on-premise data sources before enabling OData feed or cloud access for the data sources. The Data Management Gateway service is installed locally and responds to requests from the cloud service:

- ▶ When creating a data source, the credentials are encrypted with a certificate owned by you and are stored in the credential store. The credential store can be configured either within the gateway (on premise) or in the cloud service.
- ▶ To respond to data requests from the cloud service, the gateway decrypts the request with the certificate to get the credentials, and uses credentials to pull data from the data source.

Data sources are configured in the Power BI admin center. Tables or views from a database can be made available using OData feed access. Data is parsed through the gateway. Power Query provider, for example, is bundled with Data Management Gateway to connect the queries and execute Power Query to load data in the Excel sheet, as shown in the following diagram:



Power BI for Office 365 eliminates the cost of setting up an environment, and provides a complete Power BI service that is ready to be consumed. IT administrators can use the Power BI Admin Center to configure the BI infrastructure on the cloud. You can configure and manage role-based access to this cloud service and the data sources available for connected users. Data Management Gateway is the connection between the cloud and on-premise data sources. The exposed data from on-premise sources acts as feeds that can be consumed by Power Query for Excel. Shared data source usage is monitored, and this usage information is stored in the Power BI admin site.

Data queries or visualizations created with Excel add-ins can be shared across the organization through the Power BI (SharePoint) site. Power BI site allow users to upload the query reports they create and share these with anyone from their organization. Functionality in the Power BI site can be configured to make it easier for users consuming data and insights. The Q&A functionality in the Power BI site interprets your question and immediately serves the correct answer in the form of an interactive chart or graph. Q&A answers are provided using natural language queries. These visualizations change dynamically as you modify the question, creating a truly interactive experience with your data.

The Power BI site can be integrated in an existing Office 365 SharePoint site, or you can create a dedicated new site for this purpose. Access to the site is managed using SharePoint site configuration tools.

Another access portal to shared data can be provided using the Power BI for the Mobile Windows Store app. This app provides live mobile access to important business information stored in users' Office 365 accounts. HTML5 support for Power BI sites allows users to consume Power BI reports virtually anywhere, on any device.

See also

- ▶ A direct link to the installation procedure for Power BI for Office 365 can be found at <https://support.office.com/en-us/article/Create-a-Data-Management-Gateway-2ddfe0c0-bdb3-42e9-b179-aa5e39e7eab9>
- ▶ The download of the executable, along with the installation procedure and configuration of Data Management Gateway, is detailed on the portal and on Microsoft Office Online at <https://support.office.com/en-us/article/Power-BI-for-Office-365-Admin-Center-Help-5e391ecb-500c-47a3-bd0f-a6173b541044>

Useful Websites, Chapter Code, and Community Resources

Our aim at the time of writing this book was to provide a foundation for reporting knowledge and reusable skills in this business-valued area. The book is complementary to the vast pool of resources available to you, including the community websites listed in this appendix.

This appendix will list some helpful websites and communities for reporting skills resources. These resources are not always easy to find, so this should serve as a shortcut to find what you need. Note that some links may have been updated, so use your favorite browser to search using the titles as key phrases.

We recommend that you bookmark the sites and follow the blogs to enhance your knowledge.

Additionally, we have included the code referenced in *Chapter 7, Creating Reports for System Center Service Manager and Orchestrator*, for you to type appropriately. The code is included in files that are available for download at <https://www.packtpub.com/virtualization-and-cloud/microsoft-system-center-reporting-cookbook-raw>.

Authors' community blogs

- ▶ *Samuel Erskine*: <http://www.itprocessed.com>
- ▶ *Kurt Van Hoecke*: www.scug.be/scsm and www.authoringFriday.com
- ▶ *Dieter Gasser*: <http://blog.dietergasser.com/>

Useful community blogs

- ▶ *System Center Team Blog*: <http://blogs.technet.com/b/systemcenter/>
- ▶ *Nathan Lasnoski*: <http://blog.concurrency.com/author/nlasnoski/>
- ▶ *Garth Jones*: <http://be.enhansoft.com/>
- ▶ *System Center 2012 Service Manager Survival Guide*: <http://social.technet.microsoft.com/wiki/contents/articles/8113.system-center-2012-service-manager-survival-guide.aspx>

Websites for SQL and reporting

- ▶ *SQL query writing*: <http://www.w3schools.com/sql/default.asp>
- ▶ *SQL and reporting services courses*: www.lynda.com
 - *Reporting Services In-depth*
 - *SQL Server 2008 Essentials Training*

Microsoft's official page on Report Builder: <http://msdn.microsoft.com/en-us/library/dd220460.aspx>

Reporting Services Tutorials (SSRS): <http://msdn.microsoft.com/en-us/library/bb522859.aspx>

Microsoft SQL Server Community Projects and Samples: <http://sqlserversamples.codeplex.com/>

WiseOwlTutorials (SSRS): <http://www.youtube.com/user/WiseOwlTutorials>

SQL community website: <http://www.sqlcoffee.com/index.htm>

Cost-effective vendor reporting visualization and analytical options

Squared up provides great enhancements to SCOM reporting options for creating dashboards and integration with Microsoft Visio. You can find out more about this at <https://www.squaredup.com>.

Microsoft Azure Operational Insights (OpInsights) is the next generation in cloud-based analysis of semi-structured and unstructured data analysis and reporting. You can connect your SCOM environment (or environments) and create reports with very little configuration effort. Find out more about this at <https://OpInsights.Azure.com>.

Online wikis and curations

Microsoft TechNet Wiki page on SQL Database Engine: <http://social.technet.microsoft.com/wiki/contents/articles/22759.sql-server-general-database-engine-resources-on-the-technet-wiki.aspx>

Microsoft TechNet Wiki page on Opalis Survival Guide: <http://social.technet.microsoft.com/wiki/contents/articles/768.opalis-survival-guide.aspx>

SQL Server on Curah!: <http://curah.microsoft.com/44214/sql-server>

SSRS Expression Examples: <https://curah.microsoft.com/23335/ssrs-expression-examples>

Social network resources

Microsoft SQL Server on Facebook: <https://www.facebook.com/sqlserver>

SSRS on Twitter: <https://twitter.com/SSRS>

Chapter code

This section lists the code referenced in the relevant chapters that you must type to complete the recipe steps. You can also download the code files from the Packt Publishing website using the link provided in the introduction section of this appendix.

Custom.IncidentDashboard.View.xml code

The following is the code:

```
<ManagementPack ContentReadable="true" SchemaVersion="1.1"
OriginalSchemaVersion="1.1" xmlns:xsd="http://www.w3.org/2001/
XMLSchema" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <Manifest>
    <Identity>
      <ID>Custom.IncidentDashboard.View</ID>
      <Version>1.0.0.0</Version>
    </Identity>
    <Name>Custom.IncidentDashboard.View</Name>
    <References>
      <Reference Alias="System">
        <ID>System.Library</ID>
        <Version>7.5.8501.0</Version>
```

```

        <PublicKeyToken>31bf3856ad364e35</PublicKeyToken>
    </Reference>
    <Reference Alias="Console">
        <ID>Microsoft.EnterpriseManagement.ServiceManager.UI.Console</
ID>
        <Version>7.5.1561.0</Version>
        <PublicKeyToken>31bf3856ad364e35</PublicKeyToken>
    </Reference>
    <Reference Alias="WorkItem">
        <ID>ServiceManager.WorkItem.Library</ID>
        <Version>7.5.1561.0</Version>
        <PublicKeyToken>31bf3856ad364e35</PublicKeyToken>
    </Reference>
</References>
</Manifest>
<PresentationTypes>
    <ViewTypes>
        <ViewType ID="ServiceManager.DashboardViewType"
Accessibility="Public">
            <Configuration>
                <xsd:any minOccurs="0" maxOccurs="unbounded"
processContents="skip"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" />
            </Configuration>
            <ViewImplementation>
                <Assembly>Console!WpfViewsAssembly</Assembly>
                <Type>Microsoft.EnterpriseManagement.UI.WpfViews.Overview</
Type>
            </ViewImplementation>
        </ViewType>
    </ViewTypes>
</PresentationTypes>
<Presentation>
    <Views>
        <View ID="CustomIncidentDashboardView" Accessibility="Public"
Enabled="true" Target="System!System.Entity"
TypeID="ServiceManager.DashboardViewType" Visible="true">
            <Category>Overview</Category>
            <Configuration>
                <Definitions />
                <Presentation>
                    <Header />
                    <Content>

```

```

        <WebBrowser Name="wb1"
Source="http:// [SSRSServerName] /Reports/Pages/Report.aspx?ItemPath
=%2fSystemCenter%2fServiceManager%2fCustom%2fIncident+Dashboard"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" />
        </Content>
    </Presentation>
</Configuration>
</View>
</Views>
<FolderItems>
    <FolderItem ElementID="CustomIncidentDashboardView"
ID="CustomIncidentDashboardView.FolderItem"
Folder="WorkItem!ServiceManager.Console.WorkItem.Root" />
</FolderItems>
</Presentation>
<LanguagePacks>
    <LanguagePack ID="ENU" IsDefault="true">
        <DisplayStrings>
            <DisplayString ElementID="Custom.IncidentDashboard.View">
                <Name>Custom Incident Dashboard Views</Name>
            </DisplayString>
            <DisplayString ElementID="CustomIncidentDashboardView">
                <Name>Incident Dashboard</Name>
            </DisplayString>
        </DisplayStrings>
    </LanguagePack>
</LanguagePacks>
</ManagementPack>

```

Accessing data using Microsoft Excel query code

Use DWDataMart:

GO

```
CREATE VIEW v_Custom_r_AllIncidents AS
```

```
SELECT
```

```

    I.Id AS [ID],
    CONVERT(smалldatetime, I.CreatedDate) AS [Created Date],
    IStatusDS.DisplayName AS [Status],
    AffectedUser.DisplayName AS [Affected User],
    I.Title AS [Title],
    I.Description AS [Description],

```

```
IImpactDS.DisplayName AS [Impact],
IUrgencyDS.DisplayName AS [Urgency],
I.Priority AS [Priority],
ISourceDS.DisplayName AS [Source],
ITierQueueDS.DisplayName [Support Group],
AssignedTo.DisplayName AS [Assigned To],
IClassificationDS.DisplayName AS [Classification Category],
CONVERT(smallerdatetime, I.ResolvedDate) AS [Resolved Date],
IResolutionCategoryDS.DisplayName AS [Resolution Category],
I.ResolutionDescription AS [Resolution Description],
CONVERT(smallerdatetime, I.ClosedDate) AS [Closed Date],
CreatedBy.DisplayName AS [Created By],
ResolvedBy.DisplayName AS [Resolved By]
FROM
    IncidentDimvw I

LEFT OUTER JOIN WorkItemDimvw WI ON
    I.EntityDimKey = WI.EntityDimKey

LEFT OUTER JOIN    WorkItemAffectedUserFactvw ON
    WI.WorkItemDimKey = WorkItemAffectedUserFactvw.WorkItemDimKey
    AND WorkItemAffectedUserFactvw.DeletedDate IS NULL
LEFT OUTER JOIN UserDimvw AS AffectedUser ON
    WorkItemAffectedUserFactvw.WorkItemAffectedUser_UserDimKey =
AffectedUser.UserDimKey

LEFT OUTER JOIN WorkItemAssignedToUserFactvw ON
    WI.WorkItemDimKey =
WorkItemAssignedToUserFactvw.WorkItemDimKey
    AND WorkItemAssignedToUserFactvw.DeletedDate IS NULL
LEFT OUTER JOIN UserDimvw AssignedTo ON
    WorkItemAssignedToUserFactvw.WorkItemAssignedToUser_UserDimKey =
AssignedTo.UserDimKey

LEFT OUTER JOIN WorkItemCreatedByUserFactvw ON
    WI.WorkItemDimKey = WorkItemCreatedByUserFactvw.WorkItemDimKey
    AND WorkItemCreatedByUserFactvw.DeletedDate IS NULL
LEFT OUTER JOIN UserDimvw CreatedBy ON
    WorkItemCreatedByUserFactvw.WorkItemCreatedByUser_UserDimKey =
CreatedBy.UserDimKey

LEFT OUTER JOIN IncidentResolvedByUserFactvw ON
    I.IncidentDimKey =
IncidentResolvedByUserFactvw.IncidentDimKey
    AND IncidentResolvedByUserFactvw.DeletedDate IS NULL
```

```
LEFT OUTER JOIN UserDimvw ResolvedBy ON
    IncidentResolvedByUserFactvw.TroubleTicketResolvedByUser_
UserDimKey
y = ResolvedBy.UserDimKey

LEFT OUTER JOIN IncidentStatusvw IStatus ON
    I.Status_IncidentStatusId = IStatus.IncidentStatusId
LEFT OUTER JOIN DisplayStringDimvw IStatusDS ON
    IStatus.EnumTypeId = IStatusDS.BaseManagedEntityId
    AND IStatusDS.LanguageCode = 'ENU'

LEFT OUTER JOIN IncidentImpactvw IImpact ON
    I.Impact_IncidentImpactId = IImpact.IncidentImpactId
LEFT OUTER JOIN DisplayStringDimvw IImpactDS ON
    IImpact.EnumTypeId = IImpactDS.BaseManagedEntityId
    AND IImpactDS.LanguageCode = 'ENU'

LEFT OUTER JOIN IncidentUrgencyvw IUrgency ON
    I.Urgency_IncidentUrgencyId = IUrgency.IncidentUrgencyId
LEFT OUTER JOIN DisplayStringDimvw IUrgencyDS ON
    IUrgency.EnumTypeId = IUrgencyDS.BaseManagedEntityId
    AND IUrgencyDS.LanguageCode = 'ENU'

LEFT OUTER JOIN IncidentSourcevw ISource ON
    I.Source_IncidentSourceId = ISource.IncidentSourceId
LEFT OUTER JOIN DisplayStringDimvw ISourceDS ON
    ISource.EnumTypeId = ISourceDS.BaseManagedEntityId
    AND ISourceDS.LanguageCode = 'ENU'

LEFT OUTER JOIN IncidentTierQueuesvw ITierQueue ON
    I.TierQueue_IncidentTierQueuesId =
ITierQueue.IncidentTierQueuesId
LEFT OUTER JOIN DisplayStringDimvw ITierQueueDS ON
    ITierQueue.EnumTypeId = ITierQueueDS.BaseManagedEntityId
    AND ITierQueueDS.LanguageCode = 'ENU'

LEFT OUTER JOIN IncidentClassificationvw IClassification ON
    I.Classification_IncidentClassificationId =
IClassification.IncidentClassificationId
LEFT OUTER JOIN DisplayStringDimvw IClassificationDS ON
    IClassification.EnumTypeId =
IClassificationDS.BaseManagedEntityId
    AND IClassificationDS.LanguageCode = 'ENU'
```

```

LEFT OUTER JOIN IncidentResolutionCategoryvw
IResolutionCategory ON
    I.ResolutionCategory_IncidentResolutionCategoryId =
IResolutionCategory.IncidentResolutionCategoryId
LEFT OUTER JOIN DisplayStringDimvw IResolutionCategoryDS ON
    IResolutionCategoryDS.EnumTypeId =
IResolutionCategoryDS.BaseManagedEntityId
    AND IResolutionCategoryDS.LanguageCode = 'ENU'
WHERE
    I.IsDeleted = 0

```

Extending the Service Manager Data Warehouse code

The following is the code:

```

<ManagementPack ContentReadable="true" SchemaVersion="2.0"
OriginalSchemaVersion="1.1" xmlns:xsd="http://www.w3.org/2001/
XMLSchema" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <Manifest>
    <Identity>
      <ID>Custom.MobilePhone.Library</ID>
      <Version>1.0.0.0</Version>
    </Identity>
    <Name>Custom.MobilePhone.Library</Name>
    <References>
      <Reference Alias="System">
        <ID>System.Library</ID>
        <Version>7.5.8501.0</Version>
        <PublicKeyToken>31bf3856ad364e35</PublicKeyToken>
      </Reference>
      <Reference Alias="Console">
        <ID>Microsoft.EnterpriseManagement.ServiceManager.UI.Console</
ID>
        <Version>7.5.1561.0</Version>
        <PublicKeyToken>31bf3856ad364e35</PublicKeyToken>
      </Reference>
      <Reference Alias="Authoring">
        <ID>Microsoft.EnterpriseManagement.ServiceManager.
UI.Authoring</ID>
        <Version>7.5.1561.0</Version>
        <PublicKeyToken>31bf3856ad364e35</PublicKeyToken>
      </Reference>
      <Reference Alias="DWBase">
        <ID>Microsoft.SystemCenter.Datawarehouse.Base</ID>
        <Version>7.5.1561.0</Version>

```

```

    <PublicKeyToken>31bf3856ad364e35</PublicKeyToken>
  </Reference>
</References>
</Manifest>
<TypeDefinitions>
  <EntityTypes>
    <ClassTypes>
      <ClassType ID="Custom.MobilePhone" Accessibility="Public"
Abstract="false" Base="System!System.ConfigItem" Hosted="false"
Singleton="false" Extension="false">
        <Property ID="MobilePhoneID" Type="string"
AutoIncrement="true" Key="true" CaseSensitive="false" MaxLength="256"
MinLength="0" Required="false" Scale="0" DefaultValue="MP{0}" />
        <Property ID="SerialNumber" Type="string"
AutoIncrement="false" Key="false" CaseSensitive="false"
MaxLength="256" MinLength="0" Required="false" Scale="0" />
        <Property ID="Manufacturer" Type="enum"
AutoIncrement="false" Key="false" CaseSensitive="false"
MaxLength="256" MinLength="0" Required="false" Scale="0" EnumType="Mob
ilePhoneManufacturerEnum" />
      </ClassType>
    </ClassTypes>
    <RelationshipTypes>
      <RelationshipType ID="MobilePhoneOwnedByUser"
Accessibility="Public" Abstract="false" Base="System!System.
Reference">
        <Source ID="MobilePhoneOwnedByUserSource" MinCardinality="0"
MaxCardinality="2147483647" Type="Custom.MobilePhone" />
        <Target ID="MobilePhoneOwnedByUserTarget" MinCardinality="0"
MaxCardinality="1" Type="System!System.User" />
      </RelationshipType>
    </RelationshipTypes>
    <EnumerationTypes>
      <EnumerationValue ID="MobilePhoneManufacturerEnum"
Accessibility="Public" />
    </EnumerationTypes>
  </EntityTypes>
</TypeDefinitions>
<Categories>
  <Category ID="Custom.MobilePhone.Library.Category"
Value="Console!Microsoft.EnterpriseManagement.ServiceManager.
ManagementPack">
    <ManagementPackName>Custom.MobilePhone.Library</
ManagementPackName>
    <ManagementPackVersion>1.0.0.0</ManagementPackVersion>
  </Category>

```



```
<Category ID="MobilePhoneManufacturerEnum.
EnumerationViewTasks.Category" Target="MobilePhoneManufacturerEnum"
Value="Authoring!Microsoft.EnterpriseManagement.ServiceManager.
UI.Authoring.EnumerationViewTasks" />
<Category ID="MobilePhoneManufacturerEnum.VisibleToUser.Category"
Target="MobilePhoneManufacturerEnum" Value="System!VisibleToUser" />
</Categories>
<Warehouse>
  <Outriggers>
    <Outrigger ID="CustomMobilePhoneManufacturer"
Accessibility="Public">
      <Attribute ID="MobilePhoneManufacturer"
PropertyPath="$Context/Property[Type='Custom.MobilePhone']/
Manufacturer$" />
    </Outrigger>
  </Outriggers>
  <Dimensions>
    <Dimension ID="CustomMobilePhoneDim" Accessibility="Public"
Target="Custom.MobilePhone" InferredDimension="true" HierarchySupport=
"IncludeExtendedClassProperties" Reconcile="true" />
  </Dimensions>
  <Facts>
    <RelationshipFact ID="CustomMobilePhoneOwnedByUserFact"
Accessibility="Public" Domain="DWBase!Domain.ConfigurationManagement"
TimeGrain="Daily" SourceType="Custom.MobilePhone" SourceDimension="CustomMobilePhoneDim">
      <Relationships RelationshipType="MobilePhoneOwnedByUser" TargetDimension="DWBase!UserDim" />
    </RelationshipFact>
  </Facts>
</Warehouse>
<LanguagePacks>
  <LanguagePack ID="ENU" IsDefault="true">
    <DisplayStrings>
      <DisplayString ElementID="Custom.MobilePhone.Library">
        <Name>Custom Mobile Phone Library</Name>
      </DisplayString>
      <DisplayString ElementID="Custom.MobilePhone">
        <Name>Mobile Phone</Name>
        <Description></Description>
      </DisplayString>
    </DisplayStrings>
  </LanguagePack>
</LanguagePacks>
```

```
</DisplayString>
  <DisplayString ElementID="Custom.MobilePhone"
SubElementID="Manufacturer">
  <Name>Manufacturer</Name>
</DisplayString>
  <DisplayString ElementID="Custom.MobilePhone"
SubElementID="MobilePhoneID">
  <Name>Mobile Phone ID</Name>
</DisplayString>
  <DisplayString ElementID="Custom.MobilePhone"
SubElementID="SerialNumber">
  <Name>Serial Number</Name>
</DisplayString>
  <DisplayString ElementID="MobilePhoneManufacturerEnum">
  <Name>Mobile Phone Manufacturer</Name>
</DisplayString>
  <DisplayString ElementID="MobilePhoneOwnedByUser">
  <Name>Mobile Phone Owned by User</Name>
</DisplayString>
  <DisplayString ElementID="MobilePhoneOwnedByUser" SubElementID
="MobilePhoneOwnedByUserSource">
  <Name>Mobile Phone</Name>
</DisplayString>
  <DisplayString ElementID="MobilePhoneOwnedByUser" SubElementID
="MobilePhoneOwnedByUserTarget">
  <Name>Owned by User</Name>
</DisplayString>
</DisplayStrings>
</LanguagePack>
</LanguagePacks>
</ManagementPack>
```


Index

A

- Active Alerts report**
 - creating 246-251
- Active Directory account user**
 - read access, creating 164-166
 - read access, delegating 164-166
- Active Directory group**
 - for reporting role 84
- Alert database schema**
 - about 131
 - Alert.vAlertDetail view 132
 - Alert.vAlertParameter view 133
 - Alert.vAlertResolutionState view 133
 - Alert.vAlert view 131, 132
- alert reports**
 - alert dataset 131
 - creating 131-136
 - Report Builder 131
 - working 137
- Application Performance Monitoring (APM)**
 - dataset 123**
- availability reports**
 - creating 150-157
 - datasets 154
 - datasets, creating with parameters 157
 - report layout, designing 158
 - state dataset 151, 157
 - working 157
- B**
- backup status report, DPM**
 - creating, community template used 186-190
- Business Intelligence (BI) 278**

C

- Chargeback reports**
 - analyzing, System Center 2012 R2 data
 - used 265, 266
 - data, analyzing 274-276
 - price sheets, creating 273, 274
 - URL 270
- charts**
 - drillthrough reports, using 257
- ClickOnce**
 - installing 42-44
 - requisites 39
- Client Monitoring dataset 123**
- clouds**
 - creating, in VMM 267
 - virtual machines, provisioning 268, 269
- community blogs**
 - URL 313, 314
- community template**
 - used, for creating backup status report 186-190
- Configuration Item report, Service Manager**
 - creating 204-209
- Configuration Manager custom reports**
 - company logo, inserting 106, 107
 - creating 92-102
 - datasets, creating 94-97
 - data source, creating 94-97
 - description, adding with text box 102-106
- Configuration Manager database schema**
 - exploring 76-82
 - reporting schema, exploring with MSSMS 79
 - reporting schema, exploring with SCCM console 80, 81

Configuration Manager reports, out-of-the-box

- access, delegating 83-87
- Active Directory group, for reporting role 84
- Active Directory user group, adding 86
- reporting scope, configuring 86
- Report Only security scope, creating 84

Configuration Manager, System Center components 20**Coretech blog, Kent Agerlund**

- URL 88

cube processing 193**Curah!**

- SQL Server, URL 315

custom database objects, Service Manager reports

- creating 203
- examples 203

D**dashboard-like report, Service Manager reports**

- creating, operational database used 218-222

dashboards

- about 9-11
- Active Alerts report, creating 246-249
- combined dashboard, creating 250, 251
- creating, from basic reports 240
- incident management dashboard 10
- integrating, in Service Manager console 223
- Open Incidents report, creating 240-245
- static dashboards 9

data

- accessing, Microsoft Excel query code used 317
- analyzing 274-276

Data Analysis Expressions (DAX) 298**Data Analysis Syntax (DAX) 282****data inputs**

- dependent data inputs, optimizing 4-6
- dependent data inputs, planning 4-6
- report inputs, optimizing 6

Data Management Gateway 310**data model**

- creating, with PowerPivot 298-305
- design 300-302

Data Protection Manager (DPM) database 163**Data Protection Manager, System Center components 20, 21****dataset query**

- creating, for DPM agent status report 168-175

datasets, availability reports

- DSState 154
- PSMonitors 154
- PSObjects 154
- PSTypes 154

datasets, OperationsManagerDW schema 123**datasets, Orchestrator runbook reports**

- DSActivities 235
- DSDiagram 235
- DSInstances 235
- DSParameters 235
- DSRunbook 235
- PSRunbooks 235

datasets, Report Builder

- creating 66-70

data sources

- creating 60
- embedded data source, creating 60-63
- shared data source, creating 64-66

Data Warehouse 117**dimensions, Service Manager****Data Warehouse 199, 200****disk space report**

- creating 176-182

DPM agent status report

- byte values, converting into GB values 175, 176
- byte values, converting into MB values 175, 176
- dataset query, creating 168-175

DPM agent version

- creating 176-182
- headers background color, changing 183
- size indicator bar, adding to used space column 183-185
- title, color changing 182, 183
- title, font changing 182, 183

DPM database views

URL 176

DPM reporting environment

management pack, URL 190

preparing 164

shared data source, creating 167, 168

URL 186

DPM Report Manager

delegation, performing 166, 167

folders, creating 166, 167

drillthrough reports

Incident Details report, creating 252-255

Open Incidents report, linking to Incident

Details report 256, 257

using, in charts 257

working with 252

DWDDataMart database

about 192

accessing, Microsoft Excel used 224-226

E

embedded data source

creating 60-63

environment, for authoring reports

preparing 129

shared data source, creating 129, 130

Event database schema

Event.vEventDetail view 139

Event.vEventParameter view 140

Event.vEventRule view 140

Event.vEvent view 138

Event Levels shared dataset

creating 259-264

event reports

creating 138-143

event dataset 138

working 144

Events Last 30 Days report

creating 261-264

Event.vEvent view

about 138

vEventCategory 138

vEventChannel 138

vEventLevel 138

vEventLoggingComputer 138

vEventPublisher 139

vEventUserName 139

Excel

mocking up 9

extract, transform, and load (ETL) 192

F

facts, Service Manager Data Warehouse 200

folders, DPM Report Manager

creating 166, 167

H

highlight reporting 12, 13

I

Incident Details drillthrough report

creating 252-255

Incident Details report

Open Incidents report, linking to 256, 257

J

JDI (Just Do It) 51

K

Key Performance Indicators (KPIs) 298

L

linked report, Service Manager reports

about 198

creating 198

M

managed entities, OperationsManagerDW

schema

about 124

vManagedEntity 124

vManagedEntityType 124, 125

vManagementGroup 127

vManagementPack 125

vRelationship 126

MDX (Multidimensional Expression) query

language 215

Microsoft

tools 37

Microsoft Cloud services

used, for sharing System Center data 309-312

used, for visualizing System Center data 309-312

Microsoft Excel

configuring, for System Center data analysis 278-283

PowerPivot, enabling 280-282

Power View, enabling 280-282

used, for accessing DWDDataMart database 224, 225

Microsoft Report Builder. *See* **Report Builder**

Microsoft SQL Server Community Projects and Samples

URL 314

Microsoft SQL Server Management Studio (MSSMS)

about 20, 76, 170

used, for exploring reporting schema 79

Microsoft SQL Server Reporting Services. *See* **SQL Server Reporting Services (SSRS)**

Mobile Device Management (MDM) 77

N

NOLOCK option 251

NotePad++ 223

O

OLAP cubes

used, for creating Service Manager reports 211-214

Opalis Survival Guide

Microsoft TechNet Wiki page 315

Open Incidents report

creating 240-251

linking, to Incident Details report 256, 257

operational database

used, for creating Service Manager reports 215-217

OperationsManagerDW schema

about 123

datasets 123

managed entities 124

working 128

Operations Manager Event Collection Rules 138

Operations Manager reports

accessing 118, 119

alert reports, creating 131

availability reports, creating 150

event reports, creating 138

favorite reports, creating 120

folder for custom reports, creating 118

performance reports, creating 144

publishing 120

scheduling 120-122

using 118

Operations Manager, System

Center 2012 R2 269, 270

Operations Manager, System Center components 22, 23

Orchestrator runbook reports

creating 232-238

datasets 235

views 233, 234

Orchestrator, System Center

components 21

outriggers, Service Manager Data Warehouse 201, 202

P

parameters

working with 258, 259

Perf database schema

about 144

Perf.vPerfDaily 145

Perf.vPerfHourly 145

Perf.vPerfRaw 146

performance reports

creating 144-150

custom view, creating 150

Excel Pivot table 150

performance dataset 144, 150

security delegation 150

working 150

Perf.vPerfDaily and Perf.vPerfHourly views

AverageValue 145

MaxValue 145

MinValue 145
StandardDeviation 145

Perf.vPerfRaw view 146

Power BI Excel interface

exploring 286-289
overview 286-289

Power Map

about 278
add-in, installing 279
URL 278
used, for analysis 305-309
visualization, creating 307, 308

PowerPivot

about 278
enabling, in Microsoft Excel 280-283
URL 278
used, for creating data model 298-304
visualization 302-304

Power Query

about 277
add-in, installing 279
URL 278
used, for connecting System Center data sources 283-286, 296

Power View

about 278
enabling, in Microsoft Excel 280-282
URL 278
used, for analysis 305-309
visualization, creating 306, 307

price sheets, Chargeback reports

creating 273, 274

proactive reporting 12

Publisher or Content Manager

permissions 120

Q

query

extending, with data from other queries 290-294
functions, adding 294-296

Question and Answer (Q&A) 310

R

reactive reporting 12

read access

creating, to Active Directory account user 164-166
delegating, to Active Directory account user 164-166

Report Builder

ClickOnce 38
ClickOnce, installation 42, 44
ClickOnce, requisites 39
dataset, creating 66-70
installation, planning 38, 39
installation planning, requisites URL 39
installing 40
interface, navigating 46
online documentation, URL 45
report, creating 66, 71-73
requisites 38, 39
Standalone installation 38-42
URL 38, 314

Report Builder interface

about 45
Built-in fields 46
Connection string 46
dataset 45
Data source 45
Home tab 48
Images 46
Insert tab 48
navigating 45-51
Parameters 46
Query Designer 45
Report authors 46
Report Builder 45
Report consumers 46
Report Manager 45
Report parts 45
Report rendering 46
Report types 45
View tab 48

report data filter options

vMonitor 128
vRule 128

reporting

- about 2
- design, documenting 14, 15
- goals 2, 3
- highlight reporting 12, 13
- input, optimizing 6
- life cycle, stages 3, 4
- proactive reporting 12
- reactive reporting 12
- SCCM reporting environment, preparing for 88
- targeting, to decision stake holders 11

reporting environment

- organization folders access, delegating 56-59
- organization, planning 52
- organizing 52
- report delegation roles, planning 54
- reporting organization folders, creating 54-56
- report manager folders, planning 52, 53

reporting schema

- exploring, MSSMS used 79
- exploring, SCCM console used 80, 81

Reporting Services Tutorials (SSRS)

- URL 314

report inputs

- optimizing 6

Report Manager website

- delegation, verifying in 87, 88

Report Only security scope

- creating 84-86

report outputs

- planning 7, 8

report subscriptions

- preparing for 73, 74

reportuser database role 203

Resource Explorer 80

role-based security, applying to

custom reports

- about 108
- Configuration Manager administrative users dataset, creating 111
- datasets, modifying to role-based 113, 114
- report user SID, obtaining as parameter 109-111

- user SID parameter, creating 111-113

- working 114, 115

ROUND function

- URL 176

RTM (Read The Manual) 51

S

SCCM

- about 75
- used, for exploring reporting schema 80, 81

SCCM database views

- references 82

SCCM reporting environment

- preparing, for reporting 88
- SCCM Report Manager folders, creating 89
- shared data source, creating 91, 92

SCCM Report Manager folders

- creating 89, 90

SCCM role-based security

- references 115

SCOM Connector

- configuring 271-273

security delegation

- URL 60

Self-service BI, Microsoft Excel

- prerequisites 278

Service Manager console

- dashboard, integrating in 223

Service Manager Data Warehouse

- attributes, for dimensions 229
- attributes, for relationship fact 230
- code, extending 320
- data, transferring to databases 192
- extending 227-231

Service Manager Data Warehouse data mart

- about 199
- custom database objects, creating for reporting 203
- database permissions, required for reporting 203
- dimensions 199, 200
- facts 200
- outriggers 201
- reference link 202
- working 202

Service Manager OLAP cubes, out-of-the-box
using 193-196

Service Manager reports
accessing, from browser 196, 197
Configuration Item report, creating 207-209
creating, OLAP cubes used 211-214
creating, operational database used 215-217
favorite reports, creating 198
linked reports, creating 198
Service Level reports, creating 209
using 193, 194
Work Item report, creating 205-209

Service Manager, System Center 2012 R2
about 270

Chargeback management pack,
importing 270, 271
SCOM Connector, configuring 271-273

**Service Manager, System Center
components 23, 24**

shared datasets

Event Levels shared dataset,
creating 259, 260
Events Last 30 Days report,
creating 261-264
multiple values, selecting for
parameter 264, 265
working with 258, 259

shared data source

creating 64-66, 129, 130
creating, for DPM reports 167, 168
creating, for SCCM reports 91

Social network resources

URL 315

SQL and reporting websites

URL 314

SQL community

URL 314

SQL Database Engine

Microsoft TechNet Wiki page 315

SQL queries

building 31-35

SQL Reporting Services (SSRS)

interface with System Center 27-31

Standalone, Report Builder

installing 40-42

State database schema

HealthServiceUnavailableMilliseconds 151
InDisabledStateMilliseconds 151
InPlannedMaintenanceMilliseconds 151
InRedStateMilliseconds 151
InUnplannedMaintenanceMilliseconds 151
InYellowStateMilliseconds 151
State.vStateRaw view 152

state dataset

about 151
State.vStateDaily view 151
State.vStateHourly view 151

static dashboards 9

Structured Query Language (SQL) 123, 202

System Center

Configuration Manager 28
Configuration Manager, URL 36
Data Protection Manager 30
features 28
input channel, for data 28
Operations Manager 29
reporting interface 28
Service Manager 30
SQL Reporting Services interface,
interface with 27-31
storage component 28

System Center 2012 R2 data

clouds, creating in VMM 267
Operations Manager 266-270
Service Manager 266, 270
Service Manager Data Warehouse 266
used, for analyzing Chargeback
reports 265, 266
Virtual Machine Manager 266
virtual machines, provisioning
to clouds 267, 269

System Center components

Configuration Manager 18-20
Data Protection Manager 18-21
multiple databases, components with 26
online documentation, URL 27
Operations Manager 18-23
Orchestrator 18-21
reporting schema 18, 19
Service Manager 18-24
Virtual Machine Manager 18-25

System Center Configuration

Manager. *See* **SCCM**

System Center data

Microsoft Excel, configuring 278-283

sharing, with Microsoft Cloud

services 309-312

visualizing, with Microsoft Cloud

services 309-312

System Center data sources

connecting, with Power Query 283-286, 296

System Center Operations

Manager (SCOM) 117, 118

System Center Orchestrator 193

System Center Service Manager 192

System Center Service Manager Authoring

Tool 227

System Center suite

(System Center product) 18

V

vendor reporting visualization

URL 314

view category type, SCCM 2012

Application Management 77

Client Deployment 77

Client Status 77

Collection 77

Compliance Settings 77

Content Management 77

Discovery 77

Endpoint Protection 77

Inventory 77

Migration 77

Mobile Device Management 77

Network Access Protection 77

Operating System Deployment 77

Out of Band Management 77

Power Management 77

Query 78

Schema 78

Security 78

Site Administration 78

Software Metering 78

Software Updates 78

Status and Alert 78

URL 79

Wake On LAN 78

Virtual Machine Manager (VMM) reports

prerequisites 158, 159

using 158-161

Virtual Machine Manager (VMM), System

Center components 24-26

Virtual Machine Manager (VMM), System

Center 2012 R2

about 266

clouds, creating 267

provisioning, to clouds 267-269

Visual C++ 2013 Runtime Libraries (x86) 280

vManagedEntity view 124

vManagedEntityType view 124, 125

vManagementGroup view 127

vManagementPack view 125

vRelationship view 126, 127

W

Windows Management

Instrumentation (WMI) 76

WiseOwlTutorials (SSRS)

URL 314

Work Item report, Service Manager

creating 204-209



Thank you for buying Microsoft System Center Reporting Cookbook

About Packt Publishing

Packt, pronounced 'packed', published its first book, *Mastering phpMyAdmin for Effective MySQL Management*, in April 2004, and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution-based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern yet unique publishing company that focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website at www.PacktPub.com.

About Packt Enterprise

In 2010, Packt launched two new brands, Packt Enterprise and Packt Open Source, in order to continue its focus on specialization. This book is part of the Packt Enterprise brand, home to books published on enterprise software – software created by major vendors, including (but not limited to) IBM, Microsoft, and Oracle, often for use in other corporations. Its titles will offer information relevant to a range of users of this software, including administrators, developers, architects, and end users.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, then please contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.



System Center 2012 R2 Virtual Machine Manager Cookbook

Second Edition

ISBN: 978-1-78217-684-8 Paperback: 428 pages

Over 70 recipes to help you design, configure, and manage a reliable and efficient virtual infrastructure with VMM 2012 R2

1. Create, deploy, and manage datacenters and private and hybrid clouds with hybrid hypervisors using VMM 2012 R2.
2. Integrate and manage fabric (compute, storages, gateways, and networking), services and resources, and deploy clusters from bare metal servers.



Microsoft System Center Configuration Manager Advanced Deployment

ISBN: 978-1-78217-208-6 Paperback: 290 pages

Design, implement, and configure System Center Configuration Manager 2012 R2 with the help of real-world examples

1. Learn how to design and operate Configuration Manager 2012 R2 sites.
2. Explore the power of Configuration Manager 2012 R2 for managing your client and server estate.
3. Discover up-to-date solutions to real-world problems in System Center Configuration Manager administration,

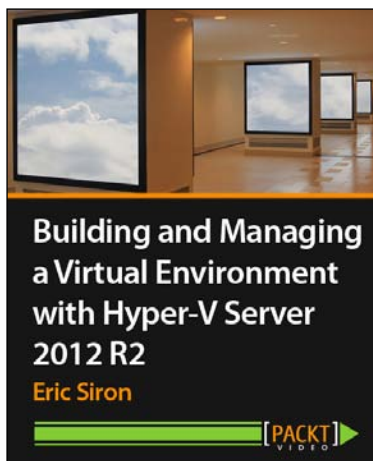


Microsoft System Center 2012 Service Manager Cookbook

ISBN: 978-1-84968-694-5 Paperback: 474 pages

Learn how to configure and administer System Center 2012 Service Manager and solve specific problems and scenarios that arise

1. Practical cookbook with recipes that will help you get the most out of Microsoft System Center 2012 Service Manager.
2. Learn the various methods and best practices administrating and using Microsoft System Center 2012 Service Manager.
3. Save money and time on your projects by learning how to correctly solve specific problems and scenarios that arise while using System Center Service Manager.



Building and Managing a Virtual Environment with Hyper-V Server 2012 R2 [Video]

ISBN: 978-1-78217-698-5 Duration: 03:30 hours

Build, deploy, and manage Hyper-V in failover cluster environments

1. Configure node computers for participation in a Hyper-V cluster.
2. Tackle the complicated subjects of storage and networking in a Hyper-V cluster.
3. Maximize uptime for the services provided by your virtual machines.